Movie rating prediction based on movie description

Ionova Darya

February 2023

Abstract

Movie rating by its description is the process of predicting the rating of a movie based on its text description. This is a challenging problem that requires understanding the context and meaning of the text, as well as accurately categorizing it into a numerical rating system. Natural Language Processing (NLP) techniques can be used to address this problem, including using machine learning algorithms to predict the rating based on the movie description and using sentiment analysis models to analyze the overall sentiment of the description and assign a rating based on that. Link to my project code right here: `https://github.com/fourniaforever/MOVIE-PREDICTING`.

1 Introduction

Movie rating by its description can be a challenging problem because it requires understanding the context and meaning of the text, as well as being able to accurately categorize it into a numerical rating system. Natural Language Processing (NLP) techniques can be used to address this problem.

One approach is to use a machine learning algorithm to predict the rating based on the movie description. This involves training the algorithm on a large dataset of movie descriptions and their corresponding ratings, so that it can learn to recognize patterns and make accurate predictions.

To do this, the text description of the movie needs to be preprocessed using techniques such as tokenization, stemming, and lemmatization. These techniques can help to reduce the complexity of the text and make it easier for the algorithm to analyze.

Next, the algorithm can be trained using a supervised learning approach. This involves feeding the algorithm the preprocessed movie descriptions and their corresponding ratings, and then evaluating how well it performs on a separate test dataset.

Some popular machine learning algorithms that can be used for this task include logistic regression, decision trees, and support vector machines.

Another approach is to use a sentiment analysis model to analyze the movie description and predict its overall sentiment. This can then be used as a proxy for the movie's rating. For example, if the sentiment analysis model determines that the movie description is generally positive, it could be assigned a higher rating than if the sentiment was negative.

Overall, using NLP techniques to predict movie ratings based on their descriptions can be a challenging problem, but with the right approach, it is possible to build accurate models that can help people make informed decisions about which movies to watch

1.1 Team

Ionova Darya prepared this document.

2 Related Work

Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on the interaction between computers and human language. It involves using algorithms and computational techniques to analyze, understand, and generate human language.

NLP has many applications, including language translation, sentiment analysis, information retrieval, speech recognition, and chatbots. Here are some examples of NLP in action:

Language Translation: One of the most common applications of NLP is language translation. Google Translate, for example, uses a combination of statistical and rule-based algorithms to translate text from one language to another.

Sentiment Analysis: NLP can also be used to analyze the sentiment of a piece of text, such as a customer review or social media post. This can help companies gauge customer satisfaction and identify areas for improvement. Amazon, for example, uses sentiment analysis to analyze customer reviews and provide product recommendations.

Information Retrieval: NLP can be used to extract relevant information from large volumes of text. Search engines, for example, use NLP algorithms to understand the meaning of search queries and retrieve relevant results.

Speech Recognition: NLP can be used to convert spoken language into text. Speech recognition technology is used in applications such as virtual assistants (e.g., Siri and Alexa) and dictation software.

Chatbots: NLP is also used to power chatbots, which can simulate human conversation and assist with customer service, scheduling, and other tasks. Many companies use chatbots to provide 24/7 customer support and reduce workload for human agents.

Overall, NLP is a rapidly advancing field with many exciting applications. Its impact can be seen in everyday technologies and has the potential to transform the way we communicate and interact with computers.[**?**]

RNN (Recurrent Neural Network): A type of neural network designed to handle sequential data by allowing the network to have memory of previous inputs.[Miljanovic, 2012] It's commonly used for tasks such as natural language processing, speech recognition, and time-series prediction.[Hyötyniemi, 1996]

CNN (Convolutional Neural Network): A type of neural network that uses convolutional layers to automatically learn spatial hierarchies of features from input data.[Valueva, 2020] It's commonly used for tasks such as image classification, object detection, and segmentation.[Miljanovic, 2012]

Bag of Words: A simple technique for text representation that counts the occurrence of each word in a document and represents the text as a vector of word frequencies.[Michael McTear, 2016] It's commonly used for tasks such as document classification and sentiment analysis.[Harris, 1954]

TF-IDF (Term Frequency-Inverse Document Frequency): A technique for text representation that assigns weights to each word in a document based on its frequency in the document and its rarity in the corpus. [Rajaraman, 2011] It's commonly used for tasks such as information retrieval, text classification, and recommendation systems.[Breitinger, 2015]

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a powerful natural language processing model that has proven to be highly effective in text classification tasks. BERT is a deep learning model that was developed by Google and released in 2018, and it has since become a popular choice for many NLP tasks, including text classification.

