Kaggle Username: crispyfriedchicken

Introduction

This assignment aims to predict user ratings in an Amazon website recommendation system. The primary goal is to predict the ratings users will give to products accurately. This task presents three main challenges:

1. Cold Start Problem
   This issue arises when the system cannot generate accurate recommendations for new users who have rated only a small number of items (Guo,2012). The lack of sufficient data makes it difficult to understand the preferences of these users.
2. Data Sparsity
   This problem occurs when the system struggles to find enough reliable similar users because active users typically rate only a small portion of available items (Guo,2012) . This could be due to users just joining the platform or rarely providing ratings.
3. Large Dataset
   Handling a large training dataset efficiently is another challenge. The system must be designed to process and analyse vast amounts of data without compromising performance.

Based on the three challenges identified in this assignment, I used content-based filtering to address the cold start problem. In content-based recommender systems, data items are grouped into different item profiles based on their descriptions. Content-based techniques can effectively tackle the cold start problem when new items have sufficient descriptions (Roy & Dutta, 2022). By using content-based filtering, I was able to overcome the cold start problem by leveraging the detailed descriptions of these new products. This method relies on the similarity of items based on their features (product name) to generate a prediction.

To handle the large dataset problem, I used Collaborative Filtering because collaborative filtering could handle large datasets in an efficient way. Based on these two problems, I decided to use **Hybrid Based**, which combines the advantages of those two approaches (Casalegno, 2022).

For content-based filtering, I used Word2Vec embeddings and cosine similarity. Word2Vec is employed as the content-based filtering model to capture the semantic similarity between products based on their names, ensuring that the recommendations are meaningful and contextually relevant. After that, I use cosine similarity to find the most similar item to that new item. The average rating of similar items is then used to estimate the rating for the new item. For the collaborative filtering component, I used three different models: SVD, Collaborative Neural Network, and KNN. Based on my experiment in collaborative filtering, I chose SVD. Therefore, the hybrid models I implemented are Hybrid SVD.

The next section will discuss the exploratory data analysis of the train and test sets and explain the rationale behind choosing a hybrid model. The methodology will focus on the architecture of each model. The results and analysis section will compare the performance of each model, detailing the evaluation process. Finally, the conclusion will summarise the findings to determine the best model for addressing the stated challenges.

Explanatory Data Analysis

I plotted the rating distribution in the training dataset to understand the dataset. This distribution shows a significant skew towards higher ratings, with a notable concentration around 4.0 and 5.0. This skewness indicates a prevalence of high ratings among users in the training dataset, which could lead

to model bias. In such cases, prediction models may overly focus on high ratings at the expense of accuracy for lower ratings.

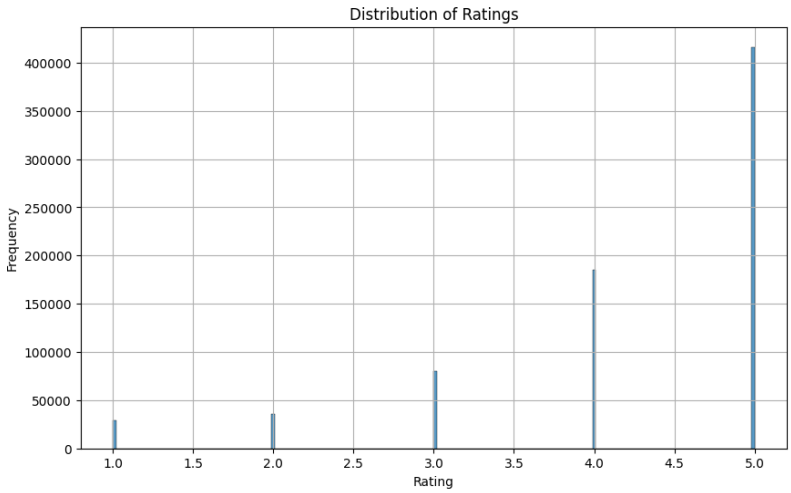


Figure 1 Distribution of Rating in Train Dataset

Next, I will visualise the distribution of new versus existing products in the test set. By identifying that 18.3% of products in the test set are new and were not featured in the training dataset, I have chosen to implement content-based filtering to address the cold start problem. Content-based filtering is particularly suited for this scenario because it relies on the characteristics and attributes of items themselves rather than on user-item interactions. This method enables the recommendation system to generate relevant ratings for new products by relying solely on their features. Following the approach outlined by Joseph and Ms. Jetty Benjamin (2022), I will implement content-based filtering using cosine similarity to determine the similarity between each product. This strategy helps tackle the challenge posed by new products in the dataset, ensuring effective recommendations are made by comparing content similarities among items.

