Hybrid Movie Recommendation System

Tao Shan t4shan@uwaterloo.ca
Department of Data Science
University of Waterloo
Waterloo, ON, Canada

Editor:

Abstract

The increasing demand for personalized movie recommendations has motivated researchers to develop more accurate prediction systems that not only enhance user experience but also save costs for companies by targeting their marketing efforts more effectively Ricci et al. (2011) Lu et al. (2015). The primary challenge addressed in this study is to estimate users' movie ratings for unwatched movies based on their historical rating feedback, in order to provide tailored recommendations Koren et al. (2009). Our proposed methodology uses deep learning techniques (BERT and CNN) that combined text and images data in a single network to analyze users' historical ratings, movie profiles, descriptions, and posters for predicting future ratings Devlin et al. (2018) Krizhevsky et al. (2012).

The key contribution of our research is the development of a hybrid prediction system that integrates various data sources to improve the accuracy of movie recommendations. The resulting model demonstrates a promising level of performance, with a Root Mean Square Error (RMSE) of 0.677. This research has the potential to significantly impact the field of movie recommendation systems and contribute to the development of more efficient solutions for both users and companies in the entertainment industry Bobadilla et al. (2013).

Keywords: Recommendation System, Machine Learning, Deep Learning, Natural Language Preprocessing, Convolutional Neural Networks (CNN), Hybrid Prediction System

1 Introduction

In today's world, we are constantly bombarded with an overwhelming amount of content to choose from, including watching movies. With so many movies available, deciding what to watch for clients could be challenging. So that's why we need a movie recommendation system. A recommendation system is a tool designed to assist users in discovering the items they desire from a vast collection of available options. Konstan and Riedl (2012) Specifically, the movie recommendation system provides recommendations based on the user's preference, watching history, previous movie rating, the director and language. For example, Netflix uses algorithms to suggest movies based on more than 65 million members and more than 100 million hours of movies and TV shows daily. Gomez-Uribe and Hunt (2015) A movie recommendation system has the ability to detect regularities and trends in a user's watching behaviour through machine learning algorithms and data analysis techniques and recommend films that they are probable to enjoy.

From the user's point of view, the success of a recommendation depends on several factors, such as system usability, usefulness, interface quality, and user satisfaction. Pu et al. (2011) The analytical model is based on these factors, using historical information to recommend films to users. A successful model enhances the decision-making process and quality, which can help the user find the correct film within a short period of time. Pathak et al. (2010) By providing personalized and relevant suggestions, clients are more likely to remain engaged with the platform and continue using the service.

Our research aims to answer the following question: Can we improve the accuracy of movie rating predictions by incorporating multiple data sources and implementing a combination of BERT Devlin et al. (2018) and Convolutional Neural Networks (CNN). Krizhevsky et al. (2012) To address this question with more up-to-date methods, we will implement and modify the BERT and CNN algorithms as our main methods to process textual and visual data, respectively. In order to evaluate the performance of these algorithms, we will compare them to three baseline methods: XGBoost Chen and Guestrin (2016), Neural Networks Goodfellow et al. (2016), and Random Forest Breiman (2001). By using these baseline methods, we can ensure that our main algorithms perform reasonably, providing more accurate regression for future recommending movies.

2 Related Works

Previous papers discussed recommendation systems' information collection, feedback, prediction methods, and Recommendation filtering techniques. Isinkaye et al. (2015) For filtering methods, there are content-based filtering techniques, collaborative filtering techniques and hybrid filtering techniques. The content-based filtering (CBF) technique uses the attributes of items to generate predictions and recommend items to users. The recommendation is made based on the user's profile using features extracted from the content of the items they have evaluated in the past. CBF could provide analysis even if there is no rating for users, but it requires in-depth knowledge and description of the features of the items in the profile. Isinkaye et al. (2015) Collaborating filtering based on the user's past behaviour, matches users with similar behaviour and recommends similar items. It has promising results according to similar user ratings, The cold start problem arises when dealing with new products or different users. Das et al. (2017) For the hybrid recommendation filtering, it is a combination of the filters above and aims to solve the limitation of both filters. The hybrid model can be designed in several ways, such as by combining the predictions from different models, by using one model to complement the other, or by switching between different models based on the context or user preferences. Ko et al. (2022) Goyani and Chaurasiya (2020)