One of the key features of BERT is its ability to process bidirectional text, which means that it can consider the context of a word not only from the words that come before it, but also from the words that come after it. This allows BERT to capture more nuanced meanings and semantic relationships between words than previous models that only processed text in one direction.

In text classification tasks, BERT is typically used in a transfer learning approach, where the model is pre-trained on a large amount of text data and then fine-tuned on a specific classification task. During pre-training, BERT learns to predict missing words in sentences by considering the surrounding context, and this allows it to develop a deep understanding of the relationships between words.

To use BERT for text classification, the pre-trained model is fine-tuned on a specific task using a labeled dataset. The fine-tuning process involves adjusting the weights of the model to optimize its performance on the classification task, and it typically requires a smaller amount of labeled data than training a model from scratch.[Kim, 2014]

One of the key benefits of BERT for text classification is its ability to capture contextual information and produce embeddings that are sensitive to the order and position of words in a sentence. This allows BERT to model complex relationships between words and produce more accurate representations of text data compared to traditional bag-of-words approaches. The authors also note that pre-trained BERT models are readily available and can be fine-tuned on task-specific data with relatively small amounts of annotated data, which makes it an attractive option for many text classification applications.[Devesh Maheshwari and Shrivastava, 2021]

Overall, BERT has proven to be highly effective in text classification tasks, achieving state-of-the-art performance on many benchmarks. Its ability to capture nuanced relationships between words and its transfer learning approach make it a powerful tool for NLP practitioners working on text classification problems.[Hongliang Yu and Gao, 2021]

3 Model Description:TF-IDF

1. TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used technique for text data preprocessing in natural language processing (NLP) and information retrieval. It is a statistical measure that reflects the importance of each word in a document corpus. The TF-IDF model is calculated as follows: Term Frequency (TF) measures how often a word appears in a document. It is calculated as the frequency of a word in a document divided by the total number of words in the document. Inverse Document Frequency (IDF) measures how important a word is to the entire document corpus. It is calculated as the logarithm of the total number of documents in the corpus divided by the number of documents containing the word. The TF-IDF score is the product of TF and IDF. It gives higher weight to words that appear frequently in a document and less weight to words that appear frequently in the entire document corpus. The TF-IDF model is used to convert a collection of documents into a matrix of numerical values. Each row of the matrix represents a document, and each column represents a unique word in the entire corpus. The value in each cell is the TF-IDF score of the corresponding word in the corresponding document. The TF-IDF model is often used as a feature extraction technique for machine learning models in NLP, such as text classification, information retrieval, and clustering.

4 Model Description: bag-of-words

2. The bag-of-words model is a common approach used in natural language processing for representing text data. It is a simple way of converting text data into numerical form, which can be used for various machine learning tasks.

In the bag-of-words model, a document is represented as a "bag" (unordered set) of its constituent words, ignoring grammar and word order. The model counts the frequency of each word in the document, creating a vector of word counts that can be used to represent the document. This vector can be normalized by dividing each count by the total number of words in the document or by using other normalization methods.

The bag-of-words model can be used to transform a collection of documents into a matrix, where each row corresponds to a document and each column corresponds to a word in the vocabulary. The values in the matrix represent the frequency of each word in each document. This matrix can be used as input to machine learning algorithms for various tasks, such as sentiment analysis,

$$\mathbf{tf}(t, d) = \frac{f_d(t)}{\max_{w \in d} f_d(w)}$$

$$\mathbf{idf}(t, D) = \ln\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

$$\mathbf{tfidf}(t, d, D) = \mathbf{tf}(t, d) \cdot \mathbf{idf}(t, D)$$

$$\mathbf{tfidf}'(t, d, D) = \frac{\mathbf{idf}(t, D)}{|D|} + \mathbf{tfidf}(t, d, D)$$

$f_d(t) :=$ frequency of term t in document d

$D :=$ corpus of documents

Figure 1: TF-IDF formula.

document classification, and topic modeling.