To handle the given large dataset, I chose Collaborative Filtering due to its scalability and independence from individual user data. This approach is particularly suitable for the existing product names, which constitute 83% of the test set. To leverage the benefits of both content-based and collaborative filtering, I decided to implement a hybrid model. This hybrid model combines the strengths of collaborative filtering for scalability and accurate recommendations with the ability of content-based filtering to handle new or less frequently interacted products, thus enhancing the overall recommendation system's performance and adaptability.

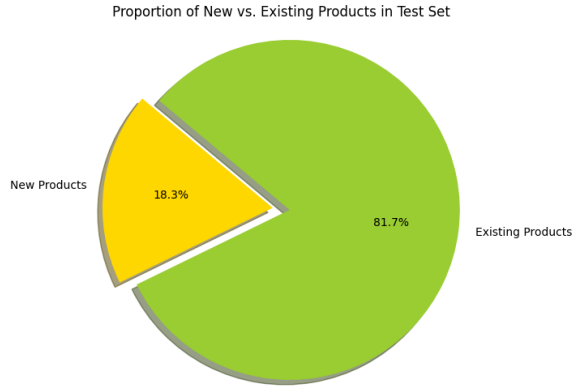


Figure 2 Proportion of New and Existing Products in Test Set

Methodology

Content Based Filtering

I developed a content-based filtering model to address the challenge of recommending new products. The model utilises word embeddings to capture semantic relationships within product descriptions, particularly focusing on product names.

For each new product:

If both the product and the user are new, I apply cosine similarity to compare the new product with all existing products in our dataset. I identify similar products and use their ratings as a basis to estimate a rating for the new product.

If the product is new but the user has a history of ratings, I combine the user's rating history (user profile) with the cosine similarity of the new product to existing products. This approach customises the predicted rating based on the user's unique preferences, ensuring that new users don't receive a generic rating based on product similarity alone.

I use Cosine similarity because it could measure the similarity between product embeddings, which helps in identifying products with similar characteristics. These similarities are then used to infer potential ratings for new products (Gomes, 2023).

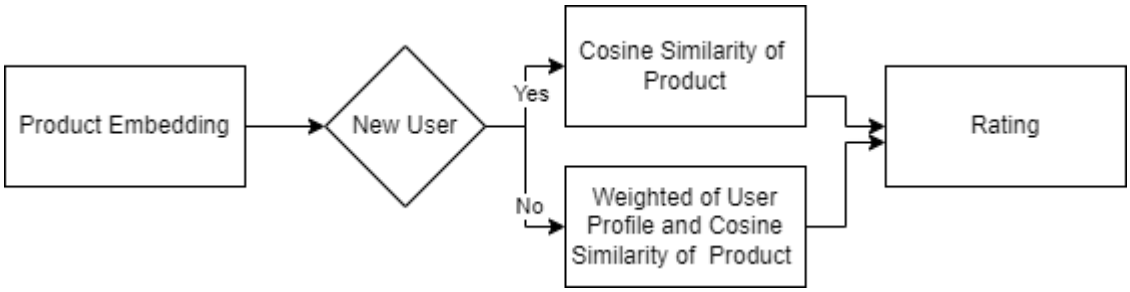


Figure 3 Content Based Filtering

To evaluate my approach, I tested word embeddings: BERT, Word2Vec, and FastText

Table 1 Experiment for Content Based Filtering on Validation Dataset

Model Name	Setup	RMSE	MAE	Precision (P@K)	Recall (R@K)
BERT		1.9178	1.4010	0.8	0.0001

Word2Vec	Vector Size:300,Window:5	1.5695	1.2727	0.9	0.00011
FastText	Learning Rate-0.1,Epochs:25	6.3235	0.91	0.5	5.193