In recent years, researchers have explored the use of more advanced and computationally expensive techniques, such as deep learning algorithms like Convolutional Neural Networks (CNN) and BERT, to improve recommendation systems. For instance, CNN has been employed in numerous studies to analyze visual content, such as movie posters, to enhance the recommendation process Wang et al. (2019). BERT, a pre-trained deep bidirectional transformer, has been utilized to process textual data, such as movie descriptions, to provide more accurate recommendations Sun et al. (2019).

The primary focus of our study is to investigate the effectiveness of a hybrid system that combines both image and text analysis using CNN and BERT, respectively. By leveraging these modern, sophisticated methods, we aim to develop a more accurate and efficient recommendation system that outperforms traditional content-based and collaborative filtering techniques.

3 Methdology

In this article we present a regression model developed by utilizing datasets from both the MovieLens and TMDB movie APIs. The data generation process is thoroughly documented in the notebook "data_generation_TMDB_large.ipynb", and the MovieLens dataset containing 25 million rating records. The TMDB dataset comprises information on 66,657 movies, including their descriptions and posters.

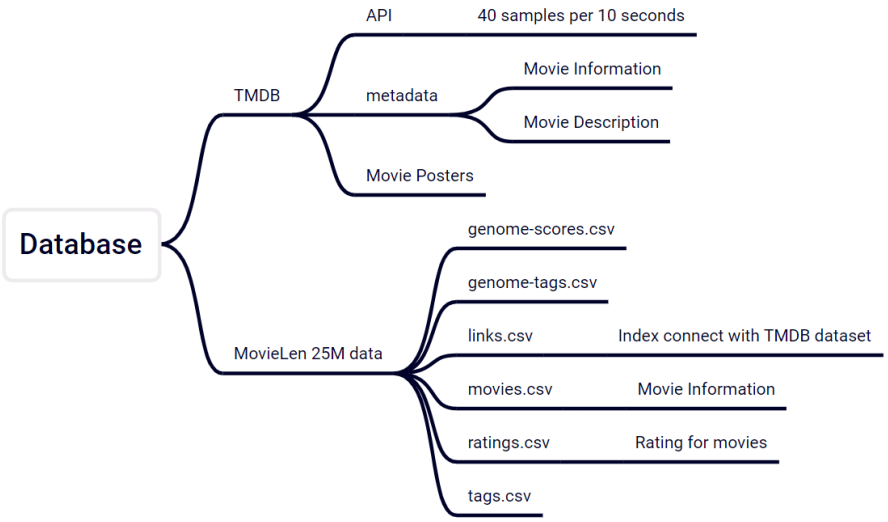

Figure 1: Database Structure

Figure 1 shows the original database system for TMDB and MovieLens dataset. The dataset combines all of these information together, merge accroding to the indexes table between MovieLen and TMDB database.

The MovieLens dataset contains a total of 25 million rating records submitted by 162,541 users. Due to the high computational demands of our hybrid model, which combines BERT and CNN architectures, we opted to work with a smaller sample of the data to evaluate our model's performance. The sample consists of 100 user's rating data including 1,600 rating records, then left join with all movie's data including 66,657 movies' information, descriptions, and posters.

By employing this sample data, we aim to develop a more efficient regression model and potentially use it in the larger dataset, accounting for any potential differences in results. This article will mainly discover the possibility for combine image and text data together in a hybrid deep learning system, including algorithm BERT, CNN, and other comparison

methods including XGBoost and Random Forest. The model is evaluate through RMSE, to test the accurate prediction for movie rating scores.