One of the limitations of the bag-of-words model is that it does not capture the meaning or context of words, as it only considers their frequency. This can result in the loss of information and potentially limit the performance of models trained on this representation. Additionally, the size of the vocabulary can be large, which can lead to sparse data and computational challenges.

5 Model Description: CNN

3. A Convolutional Neural Network (CNN) is a type of neural network that is particularly well-suited for image classification tasks, but can also be used for other types of data, such as audio and text. CNNs use convolutional layers to extract features from the input data, such as edges, shapes, and textures, by applying a set of learnable filters to the input. These filters slide over the input data and perform a dot product operation, producing a feature map that highlights the presence of particular patterns or features in the input. These feature maps are then passed through non-linear activation functions, such as ReLU, to introduce non-linearity into the model. CNNs also typically include pooling layers, which downsample the feature maps by summarizing nearby values, reducing the spatial dimensions of the feature maps while retaining the most

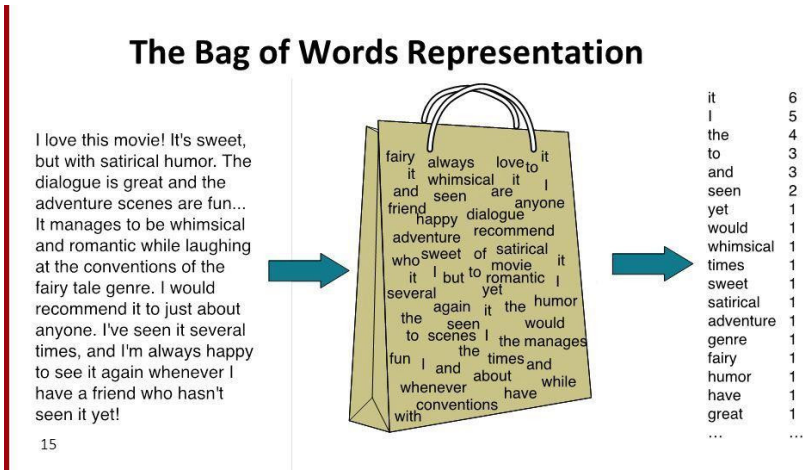

Figure 2: Bag of words representation.

important information. The most common type of pooling layer is max pooling, which selects the maximum value within a given window. After the convolutional and pooling layers, CNNs typically include one or more fully connected layers, which take the flattened feature maps as input and produce the final output, such as a probability distribution over classes for classification tasks. CNNs can be trained using backpropagation and gradient descent, adjusting the weights of the filters and fully connected layers to minimize a loss function, such as cross-entropy loss for classification tasks. CNNs have achieved state-of-the-art results in a variety of computer vision tasks, such as image classification, object detection, and segmentation. They have also been applied to other types of data, such as speech and text, using similar principles.

Lets take a look at architecture.

Embedding layer: This layer is used to map the words in the input text to a high-dimensional vector space, where words with similar meanings are close to each other. The input-dim parameter sets the size of the vocabulary, i.e., the maximum number of words to consider in the text, and output-dim sets the size of the embedding vectors. input-length sets the length of the input sequences (i.e., the maximum number of words in a document).

Conv1D layer: This layer applies a one-dimensional convolution operation to the input sequence. The 128 parameter sets the number of filters (i.e., the number of output channels), and 5 sets the size of the kernel (i.e., the size of the sliding window used for convolution). The activation parameter specifies the activation function to be applied to the output of each filter.

GlobalMaxPooling1D layer: This layer performs a global max pooling operation over the output of the convolutional layer, which means that it selects the maximum value of each feature map (i.e., the output of each filter) and concatenates them into a single vector. This is a way to extract the most salient

features from the output of the convolutional layer.

Dense layer: This layer is a fully connected layer with 64 units and relu activation function, which means that it applies a linear transformation to the input followed by a rectified linear activation function. This layer is used to perform a non-linear transformation of the features extracted by the convolutional layer.

Dropout layer: This layer randomly sets a fraction (0.5) of the input units to 0 at each update during training time, which helps to prevent overfitting.

Dense layer: This is the output layer with a single unit and linear activation function, which means that it applies a linear transformation to the input and produces a real-valued output. In this case, the model is designed for regression, where the output is a continuous value (e.g., a score or a probability). If the problem was a binary classification problem, we would use a sigmoid activation function instead. If it was a multi-class classification problem, we would use a softmax activation function and set the number of units to the number of classes.