Among the three models, BERT exhibits the highest RMSE and MAE scores, indicating the largest average errors in predictions. Despite this, BERT achieves a respectable recall and precision score, yet its recall score is significantly low. On the other hand, Word2Vec shows the best performance with the lowest RMSE and MAE values, and it also surpasses BERT in precision. FastText, while presenting moderate RMSE scores, has the lowest MAE among the three models. However, its precision is lower than that of Word2Vec. In terms of recall, FastText performs better than both BERT and Word2Vec, indicating a relatively better ability to identify all relevant items. These results are based on the characteristics of the feature I used, which is the product name, to calculate the similarity of products. BERT is designed to understand the context of each word within a sentence by considering the words around it (Rani Horev, 2018). It might be somewhat overkill for short texts like product names, where less contextual information is available. Word2Vec learns to represent text in a numerical format by predicting a word from its neighbouring words or predicting neighbouring words from a word (Dutta, 2021). This model generally performs well with larger datasets where more examples of word contexts can be learned, making it suitable for this task. For short texts like product names, Word2Vec has been shown to perform very effectively, achieving the highest recall and precision. FastText considers not just words but also sub-word segments, which allows it to better handle rare words or typos (Durna, 2024). However, despite its capability to understand substrings within words, its overall performance still lags behind Word2Vec in terms of precision and recall, though it does well in terms of MAE. Based on these justifications, I chose Word2Vec for my content-based filtering.

Collaborative Based Filtering

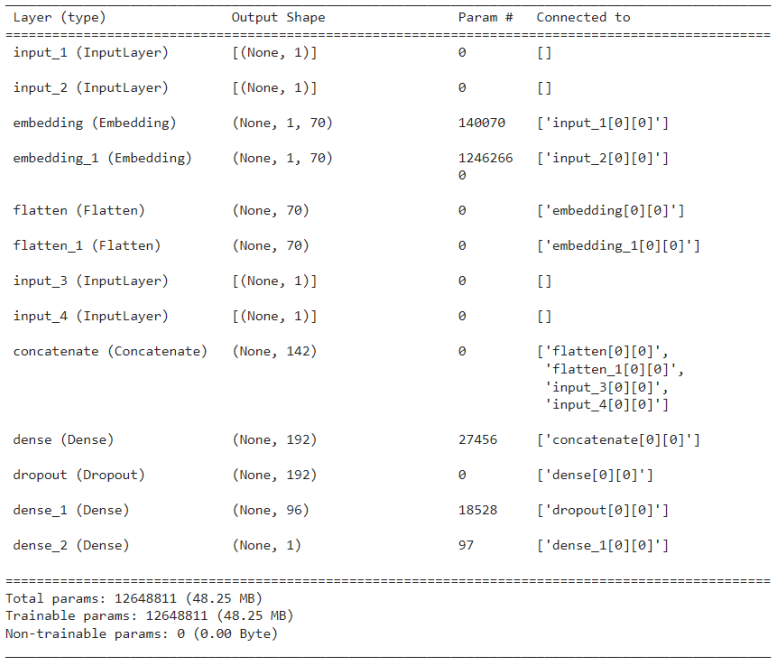
I chose collaborative filtering because it predicts a user's preferences based on past behaviour and similarities with other users. Since I have training data that stores past behaviours, I can use it as a reference to predict other users' ratings. In this section, I evaluated KNN, SVD, and Neural Network models.

Singular Value Decomposition (SVD) is a mathematical technique used to decompose a matrix (A) into three key components: two orthogonal matrices and one diagonal matrix. In this task, (A) represents the user-item interaction matrix, (P) the user matrix, (∑) weights the significance of latent factors, and (Q^T) is the transpose of the item matrix (Zhao, 2024). SVD helps to reduce the dimensionality of the training data, enhancing feature extraction and noise reduction. Because of that, SVD is suitable for large datasets (this assignment). However, it relies on patterns of interactions among users and items to suggest personalised recommendations (Banerjee, 2024). To solve that problem, I tried to tune the system to find the best interaction level that enables accurate rating predictions. After establishing this, I also fine-tuned each parameter based on the chosen approach.

k-Nearest Neighbors (kNN) algorithm predicts user preferences by identifying similar users based on common ratings and averaging the ratings of the top-k closest users (Sharma et al., 2023). I chose kNN because it is simple to understand and implement. However, it can be computationally expensive in the prediction phase and requires substantial memory, as it retains the training data rather than learning from it.

Neural Collaborative Filtering (NCF) is an approach that enhances traditional matrix factorisation by integrating a neural network to capture both linear and non-linear relationships. NCF's architecture includes two main components: Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). The GMF part focuses on linear relationships through direct multiplication of user and item embeddings. The MLP part uses deep neural networks with ReLU

activations to process concatenated embeddings, targeting non-linear patterns. These two branches merge at a layer called NeuMF, which combines their outputs before the final sigmoid activation produces the predicted rating or interaction probability (Gupta,2023). Figure 4 below shows the architecture of NCF that I used in this task.



