Predicting Movie Success: A Deep Learning Approach

Liuxinhao Gao, Akshay Navada, Tanvi Saini, Mike Shu, Nina Wu

Motivation and Data Understanding

The entertainment industry, particularly film production, faces high levels of risk and uncertainty when determining the potential success of a movie. The ability to predict whether a movie will be commercially successful before its release is valuable not only for studios and production companies but also for distributors, marketers, and investors. Having a reliable tool to forecast a movie's success can influence budget allocation, marketing strategies, and overall resource management.

This project aims to build a deep-learning model that predicts the success of a movie based on its plot. The features taken into consideration include textual data (such as movie descriptions) along with categorical attributes (genre, keywords, etc.). The model will leverage advanced natural language processing (NLP) techniques, specifically word embeddings, to process the textual information. Word2Vec, a widely-used pre-trained model, will convert words in the descriptions into vector representations, making them comprehensible to a deep neural network. This model's prediction can assist movie studios in making data-driven decisions, reducing risks associated with movie production.

Data Description and Sources:

The data used for this project includes several key variables:

- **Textual Features**: These include descriptions or summaries of the movie script, associated keywords, director, casting director, and producer.
- **Categorical Features**: This encompasses the movie's genres

The primary data sources for this project are movie databases available on Kaggle at https://www.kaggle.com/datasets/gufukuro/movie-scripts-corpus.

Relevance in Practice and Business Applications:

A reliable movie success predictor could greatly benefit production companies by helping them predict a movie's performance before development. This would allow for better decisions regarding production budgets, target markets, and promotional activities. The model would assist script analysts in production houses who evaluate numerous scripts daily, identifying promising projects. It could also be extended to other entertainment forms like TV shows, video games, or streaming content.

Data Preparation

1. Text Data Preprocessing:

The movie plots are first tokenized, meaning they are broken down into words. These words are then transformed into vector representations using Word2Vec. Word2Vec is a pre-trained model that maps words into a continuous vector space, capturing semantic similarities between words. By using this model, the descriptions of movies can be turned into a numerical format that the neural network can understand.

First, textual features such as plot, keywords, producers, directors, casting directors, and taglines are processed using Word2Vec to obtain vector representations. These textual features are converted into fixed-length numerical vectors by averaging the word embeddings for each word in the respective columns.

For categorical data, the genres column is processed using a **MultiLabelBinarizer** (mlb) to convert the genres into binary vectors, indicating the presence of each genre for each movie. Additionally, the year column, which contains numerical values, is standardized using a scaler to ensure consistent scaling of the data.

This combination of processed textual, categorical, and numerical features creates the final input data, ready to be used in a deep learning model.

2. Train - Validation - Test Split:
In this step, we split the dataset into three subsets: 60% for training (X_train, y_train_ratings), 20% for validation (X_val, y_val_ratings), and 20% for testing (X_test, y_test_ratings) by first splitting the data into training and temporary sets and then further dividing the temporary set into validation and test sets using a 50-50 split.

Modeling

Model Architecture:

We create a MovieSuccessPredictor Model which is a deep feedforward neural network designed for predicting IMDb ratings (from 1 to 10) based on various features like the movie's plot, keywords, genres, and other attributes. It handles high-dimensional input features (1829 dimensions) and is optimized for a batch size of 32. The architecture employs techniques such as dropout and batch normalization for stability and regularization.

Work Flow:

Input → [FC1 → BatchNorm1 → ReLU] → [FC2 → BatchNorm2 → ReLU → Dropout2] → [FC3 → ReLU] → Output

Layer	**Type**	**Input**	**Output**
Input Layer	Fully Connected (fc1)	1829	256
BatchNorm1	Batch Normalization	256	256
Activation 1	ReLU	256	256
Hidden Layer 2	Fully Connected (fc2)	256	128
BatchNorm2	Batch Normalization	128	128
Activation 2	ReLU	128	128
Hidden Layer 3	Fully Connected (fc3)	128	64
Activation 3	ReLU	64	64
Output Layer	Fully Connected	64	1

Hyperparameters used:

We used validation data to fine-tune the model and found the following hyperparameters to perform the best for our model.

Optimizer	Adam
Loss Function	MSE Loss
Learning Rate	0.001
Batch Size	32
Number of Epochs	300

Implementation

The implementation of the Movie Success Predictor involved several critical steps, beginning with data preprocessing and culminating in training a deep learning model. The following highlights the key aspects of the implementation:

1. **Framework and Libraries**:
   We used PyTorch for model development due to its flexibility. Gensim's Word2Vec was employed to convert movie descriptions into word embeddings, while NumPy and pandas handled data manipulation.
2. **Data Integration**:
   Textual data (e.g., plot, keywords) was converted into vector representations using

Word2Vec, and MultiLabelBinarizer encoded categorical features like genres. The year feature was scaled using **StandardScaler**.

3. **Model Training**:

   The model is a fully connected neural network with three hidden layers (256 → 128 → 64), incorporating dropout (0.6) to prevent overfitting and batch normalization for training stability. It was trained using the Adam optimizer with a learning rate of 0.001 and MSE as the loss function over 300 epochs with a batch size of 32.

4. **Optimization and Fine-Tuning**:

   Hyperparameter tuning included experimenting with:

   - The number of layers and neurons.
   - Dropout rates to reduce overfitting.
   - Batch normalization for smoother training.

The architecture was refined iteratively to balance performance and efficiency.