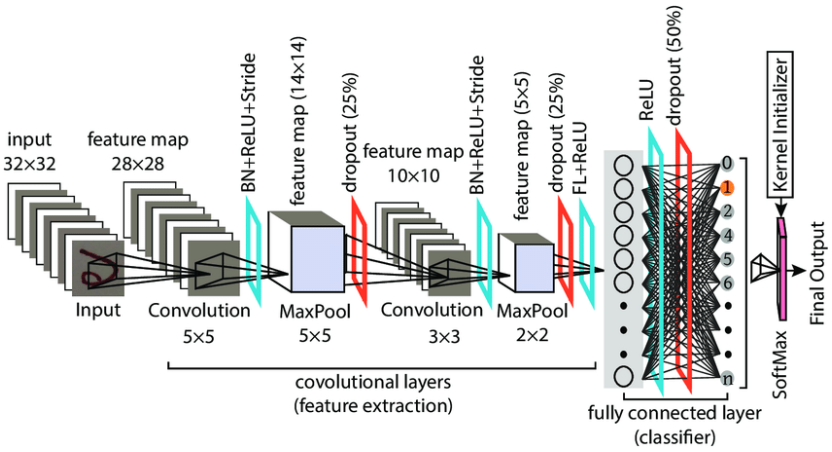

Figure 3: CNN.

6 Model Description: RNN

4. A Recurrent Neural Network (RNN) is a type of neural network that is well-suited for sequence data, such as text, speech, or time-series data.

RNNs use recurrent connections to maintain a memory of past inputs, allowing the network to process sequences of arbitrary length. At each time step, the RNN takes as input the current data point and a hidden state vector that represents the network's memory of past inputs. The network produces an output and updates the hidden state based on the current input and the previous hidden state.

The hidden state at each time step is calculated by applying a non-linear

activation function, such as tanh or ReLU, to a linear combination of the current input and the previous hidden state. This allows the network to learn to capture long-term dependencies and context in the sequence data.

RNNs can be trained using backpropagation through time, which involves unrolling the network over time and calculating the gradients of the loss function with respect to the weights at each time step. This can lead to the problem of vanishing or exploding gradients, which can make it difficult to train RNNs over long sequences.

To address this problem, several variants of RNNs have been developed, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, which use gating mechanisms to selectively update and forget information in the hidden state.

RNNs have been used for a variety of tasks, such as language modeling, machine translation, speech recognition, and time-series prediction. They are particularly well-suited for tasks that require modeling sequential dependencies or handling variable-length input sequences.

The architecture consists of an input layer, an embedding layer, an LSTM layer, a dense layer, and an output layer. The input layer defines the shape of the input data, which is a 1D tensor with length equal to the maximum sequence length. The embedding layer maps each word in the input sequence to a high-dimensional vector space, with the size of the embedding space determined by the embedding-dim parameter. The LSTM layer analyzes the sequence of word embeddings and produces a fixed-size output vector that summarizes the information in the sequence. The dense layer applies a non-linear transformation to the output of the LSTM layer, and the output layer produces a scalar prediction by applying a linear transformation to the output of the dense layer.

The LSTM layer has 128 hidden units, which means that the output vector produced by the LSTM layer has a length of 128. The dense layer has 10 hidden units and uses the ReLU activation function, which means that the output of the dense layer is a 10-dimensional vector with non-negative entries. Finally, the output layer uses a linear activation function, which means that the scalar prediction produced by the output layer is simply a weighted sum of the 10-dimensional output vector produced by the dense layer.

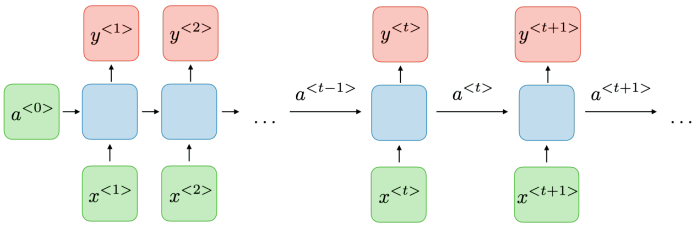

Figure 4: RNN.

