Movie Recommendation System: A Hybrid Approach

A Project Report
Submitted in partial fulfillment of
the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

by

Bandaru Bhavyanath	B180399EC
Chevuru Sai Jaswanth	B180668EC
Eruguri Jeshwanth	B180556EC
Jarapala Mohith Pamar	B181027EC

Under the guidance of

Dr. Vinay Joseph

Dr. Nujoom Sageer Karat

Department of Electronics and Communication Engineering
National Institute of Technology Calicut
Calicut, Kerala, India – 673 601

2021-2022

Acknowledgments

We take this opportunity to express our deepest gratitude to everyone who supported us through this online semester, and helped and motivated us in completing this project.

We sincerely express our whole hearted gratitude to our project guides **Dr. Vinay Joseph** and **Dr. Nujoom Sageer Karat**, for helping and guiding us through this project Without whom, we would not have been able to complete the project successfully.

We would like to thank the Project Co-ordinator **Dr. V. Sakthivel** and the evaluation committee for all the valuable suggestions put forward during the project evaluations.

We would like to acknowledge the Department of Electronics and Communication Engineering, National Institute of Technology, Calicut, for extending its full support during the entirety of the project and helping us through the online and offline semester.

We would also like to thank our beloved family members who supported us and have provided a working environment at our homes, in these testing times.

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this project report are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This project report is our own work and does not contain any outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

B Bhavyanath (B180399EC)

C Sai Jaswanth (B180668EC)

E Jeshwanth (B180556EC)

J Mohith Pamar (B181027EC)

NIT Calicut
Date: 02/05/2022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Certificate

This is to certify that the project report entitled **Movie Recommendation System: A Hybrid Approach** submitted by **B Bhavyanath (B180399EC), CH Sai Jaswanth (B180668EC), E Jeshwanth (B180556EC), J Mohith Pamar (B181027EC)** to National Institute of Technology Calicut for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering, is a bonafide record of the project work carried out by them under my supervision and guidance. The content of the project report, in full or parts have not been submitted to any other institute or university for the award of any degree or diploma.

Dr. Vinay Joseph
(Project Guide)
Dept. of ECE, NITC.

Dr. Deepthi P.P.
(Professor and Head of Department)
Dept. of Electronics and Communication Engineering
NIT Calicut

Dr. Nujoom Sageer Karat
(Project Guide)
Dept of ECE, NITC.

Date: 02/05/2022

(Office seal)

Abstract

Natural language processing is crucial in today's progressing world because it helps overcome linguistic ambiguity like speech recognition and text analytics. Natural language processing has made great progress recently using the BERT framework. Recommendation System is another significant area that is very popular and useful for faster-automated decisions. Surveys show we spend more than 100 days of our lives choosing what to watch. A Movie Recommendation System suggests a collection of movies to the users depending on their preferences and similarities. To improve the quality of recommendations, we have developed a hybrid approach by combining collaborative filtering using Matrix Factorization and content based filtering using BERT4Rec which is an extension of the BERT framework for recommendation systems. Several studies indicate that the hybrid approach can provide more accurate recommendations. We have utilized the MovieLens dataset, which is a representative and popular real-world benchmark dataset in recommendation systems. We have evaluated the performance of our novel movie recommendation system

Contents

List of Abbreviations

NLP	Natural Language Processing
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
MLM	Masked Language Modelling
NN	Neural Netowork
ANN	Artificial Neural Networks
NSP	Next Sentence Prediction
BERT	Bi-directional Encoder Representations From Transformers
MC	Markov Chains
MDP	Markov Decision Process
SASREC	Self-Attentive Sequential Recommendation

List of Figures

Chapter 1

Introduction

In today's technologically progressing world, where the internet has become a crucial part of human life, users often face the matter of an excessive amount of choice. Right from trying to find a motel to finding good investment options, there's an excessive amount of information available. To assist the users in dealing with this information explosion, companies have deployed recommendation systems to guide their users. The research within the area of recommendation systems has been happening for several decades now and is exponentially developing due to the abundance of practical applications. And therefore, it is a problem-rich domain. Various such recommendation systems are being designed to complement the user experience in renowned disciplines.