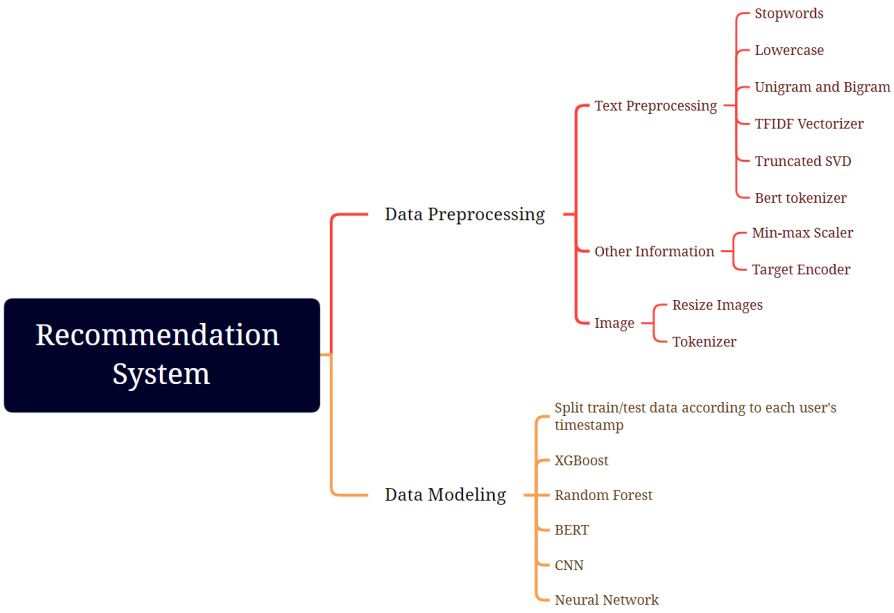

Figure 2: Recommendation System Methods

Figure 2 provides an overview of the methodologies employed in the recommendation system. The preprocessing stage includes text preprocessing, processing other information, and handling images. Text preprocessing typically involves techniques such as tokenization, stemming, and removal of stop words to ensure that the textual data is clean and ready for analysis. Other information, such as numerical or categorical data, may require normalization, one-hot encoding, or other transformation techniques to be effectively utilized by the models. For images, preprocessing usually involves resizing and data augmentation to ensure that the CNN can efficiently process the visual information.

After preprocessing the input data, various models, including the previously discussed BERT, CNN, XGBoost, and Random Forest, are employed to analyze the different types of data. The hybrid model system combines the power of these individual models to create an integrated and versatile recommendation system that can effectively process and analyze both textual and visual information. By leveraging the strengths of each model, we aim to achieve improved performance in generating accurate and relevant recommendations.

3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model developed by Google AI researchers Devlin et al. (2018). It is designed to capture deep contextualized representations by processing text data in both directions using a bidirectional Transformer architecture, enabling it to learn context-dependent em-

beddings. BERT leverages masked language modeling and next sentence prediction as pre-training objectives, which allow it to understand the structure and semantics of sentences more effectively.

The bidirectional nature of BERT allows it to better capture relationships between words and phrases in the input text, thus leading to a deeper understanding of the given context. Once pre-trained on a large corpus of text data, BERT can be fine-tuned for various downstream tasks with smaller labeled datasets, making it a versatile and efficient choice for improving the recommendation system.

3.2 CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily used for image recognition and processing tasks LeCun et al. (1998). They are designed to automatically and adaptively learn spatial hierarchies of features from input images. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which work together to capture local and global patterns in the image data.