7 Model Description: BERT

5. The code loads a pre-trained BERT model and fine-tunes it on a dataset for rating prediction. The input reviews are tokenized using the BERT tokenizer, and the resulting tokenized sentences are passed through the BERT model. The output from the BERT model is then fed through a simple linear layer to produce a single output, which is the predicted rating for the input review. During training, the Mean Square Error (MSE) loss is used as the loss function, and the Adam optimizer is used for optimization. The learning rate of the optimizer is updated using the ExponentialLR scheduler. The training is performed for 4 epochs, and the training and validation losses are recorded for each epoch.

8 Dataset

Movie dataset was presented in [BANIK, 2018] Download dataset here this.. The dataset in question is called "The Movies Dataset" and is available on Kaggle. It is a large, comprehensive dataset of metadata for over 45,000 movies, released on or before July 2017. The dataset includes a wide range of information about each movie, including titles, release dates, genres, ratings, production companies, and more.

In addition to the movie metadata, the dataset also includes information on movie credits, such as the cast and crew for each movie, as well as user ratings and reviews scraped from the website MovieLens. Average size of overview is 332 symbols.

Here some rating distribution.

Overall, the dataset provides a rich source of information for movie-related research, including analyses of movie trends, recommendations, and more. It has been used in a variety of machine learning and data science projects, including natural language processing, sentiment analysis, and collaborative filtering for movie recommendations.

9 Experiments

9.1 Metrics

1. F1 score: The F1 score is a measure of the model's accuracy that considers both precision and recall. It is the harmonic mean of precision and recall, and it ranges from 0 to 1, where a higher score indicates better performance. In the context of binary classification, the F1 score is calculated as the weighted average of precision and recall, where precision is the fraction of true positives among the predicted positives, and recall is the fraction of true positives among the actual positives.

2. Accuracy: Accuracy is a measure of the model's correctness that measures the fraction of correct predictions among all predictions. It ranges from 0 to 1, where a higher score indicates better performance. In the context of binary

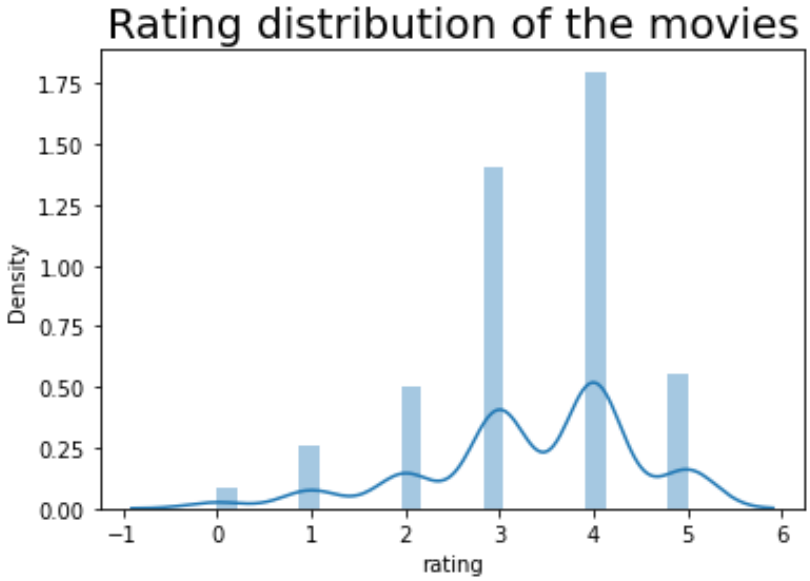

Figure 5: Rating distribution.

classification, accuracy is calculated as the fraction of correct predictions among all predictions, which includes true positives, true negatives, false positives, and false negatives.

3. Mean Squared Error (MSE): The mean squared error is a measure of the model's accuracy that measures the average of the squared differences between the predicted values and the actual values. It is a continuous metric that ranges from 0 to infinity, where a lower score indicates better performance. In the context of regression, MSE is calculated as the average of the squared differences between the predicted values and the actual values.

4. Root Mean Squared Error (RMSE): The root mean squared error is a measure of the model's accuracy that measures the square root of the average of the squared differences between the predicted values and the actual values. It is a continuous metric that ranges from 0 to infinity, where a lower score indicates better performance. In the context of regression, RMSE is calculated as the square root of the average of the squared differences between the predicted values and the actual values.