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 1)]	0	[]
input_2 (InputLayer)	[(None, 1)]	0	[]
embedding (Embedding)	(None, 1, 70)	140070	['input_1[0][0]']
embedding_1 (Embedding)	(None, 1, 70)	1246266 0	['input_2[0][0]']
flatten (Flatten)	(None, 70)	0	['embedding[0][0]']
flatten_1 (Flatten)	(None, 70)	0	['embedding_1[0][0]']
input_3 (InputLayer)	[(None, 1)]	0	[]
input_4 (InputLayer)	[(None, 1)]	0	[]
concatenate (Concatenate)	(None, 142)	0	['flatten[0][0]', 'flatten_1[0][0]', 'input_3[0][0]', 'input_4[0][0]']
dense (Dense)	(None, 192)	27456	['concatenate[0][0]']
dropout (Dropout)	(None, 192)	0	['dense[0][0]']
dense_1 (Dense)	(None, 96)	18528	['dropout[0][0]']
dense_2 (Dense)	(None, 1)	97	['dense_1[0][0]']

Total params: 12648811 (48.25 MB)
Trainable params: 12648811 (48.25 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 4 NCF Model Summary

This model includes multiple input layers for user and item IDs, as well as additional feature inputs such as votes and helpful votes. These inputs are transformed into dense embeddings, which are then flattened and concatenated to form a comprehensive feature set. This set then passes through a sequence of dense layers, including a dropout layer, to prevent overfitting. Finally, the architecture culminates in a final dense layer that produces a single value.

Table 2 Experiment for Collaborative-Based Filtering on Validation Dataset

Model Name	Setup	RMSE	MAE
KNN	k:10 ,min_k:3	0.8863	0.6179
SVD	n_factors:150, n_epochs:40, lr_all:0.01, reg_all:0.02	0.8144	0.5465
Neural Network	Epochs:5, learning_rate: 0.001 Embedding_dim: 70	0.9811	0.7399

From Table 2, we can see the KNN model shows moderate accuracy, indicated by its RMSE and MAE scores. However, the training time for the KNN model is relatively long compared to other approaches. This extended duration is due to the inherent characteristics of KNN, which involves computing distances between points in high-dimensional space, making it computationally intensive, especially as the size of the dataset increases.

The SVD model exhibits superior performance in minimizing prediction errors, as evidenced by its lower RMSE and MAE values. Conversely, the Neural Network model registers the highest scores in both RMSE and MAE. This might be attributed to the model's limitations in handling unseen products, where it defaults to average product or user ratings without further refinement, leading to less personalized predictions. Additionally, there is significant potential for enhancing the Neural Network's accuracy through further hyperparameter tuning, which could improve its ability to learn more complex patterns and interactions within the data.

I also plotted the performance analysis of each approach in data training and validation. The SVD model showed a clear improvement and stabilisation in training RMSE over epochs. It means SVD has effective learning capabilities. In contrast, the KNN model exhibits high RMSE values for both training and validation. This means the KNN model has less effective learning and predictive capabilities compared to the SVD model. Although both models achieve a similar validation RMSE of about 0.8, the SVD model reaches this level with a significantly lower training RMSE. It means SVD could do generalisation and can capture underlying patterns without overfitting.

This performance difference is also likely due to the hyperparameter tuning and the iterative training approach used in the SVD model, which allows it to learn progressively better over additional epochs. In contrast, the KNN model may be constrained by its algorithmic structure and the chosen hyperparameter settings, which could explain the persistently high RMSE that does not improve with further training iterations.