In this report, we utilize the VGG16 architecture Simonyan and Zisserman (2014), a 16-layer deep CNN developed by the Visual Geometry Group at the University of Oxford. The VGG16 structure consists of 13 convolutional layers, followed by 3 fully connected layers, and is known for its remarkable performance in various image recognition tasks. The convolutional layers in VGG16 use small filters (3x3) to capture fine-grained details in the images, while the pooling layers help reduce the spatial dimensions, leading to a more efficient and robust feature extraction process. The fully connected layers at the end of the architecture enable the model to learn higher-level representations and make predictions based on the extracted features.

By leveraging the powerful feature extraction capabilities of the VGG16 architecture, our model can effectively process image data and contribute to the overall performance of the recommendation system.

3.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is an optimized implementation of gradient-boosted decision trees Chen and Guestrin (2016). It is designed for both computational efficiency and improved performance, with features such as parallelization, regularization, and early stopping. XGBoost has been widely adopted in the machine learning community and has demonstrated its effectiveness in regression. It has consistently performed well in many machine-learning competitions, including those hosted on Kaggle.

3.4 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and combines their predictions to improve overall model performance Breiman (2001). It mitigates the overfitting issue commonly associated with single decision trees by using a technique called bagging (Bootstrap Aggregating). Random Forest introduces randomness both in the selection of the training samples and the features used for splitting nodes, resulting in a diverse set of decision trees. This ensemble approach reduces the vari-

ance of the model, leading to more accurate and robust predictions. Random Forest has been successfully applied in various domains, including image classification, regression, and feature selection.

3.5 Hybrid Model System

The hybrid model system combines the power of neural networks, BERT, and CNN architectures to create a robust and versatile model for various applications. By leveraging the strengths of each individual model, we aim to achieve improved performance in tasks that require a combination of text and image understanding.

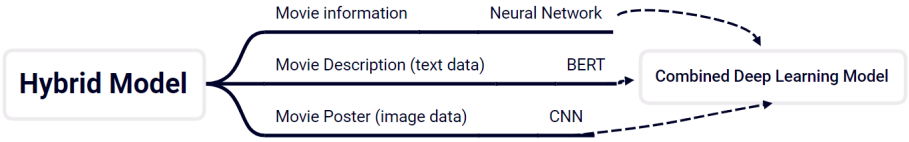

Figure 3: Hybrid Model System Structure

Figure 3 provides a more detailed view of the hybrid model's architecture, showcasing the specific layers and connections between the BERT, CNN, and neural network components. The fusion of these powerful architectures within the hybrid model system enables it to effectively process and analyze both textual and visual information, leading to more accurate predictions and insights.

4 Result

In this section, we present the performance of our algorithms on the task of predicting the target variable. Specifically, we evaluated the performance of four models: BERT, XGBoost, Random Forest, and a hybrid model that combines BERT and CNN. The evaluation is by RMSE for predicting the user's rating score. For each user, extract the last timestamp sample as testing data. We aim to discover the performance of models based on previous users' rating history.

4.1 Algorithm Design

The BERT model was implemented using the transformers library in Python. We fine-tuned the pre-trained BERT model and hybrid method on our dataset with loss function, optimizer, learning rate and number of epochs.

Figure 4 shows the detailed deep learning model structure that combined Neural Networks, BERT and CNN (VGG16). Which is a complex structure that combined text data and image data into a single network. The hybrid model was implemented using Keras with TensorFlow backend. We used a pre-trained VGG16 model for the convolutional part of the model, and fine-tuned the pre-trained BERT model on our dataset. The pre-trained model is based on "bert-base-uncased" in TensorFlow, which uses less computational cost "bert-large-uncased". In addition to the hyper-parameters tuned for BERT, we also optimized the number of neurons in the dense layers and the dropout rate for the hybrid model.