5. Mean Absolute Error (MAE): The mean absolute error is a measure of the model's accuracy that measures the average of the absolute differences between the predicted values and the actual values. It is a continuous metric that ranges from 0 to infinity, where a lower score indicates better performance. In the context of regression, MAE is calculated as the average of the absolute differences between the predicted values and the actual values.

9.2 Experiment Setup

10 1.TF-IDF

In this work code creates a TF-IDF vectorizer, which is used to transform the training data and test data into a matrix of TF-IDF features. It then trains a logistic regression model on the training data and makes predictions on the test data. Finally, it prints the predicted ratings and actual ratings for each example in the test data.

The "train-test-split" function is used to split the data into a training set and a test set. The "fit-transform" method of the TfidfVectorizer class is used to compute the TF-IDF features for the training set, and the transform method is used to compute the features for the test set.

The logistic regression model is trained using the fit method of the LogisticRegression class, with the TF-IDF features for the training set as input and the corresponding ratings as output. The predict method of the model is used to make predictions on the TF-IDF features for the test set.

The mean-squared-error function from the sklearn.metrics module is used to compute the mean squared error between the predicted ratings and the actual ratings in the test set. The square root of the mean squared error is then computed using the sqrt function from the math module to get the root mean squared error.

The mean-absolute-error function from the sklearn.metrics module is used to compute the mean absolute error between the predicted ratings and the actual ratings in the test set.

The f1-score function from the sklearn.metrics module is used to compute the F1 score between the predicted ratings and the actual ratings in the test set. The 'weighted' option is used to compute the weighted average of the F1 score for each class.

The accuracy-score function from the sklearn.metrics module is used to compute the accuracy of the predictions by comparing the rounded predicted ratings with the actual ratings in the test set.

Finally, the code prints the values of the MSE, RMSE, MAE, F1 score, and accuracy metrics. These metrics are commonly used to evaluate the performance of regression models and classification models.

11 2.Bag-of-words

In this particular situation,code trains a logistic regression model using a bag-of-words representation of the movie descriptions as input features. The Bag of Words (BoW) model is created using the CountVectorizer from the sklearn.feature-extraction.text module, which counts the number of occurrences of each word in the text and creates a sparse matrix where each row represents a document and each column represents a word.

The data is split into training and testing sets using the first 80

The target variable is defined as the movie rating, and the logistic regression model is trained using the fit method of the LogisticRegression class, with the BoW vectors for the training set as input and the corresponding ratings as output. The predict method of the model is used to make predictions on the BoW vectors for the test set.

Finally, the code computes the evaluation metrics for the predictions and prints their values. The metrics are computed in the same way as in the previous example, using the mean-squared-error, sqrt, mean-absolute-error, f1-score, and accuracy-score functions from the sklearn.metrics module. These metrics are used to evaluate the performance of the logistic regression model on the test set.

12 3.CNN

In this case code trains a CNN model on movie overviews to predict their ratings, and then evaluates the model using F1 score, accuracy, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE).

The first part of the code splits the data into training and testing sets, tokenizes the texts and pads the sequences, defines the CNN model, compiles the model with an optimizer and a loss function, and trains the model on the training data.

The second part of the code sets the maximum vocabulary size, maximum sequence length, and embedding dimension for the tokenizer, tokenizes the texts and pads the sequences again, splits the data into train and test sets, builds a CNN model using Keras' functional API, compiles the model with an optimizer and a loss function, and trains the model on the training data. Then, the code evaluates the model using F1 score, accuracy, MSE, RMSE, and MAE.

It's worth noting that the two parts of the code are slightly different in terms of model architecture and the way the data is split and tokenized, but they are both training a CNN model on movie overviews to predict their ratings and evaluating the model using various metrics.

13 4.RNN

And in the final one,code builds and trains a LSTM-based neural network model to predict movie ratings based on movie descriptions. The steps are as follows:

Preprocessing:

1.Tokenize the movie descriptions using a Tokenizer.

2.Pad the sequences so that they are all the same length using pad-sequences.

3.Split the data into training and testing sets.

Build the model:

1.Define the input layer with the appropriate shape.

2.Add an Embedding layer to learn the representation of the words in the input sequences.

3.Add an LSTM layer to capture the temporal dependencies in the input sequences.

4.Add a Dense layer with a relu activation function to learn non-linear relationships in the data.

5.Add an output layer with a single neuron and a linear activation function to output the predicted ratings.