5. **Challenges and Solutions**:

   - **Data Alignment**: Ensured that embeddings and other features aligned correctly by validating input formats and shapes.input formats and validating shapes during the integration process.
   - **Overfitting**: Applied dropout layers and reduced model complexity to prevent the model from memorizing training data.
   - **Early Stopping**: Early stopping was implemented to halt training when validation loss did not improve after a set number of epochs (patience). This prevented overfitting and saved computational resources.

Results and Evaluation

The performance of the Movie Success Predictor was evaluated using Mean Squared Error (MSE) and Median Error Rate (MedER) for both training and validation sets. MSE measures prediction accuracy, while MedER helps assess robustness by focusing on the median error. The model's ability to generalize was tracked through validation loss and MedER, with early stopping applied to prevent overfitting if validation loss didn't improve for 25 consecutive epochs.

Performance Plots:

- **Loss Curves**: Training and validation loss decreased steadily, showing the model learned effectively while avoiding overfitting.
- **MedER Curves**: Both training and validation MedER decreased and stabilized, indicating improved generalization.

Comparison with Benchmarks:

The Movie Success Predictor (deep learning model) was compared to a linear regression model to assess its effectiveness in predicting the IMDb rating.

Linear Regression Model (Baseline model):

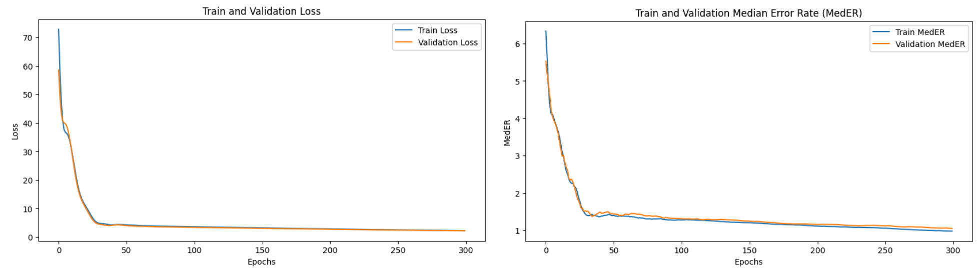
- **Preprocessing**: Used **TF-IDF** for textual data and **label encoding** for categorical features.
- **Performance**: While it performed adequately, the model struggled to capture the complex relationships between features and had higher MSE and MedER compared to the deep learning model.

Deep Learning Model:

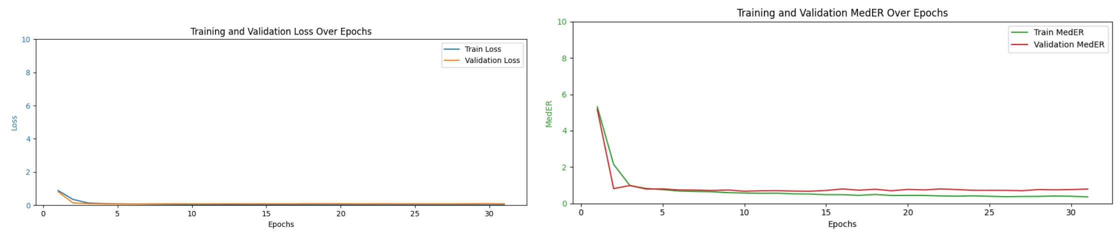
- **Metrics**: Evaluated using MSE and MedER, with early stopping implemented to prevent overfitting. The model effectively learned from complex relationships between textual (Word2Vec embeddings) and categorical features (e.g., genres, directors).
- **Performance**: Achieved higher prediction accuracy with lower MSE and MedER on both training and validation sets.

Visualization:

Plot for linear regression



Plot for Neural network model



By epoch 300, the linear regression model showed a **train MedER** of 1.2686 and **validation MedER** of 1.3160, indicating difficulty in capturing complex relationships. In contrast, the neural network achieved significantly lower and stable MedER values, with **train MedER** of 0.6641 and **validation MedER** of 0.6393 at epoch 31, demonstrating superior accuracy and generalization.

Additionally, the linear regression model had an **MSE loss** of 3.2023, while the neural network outperformed it with an MSE loss of 0.1163, highlighting the network's better performance in modeling the data.

Deployment

Objective:

Create a platform where users can input movie details (e.g., description, genre) and receive predictions about a movie's success in terms of rating, helping studios make informed pre-and-post-production decisions.

Steps:

- **API Development & Hosting**:
  The trained model is saved as a PyTorch .pt file. A **Flask** or **FastAPI** application can be used to process movie data (e.g., description, genre) via a JSON payload and return predictions (e.g., IMDb score, number of ratings). The API can be hosted on a platform like AWS, Google Cloud, or Azure for accessibility.
- **User Interface**:
  A web dashboard can be created to allow users to input movie details and view predictions, making the tool user-friendly for non-technical stakeholders.

Challenges:

- Ensuring quick predictions, even for complex inputs

- Scalability to handle multiple requests simultaneously
- Bias mitigation to avoid favoring specific genres, production companies, or regions
- More training data to refine the model predictions

Example Workflow:

- A script analyst inputs movie details (e.g., description, genre).
- The system sends data to the API, processes it with the trained model, and returns predictions (e.g., IMDb rating and number of ratings).
- The analyst uses these predictions to prioritize scripts for review.

Deployment Outcome:
The model acts as a decision-support tool, reducing guesswork and helping stakeholders make data-driven decisions.

Future Enhancements:
With improved resources, the system can evolve into a comprehensive script analysis tool that offers insights beyond success prediction, such as:

- **Genre Classification**: Automatically categorize scripts by genre
- **Sentiment Analysis**: Gauge emotional engagement
- **Character Analysis**: Evaluate character depth and relationships