Layer (type)	Output Shape	Param #	Connected to
image_input (InputLayer)	[(None, 224, 224, 3)]	0	[]
text_input (InputLayer)	[(None, 128)]	0	[]
vgg16 (Functional)	(None, 7, 7, 512)	14714688	['image_input[0][0]']
tf_bert_model_1 (TFBertModel)	TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 128, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)	109482240	['text_input[0][0]']
flatten_1 (Flatten)	(None, 25088)	0	['vgg16[0][0]']
dropout_78 (Dropout)	(None, 768)	0	['tf_bert_model_1[0][1]']
dropout_80 (Dropout)	(None, 25088)	0	['flatten_1[0][0]']
dense_6 (Dense)	(None, 64)	49216	['dropout_78[0][0]']
dense_8 (Dense)	(None, 256)	6422784	['dropout_80[0][0]']
movie_info_input (InputLayer)	[(None, 18)]	0	[]
dropout_79 (Dropout)	(None, 64)	0	['dense_6[0][0]']
dropout_81 (Dropout)	(None, 256)	0	['dense_8[0][0]']
user_info_input (InputLayer)	[(None, 3)]	0	[]
dense_5 (Dense)	(None, 4)	76	['movie_info_input[0][0]']
dense_7 (Dense)	(None, 4)	260	['dropout_79[0][0]']
dense_9 (Dense)	(None, 4)	1028	['dropout_81[0][0]']
concatenate_1 (Concatenate)	(None, 15)	0	['user_info_input[0][0]', 'dense_5[0][0]', 'dense_7[0][0]', 'dense_9[0][0]']
output (Dense)	(None, 1)	16	['concatenate_1[0][0]']

Total params: 130,670,308
Trainable params: 130,670,308
Non-trainable params: 0

Figure 4: Hybrid Model Structure

For the XGBoost model, we tuned the learning rate, number of estimators, and maximum depth parameters using the Scikit-learn library in Python. For the Random Forest model, we tuned the number of estimators, maximum depth, and minimum samples split parameters using the same library.

We encountered several technical challenges during the implementation of our system. One of the main challenges was processing the text data using BERT, as the model requires a fixed-length input sequence. We overcame this challenge by padding and truncating the input sequences to a maximum length of 128 tokens. Another challenge was the hybrid

system has a huge number of parameters that is hard to tuning the model. We overcame this challenge by rerun the notebook multiple times and choose the best result.

4.2 Performance Analysis

In the performance analysis of our models, we evaluated their performance on the test set using the root mean squared error (RMSE) as the evaluation metric. The RMSE measures the average difference between the predicted and actual values of the target variable, with a lower value indicating better performance.

Model	RMSE	Execution Time (minutes)
BERT (before tuning)	0.726	>=30
BERT	0.671	>=30
XGBoost (before tuning parameter)	0.763	1
XGBoost (after tuning parameter)	0.680	1
Random Forest (before tuning parameter)	0.694	1
Random Forest (after tuning parameter)	0.670	1
Hybrid Model (BERT & CNN before tuning parameter)	0.701	>=120
Hybrid Model (BERT & CNN after tuning parameter)	0.677	>=120

Table 1: Performance of each model

Table 1 shows the root mean squared error (RMSE) of each model on the test set. We can see that the BERT model achieved the lowest RMSE of 0.671, followed closely by XGBoost with an RMSE of 0.680. The Random Forest model achieved an RMSE of 0.670, and the hybrid model achieved an RMSE of 0.677. The XGBoost, Random Forest and BERT model uses text, movie information and rating datas, and Hybrid method additionally add the images.

Model	**Best Parameters**
BERT	Number of epochs: 21, learning rate: 0.01, batch size: 32
XGBoost	Learning rate: 0.05, number of estimators: 100, maximum depth: 3
Random Forest	max depth: 10, minimum samples split: 10, number of estimators: 200
Hybrid (BERT & CNN)	Number of epochs: 43, learning rate, batch size: 32, dropout rate: 0.1

Table 2: Summary of hyperparameters used in each model.

The number of epochs for BERT and the Hybrid (BERT & CNN) models were selected through early stopping with a root mean squared error (RMSE) loss function and a tolerance of 5 epochs. The learning rate and batch size were directly modified for these models due to their high computational cost.