Recommender Systems have helped the economy and development of significant e-commerce websites like Amazon, Netflix, etc. Amazon commerce applications provide targeted recommendations based on various factors like age, gender, etc. Approximately 35% of their sales are made because of good suggestions. Netflix made recommendation systems a salient feature of their domain. On average, 2/3rd of the movie recommendations given by their sophisticated recommendation systems are preferred and watched by their users.

Similarly, Google uses recommender systems and makes up for a massive portion of its revenue by targeting ads to users based on their profiles, which is possible because of recommendation systems. Google News generates approximately 38% more click-throughs because of its relevant recommendations.

Recommender Systems generate recommendations, and it is up to the user to accept them according to their choice. And they may also provide, immediately or at a subsequent stage, implicit or explicit feedback. The actions of the users and their feedback may be stored in a database and can be used for providing new and

better recommendations in the subsequent user-system interactions. The economic potential of these recommender systems has led some of the biggest e-commerce websites and the online movie streaming service Netflix to make these systems a crucial part of their websites. High-quality personalized recommendations add another dimension to the user experience. The web personalized recommendation systems are recently applied to provide different types of customized information to their respective users. These systems can be used for various kinds of applications and are very common nowadays.

1.1 Motivation

Users can use recommendation systems to search and select items from a large amount of information available on the internet and other electronic information sources. They supply the user with a limited collection of items well suited to the description, given a broad set of items and a description of the user's desires. Intelligent assistants for selecting and choosing websites, news headlines, TV listings, and other information have been developed recently in recommendation systems. Users of such systems frequently have a variety of competing priorities. Personal preferences, socioeconomic and educational backgrounds, and private and professional interests are prevalent. As a result, tailored advanced technologies that process, filter, and display accessible data in a way that suits each unique user become appealing.

Before the rise of the e-commerce industry, various products were sold exclusively in physical stores. The store's physical space limited the inventory, and products that didn't sell well were unprofitable. Merchants were often drawn to the idea that selling only the most popular mainstream products with a fixed inventory would make a profit. In the mid-1990s, the introduction of the online marketplace completely revolutionized the retail industry. Nowadays, online retailers have more business than physical stores because more varieties of products are available. Recommender systems play a vital role in narrowing down the prospect options for a specific user based on his wants.

1.2 Challenges

User feedback powers recommender systems. We can provide more relevant recommendations customized to users' tastes as we accumulate information about what they appreciate or oppose.

Individuals need to provide personal information (experience hyper-personalization) to the recommendation system for more beneficial services. But it causes issues of data privacy and security. Many users feel hesitant to feed their data into recommendation systems that suffer from data privacy issues. The recommendation system is bound to have users' personal information and use it to the fullest to provide personalized recommendation services. The recommendation systems' development and design must surely entitle trust among their users to provide better functionality.

The scalability of algorithms with real-world datasets under the recommendation system is one of the most significant issues. A large amount of changing data is generated by user-item interactions in the form of ratings and reviews, and hence scalability is a considerable worry for these datasets. Because recommendation systems inefficiently interpret findings from massive datasets, several advanced large-scale methods are required to address this problem.

Chapter 2

Literature Survey and Background

2.1 References

D.K. Yadav et al. proposed MOVREC, a collaborative filtering-based movie recommendation system. The information given by the user is used in collaborative filtering. That data is processed, and a movie is recommended to users, organized in order of highest to lowest rating. The method also allows the user to choose the qualities he wants the movie to be recommended.[3]

Luis M Capos et al. focused on traditional recommender systems: content-based and collaborative filtering. He presented a novel technique combining Bayesian networks with collaborative filtering because both have disadvantages. The suggested system is tailored to the task at hand and includes probability distributions that may be used to create helpful conclusions.[4]