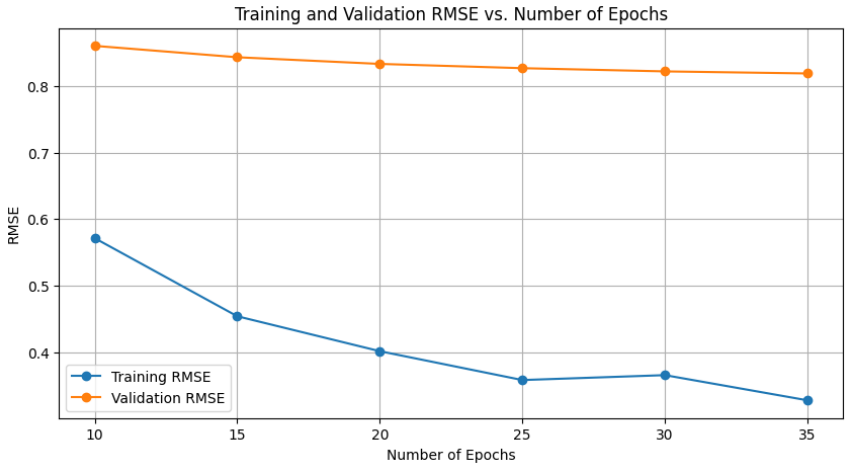


Figure 5 Learning Rate SVD

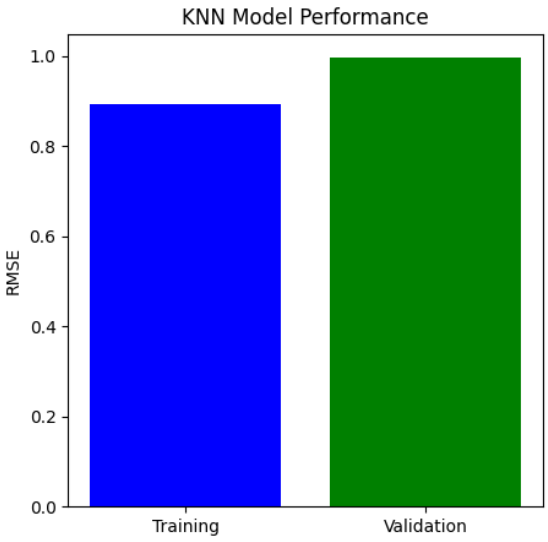


Figure 6 Comparison of KNN

Based on the results in Table 2, I have decided to combine Word2Vec with SVD. This decision is motivated by SVD's performance compared to others.

Hybrid Pre-processing

To effectively merge content-based and collaborative filtering, I begin by computing product embeddings using a trained Word2Vec model. Each product's embedding is calculated by averaging the embeddings of the words in the product's name, resulting in a single vector that represents the product. To enhance the embeddings with more contextual information, I concatenate additional product-specific metadata, such as helpful_votes and votes, to this vector.

Furthermore, I compute user profiles by aggregating the embeddings of products they have rated. For each user, a weighted average of these product embeddings is calculated, where the weights are the user's ratings. This produces a unique embedding for each user that encapsulates their preferences based on their past interactions.

Hybrid Based Filtering

For this hybrid approach, I use a Switching Hybrid Approach. This strategy introduces an additional layer to the recommendation model, which selects the appropriate model to use based on the data's characteristics (Chiang, 2021). The reason for using a switching approach is that it allows me to implement content-based filtering for new products and collaborative filtering for existing products. A new product refers to a product that hasn't been rated by any users in the training data, as well as existing products that have minimal interactions with users. The reason behind this approach is:

1. New products often suffer from the "cold start" problem in collaborative filtering due to a lack of historical interaction data. By switching to content-based filtering for these products, the system can still make meaningful recommendations based on the products' features or descriptions.
2. For products with sufficient user interactions, collaborative filtering is more effective as it leverages user behaviour. This method excels when there is enough data to understand and predict patterns based on user similarities and interactions. Because of that, I try to find the minimal interactions for the collaborative filtering.

By using this switching approach, the system can dynamically adjust its recommendation strategy, ensuring that all products, regardless of their interaction history.

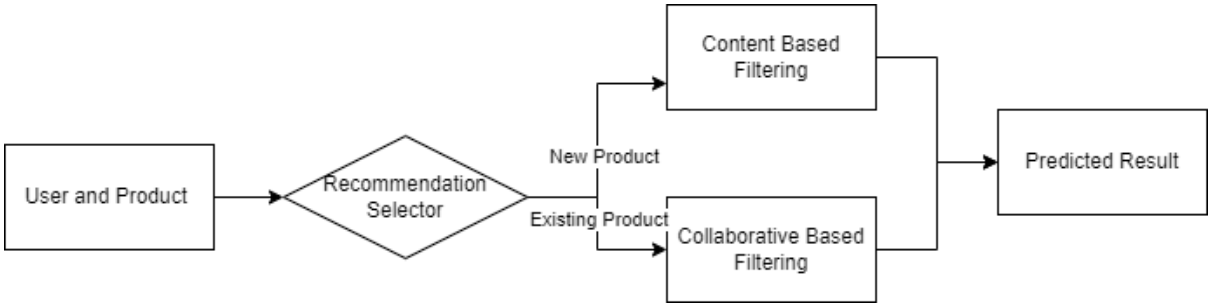


Figure 7 Switching Hybrid

For the implementation of Hybrid, I use parameters that I already tuned up in the previous section.