For XGBoost and Random Forest, the values for hyperparameters were selected using a grid search with cross-validation (CV) to identify the best combination of parameters. The goal was to select the hyperparameters that resulted in the best performance on the validation set while avoiding overfitting.

5 Discussion

5.1 Discuss the result

The result shows the models after tuning parameter has similar great performances. Random Forest relatively outperformed than other models in this case. The solution shows adding image data into the model does not necessarily improve the model performance.

Also by comparing the model execution time, BERT and Hybrid system needs large computational cost including RAM, GPU and hours of time. But XGBoost and Random Forest needs only minutes to finish the implementation. The training execution is based on Colab Permium+ with GPU A100, 83.48GB RAM and 40GB GPU RAM. The solution for this article use a relatively small sample is because lack of RAM when execute hybrid system, which shows the hybrid system is hard to implement compare with other models. Baltrušaitis et al. (2018) BERT and the hybrid system demand more advanced knowledge and computational resources, as they involve complex architectures and handling multimodal data.

Random Forest emerged as a better choice for regression with movie and text data due to several reasons. The model is designed to handle both categorical and continuous features with ease, making it suitable for the mixed data types found in movie and text data Breiman (2001) Also, Random Forest is an ensemble learning method that utilizes multiple decision trees, making it more robust to overfitting compared to single decision tree modelsLiaw and Wiener (2002) Random Forest can effectively capture non-linear relationships and complex interactions between features, which might be present in movie and text dataCutler et al. (2007) Lastly, Random Forest demonstrates relatively lower computational costs and faster execution times compared to BERT and the hybrid system, making it more efficient for practical implementations.Probst et al. (2019)

BERT is a powerful pre-trained language model that has shown great success in various natural language processing tasks, such as sentiment analysis, question-answering, and named entity recognition Devlin et al. (2018) Its ability to capture complex semantic and syntactic relationships in textual data and its capacity to handle large amounts of training data make it an ideal choice for regression tasks involving text.

XGBoost, on the other hand, is a scalable tree-boosting system that has achieved state-of-the-art results in numerous machine-learning challengesChen and Guestrin (2016). It is particularly effective in dealing with large and complex datasets, owing to its sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. The gradient boosting mechanism of XGBoost allows it to iteratively learn from previous models' errors, enabling it to have a great performance.

The result from hybrid model that combines CNN and BERT, also combines image, text and other supporting data, shows the possibility for combining these models as a single prediction system. The fusion of different data modalities and models enables the leveraging of complementary information, thereby enhancing the overall predictive power of the system. Ngiam et al. (2011); Baltrušaitis et al. (2018)

One possible problem of the hybrid model is because of the conflict of a number of epochs between CNN and BERT structures. In Multi-label Text Classification Yarullin and Serdyukov (2021), it was found that the sequence-generating BERT model achieved decent results in significantly fewer training epochs compared to the standard BERT. This shows a

relevant low number of epochs will increase the model performance. For CNN the researcher trained the model between 20-30 epochs Poudyal et al. (2022), which is a relatively large number. This shows both of these models require a different number of epochs to converge.

5.2 Limitations

This analysis selected only a small sample for user ratings data, and all samples for movie data. The number of samples selected depended on the availability of GPU RAM for the hybrid model method to compare all models with the same amount of data because of the high consumption of GPU RAM for the hybrid model. If the GPU RAM is potentially large, the analysis could use more data and change to Bert's pre-trained model "bert-large-uncased" to compare the performance. It is possible for a larger data size to make the hybrid model more outstanding.

Another limitation is the dataset does not have user profile information, which is also an essential factor for predicting which user might give a high rating to the movie. The current dataset knows how the user previously rated other movies. Still, if there is information such as the user's subscription, and demographic data, it will be better for the analysis.