Harpreet Kaur et al. have described a hybrid system. Content, as well as a collaborative filtering algorithm, are used in the system. While making recommendations, the context of the films is taken into account. In the suggestion, both the user-to-user and user-to-item relationships are essential[5]

Using the movie lens dataset, S. Agrawal and P. Jainimplemented a hybrid approach (combining content-based filtering and collaborative filtering) using the most popular technique, SVM, as a classifier. The hybrid approach shows a significant increase in the accuracy of the recommendation engine compared to traditional approaches.[6]

Costin-Gabriel Chiru et al. presented Movie Recommender, a system that provides movie suggestions based on information about the user. This approach tries to overcome the problem of unique suggestions, which arises when data specific to the user is ignored. The user's psychological profile, viewing history, and data concerning movie scores from other websites are all gathered. They are based on an estimate of aggregate similarity. The system is a hybrid approach that employs content-based and collaborative filtering techniques.[7]

2.2 Recommender Systems

Recommender systems are information search and filtering tools that assist consumers in finding relevant things and making better decisions while looking for movies, books, vacations, or electrical devices. A recommender system's primary purpose is to avoid information overload by providing tailored suggestions that can aid users in their decision-making process. Figure 1 depicts a simplified version of the general mechanism that drives recommendation systems. However, due to the growing need for high-quality personalization and suggestion, these systems frequently encounter restrictions and problems.

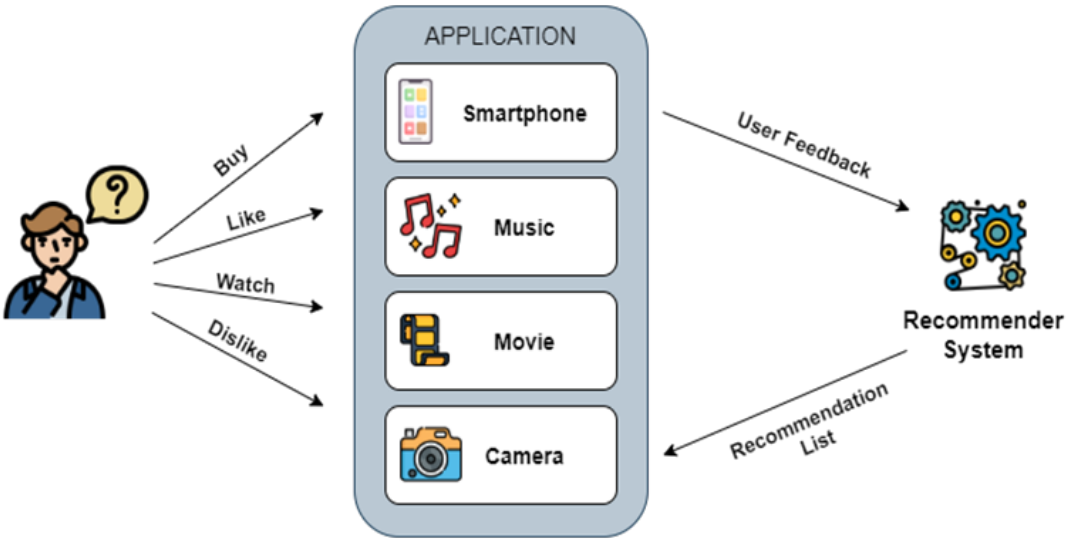

Figure 2.1: Illustration of the logical process of a recommendation system

Regardless of the application domain, recommendation systems have become a vital asset. Nowadays, Internet users rely on the WorldWideWeb for assistance in planning and selecting a variety of daily activities, such as what music to listen to