Evaluation Phase

For the evaluation phase, I evaluate my approaches in the validation dataset using RMSE (Root Mean Square Error), MAE (Mean Absolute Error), Precision@K, and Recall@K. These metrics were chosen because:

- RMSE and MAE: Both metrics measure the average magnitude of the errors in the predictions, so it will show the overall prediction accuracy.
- Precision@K and Recall@K : Precision@K measures the proportion of recommended items in the top-K set that are relevant, while Recall@K measures the proportion of relevant items that are captured in the top-K recommendations. This assesses the coverage of the recommendations across all relevant items. These metrics are particularly useful for my content-based filtering, where I need to find similarities between products. Evaluating these metrics allows me to assess the effectiveness and relevance of the similarities identified by the model.

Result & Analysis

Table 3 Experiment of Hybrid Based Filtering in Validation

Model Name	Setup	RMSE	MAE	RMSE Score in Kaggle
Word2Vec+SVD	Word2Vec(Vector Size :300, Window:5) SVD (n_factors:150, n_epochs:40, lr_all:0.01, reg_all:0.02)	0.46068	0.24	0.86767

The hybrid Word2Vec+SVD approach demonstrated the best performance among the models evaluated, likely due to Word2Vec's capability to capture semantic relationships between items combined with SVD's robustness in matrix factorisation.

When we compare this approach with my previous experiments, which utilised standalone content-based and collaborative filtering approaches, I observed a significant reduction in both RMSE and MAE for the hybrid algorithms. This improvement is largely attributed to the enhanced capabilities of the hybrid models, which effectively integrate the strengths of both content-based and collaborative filtering techniques.

Conclusion

Based on this experiment, the hybrid approach combining Word2Vec and SVD demonstrated superior performance compared to other hybrid methods. Additionally, hybrid methods, in general, outperformed standalone content-based and collaborative filtering techniques. This suggests that integrating multiple recommendation strategies can enhance prediction accuracy and overall system performance.

Reference

Banerjee, A. (2024, March 19). *Pros and Cons of Collaborative Filtering*. Medium. https://medium.com/@ashmi_banerjee/pros-and-cons-of-collaborative-filtering-9c3aa4ce44f6

Casalegno, F. (2022, December 12). *Recommender Systems — A Complete Guide to Machine Learning Models*. Medium. https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748

Chiang, J. (2021, June 30). *7 Types of Hybrid Recommendation System*. Analytics Vidhya. https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8

Durna, M. B. (2024, January 15). *Advanced Word Embeddings: Word2Vec, GloVe, and FastText*. Medium. https://medium.com/@mervebdurna/advanced-word-embeddings-word2vec-glove-and-fasttext-26e546ffedbd#:~:text=FastText%20is%20an%20advanced%20word

Dutta, M. (2021, July 13). *Word2Vec For Word Embeddings -A Beginner's Guide*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/

Guo, G. (2012). Resolving data sparsity and cold start in recommender systems. In J. Masthoff, B. Mobasher, M. C. Desmarais, & R. Nkambou (Eds.), *User modeling, adaptation, and personalization* (Vol. 7379, pp. 361-370). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-31454-4_36

Gomes, N. D. (2023, January 31). *The cosine similarity and its use in recommendation systems*. Medium. https://naomy-gomes.medium.com/the-cosine-similarity-and-its-use-in-recommendation-systems-cb2ebd811ce1

Gupta, M. (2023, July 28). *Recommendation Systems using Neural Collaborative Filtering (NCF) explained with codes*. Data Science in Your Pocket. https://medium.com/data-science-in-your-pocket/recommendation-systems-using-neural-collaborative-filtering-ncf-explained-with-codes-21a97e48a2f7

Joseph, A., & Ms. Jetty Benjamin. (2022). Movie Recommendation System Using Content-Based Filtering And Cosine Similarity. *Zenodo (CERN European Organization for Nuclear Research)*. https://doi.org/10.5281/zenodo.6791117

Rani Horev. (2018, November 10). *BERT Explained: State of the art language model for NLP*. Medium; Towards Data Science. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00592-5

Sharma, A., Nirmal, J., Rana, D., & Setia, S. (2023). *A Review On Collaborative Filtering Using Knn Algorithm*. https://doi.org/10.1109/otcon56053.2023.10113985

Zhao, T. (2024). Performance comparison and analysis of SVD and ALS in recommendation system. *Applied and Computational Engineering*, *49*(1), 142–148. https://doi.org/10.54254/2755-2721/49/20241080