6 Conclusion

In this study, we explored various recommendation systems by performing regression on user rating scores based on rating, image, text, and movie information. Our results demonstrate that the model performance are similar, and Random Forest model relatively outperforms other models, including XGBoost, BERT and the hybrid system combining CNN and BERT. Although the hybrid system did not yield a significant improvement in performance, it presents the potential of creating a unified recommendation system that leverages both text and image data.

While the best-performing technique in our analysis was Random Forest, there is still room for improvement, and no single technique can be considered as definitively solving the problem. Future research should explore other state-of-the-art models, including alternative deep learning architectures and more extensive datasets, to enhance the recommendation system's performance. However, as these models may require significant computational resources, access to better RAM and GPU setups will be crucial for efficient training.

Additionally, incorporating user profiles into the recommendation system can further personalize the suggestions and improve overall accuracy. Since user profiles are not available in the TMDB and MovieLens databases, finding a suitable database that provides this information is a priority for future work. By expanding the range of models, datasets, and user information, future research can further refine recommendation systems and enhance their performance in predicting user rating scores.

References

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2018.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Antonio Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

D. Richard Cutler, Thomas C. Edwards, Karen H. Beard, Adele Cutler, Karl T. Hess, James Gibson, and Joshua J. Lawler. Random forests for classification in ecology. *Ecology*, 88 (11):2783–2792, 2007.

Debashis Das, Laxman Sahoo, and Sujoy Datta. A survey on recommendation system. *International Journal of Computer Applications*, 160:6–10, 2017. doi: 10.5120/ijca2017913081.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2018.

Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2015. doi: 10.1145/2843948.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Mahesh M. Goyani and Neha Chaurasiya. A review of movie recommendation system: Limitations, survey and challenges. *Electronic Letters on Computer Vision and Image Analysis*, 19:18–37, 2020.

F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015. ISSN 1110-8665. doi: 10.1016/j.eij.2015.06.005.

Hyunwoo Ko, Seungsoo Lee, Yunsu Park, and Aelee Choi. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, 11(1): 141, 2022. doi: https://doi.org/10.3390/electronics11010141.

Joseph A. Konstan and John Riedl. Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, 2012. doi: 10.1007/s11257-011-9112-x. URL `https://doi.org/10.1007/s11257-011-9112-x`.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. URL `https://cogns.northwestern.edu/cbmg/LiawAndWiener2002.pdf`.

Jie Lu, Daren Wu, Miao Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

Bhuvan Pathak, Robin Garfinkel, Ram Gopal, Rajkumar Venkatesan, and Feng Yin. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, 27(2):159–188, 2010.

Sujan Poudyal, Mahnas J. Mohammadi-Aragh, and John E. Ball. Prediction of student academic performance using a hybrid 2d cnn model. *Electronics*, 11(7), 2022. ISSN 2079-9292. doi: 10.3390/electronics11071005. URL `https://www.mdpi.com/2079-9292/11/7/1005`.

Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019.

Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender Systems (RecSys'11)*, pages 57–64. ACM, 2011.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, Boston, MA, 2011.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

Chenyang Sun, Xipeng Qiu, and Xuanjing Huang. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*, 2019.

Xuan Wang, Lei Yu, and Jun Wang. A deep learning-based movie recommender system using movie posters. In *Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things*, pages 68–73, 2019.

Ramil Yarullin and Pavel Serdyukov. Bert for sequence-to-sequence multi-label text classification. In Wil M. P. van der Aalst, Vladimir Batagelj, Dmitry I. Ignatov, Michael Khachay, Olessia Koltsova, Andrey Kutuzov, Sergei O. Kuznetsov, Irina A. Lomazova, Natalia Loukachevitch, Amedeo Napoli, Alexander Panchenko, Panos M. Pardalos, Marcello Pelillo, Andrey V. Savchenko, and Elena Tutubalina, editors, *Analysis of Images, Social Networks and Texts*, pages 187–198, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72610-2.