Compile the model:

1.Use mse as the loss function since we are predicting a continuous variable.

2.Use adam as the optimizer to update the weights of the model.

Train the model:

1.Fit the model on the training data.

2.Use the testing data as validation data to prevent overfitting.

Evaluate the model:

1.Use the trained model to predict ratings for the testing data.

2.Calculate various evaluation metrics such as F1 score, accuracy, MSE, RMSE, and MAE.

Predict new ratings:

1.Use the trained model to predict ratings for new movie descriptions.

14 5.BERT

15 Results

1.TF-IDF

The MSE, RMSE, and MAE are all lower in the sBag-of-words, indicating that the model's predictions are closer to the actual values. The F1 score and accuracy are also slightly better in the second example.

2.Bag-of-words

Based on these metrics, the model does not perform very well. The MSE and RMSE are quite high, indicating that the model is not able to accurately predict the ratings. The MAE is also relatively high, meaning that the model's predictions are on average about 1 star away from the actual ratings. The F1 score and accuracy are also quite low, indicating that the model is not performing well in terms of classifying the ratings correctly.

3.CNN

This model has a low F1 score and accuracy, indicating poor performance in predicting the ratings. The mean squared error (MSE) and root mean squared error (RMSE) are high, indicating a large average difference between the predicted and actual ratings. The mean absolute error (MAE) is also high, indicating the model's predictions are far from the actual ratings on average.

4.RNN

The model has an F1 score of 0.2994, which is quite low, indicating that the model is not performing well. The accuracy is also low at 0.3453, which means the model is not able to correctly predict the ratings for most of the movies. The mean squared error is 1.5540, which means that on average, the model's

predictions are off by 1.2466 points. The mean absolute error is 0.9209, which means that on average, the model's predictions are off by 0.9209 points. These metrics suggest that the model needs to be improved to make more accurate predictions.

5.BERT

	tf-idf	bag-of-words	CNN	RNN	BERT
MSE	1.32	1.78	1.51	1.55	12.16
MAE	0.83	0.98	0.94	0.92	0.35
RMSE	1.15	1.34	1.23	1.25	1.65
F1	0.28	0.27	0.22	0.29	
Accuracy	0.38	0.32	0.31	0.35	

Table 1: Statistics of the movie rating prediction.

16 Conclusion

In conclusion, movie rating by its description is an important problem that can be addressed using Natural Language Processing (NLP) techniques. By using machine learning algorithms and sentiment analysis models, it is possible to accurately predict the rating of a movie based on its text description. This can be useful for helping people make informed decisions about which movies to watch and can improve the overall user experience in the movie industry. However,it is really hard to predict rating based only on description,so this remains an ongoing challenge and further research is needed to develop more accurate and efficient methods for predicting movie ratings based on their descriptions.

References

[BANIK, 2018] BANIK, R. (2018). Movie dataset. In *The Movies Dataset*.

[Breitinger, 2015] Breitinger, Corinna; Gipp, B. L. S. (2015). Research-paper recommender systems: a literature survey. *Applied Optics*, 17(4):305–338.

[Devesh Maheshwari and Shrivastava, 2021] Devesh Maheshwari, R. M. and Shrivastava, A. (2021). Bert for text classification: Fine-tuning considerations, pitfalls and opportunities.

[Harris, 1954] Harris, Z. (1954). Distributional structure. *WORD*, 10:146–162.

[Hongliang Yu and Gao, 2021] Hongliang Yu, X. L. and Gao, J. (2021). Investigating the sensitivity of bert in text classification.

[Hyötyniemi, 1996] Hyötyniemi, H. (1996). Turing machines are recurrent neural networks. *Finnish Artificial Intelligence Society*, 1:13–24.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification.

[Michael McTear, 2016] Michael McTear, Zoraida Callejas, D. G. (2016). *The Conversational Interface*, volume 163.

[Miljanovic, 2012] Miljanovic, M. (2012). Comparative analysis of recurrent and finite impulse response neural networks in time series prediction.

[Rajaraman, 2011] Rajaraman, A.; Ullman, J. (2011). Data mining. *Mining of Massive Datasets*, page 1–17.

[Valueva, 2020] Valueva, M.V.; Nagornov, N. L. P. V. G. C. N. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation.*, 177:232–243.