(Spotify), what consumer items to buy (Amazon, AliExpress), and what movies to watch (Netflix), among other things. Furthermore, social media platforms such as LinkedIn, YouTube, and Facebook have integrated recommendation engines that suggest groups to join, individuals to follow, videos to watch, and postings to like. Although the foundations of recommender systems may be traced back to substantial work in cognitive science, approximation theory, information retrieval, and forecasting theories, as well as links to management science and consumer choice modeling in marketing, recommender systems developed as a separate study topic in the mid-1990s when scientists began focusing on recommendation issues that explicitly rely on the rating structure. The recommendation challenge may be simplified to the difficulty of calculating ratings for goods that the user has not seen in its most basic form. This estimate is often based on the user's ratings of other things as well as some other information. It will be able to recommend the products with the highest estimated ratings to the user once it is possible to estimate the ratings for objects that have not yet been rated .
As part of intelligent systems, Recommender systems make use of a variety of types of knowledge. The knowledge needed to make recommendations comes from four places: the user, other system users, data about the goods being suggested, and finally, the recommendation field itself, which includes knowledge about how recommended items are used and what requirements they meet.

2.3 Types of Recommender Systems

Content-based Recommendations, Collaborative Recommendations, and Hybrid Recommendations are the three basic kinds of recommendation systems . In the first method, the user will be recommended things comparable to those that he or she has previously appreciated. In the collaborative method, on the other hand, suggestions are created based on things consumed by users with comparable interests and preferences to the recommended user. Finally, merging content-based and collaborative suggestions results in hybrid techniques commonly used since both types of recommendations may be useful .

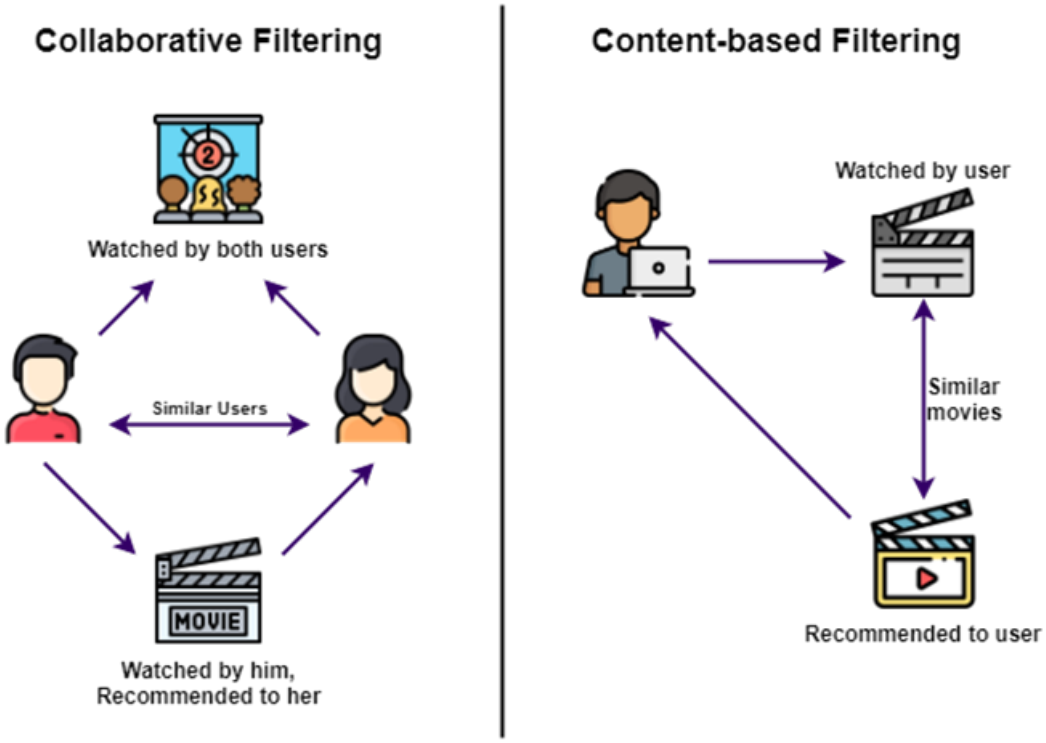

Figure 2.2: Collaborative and Content Based Filtering

2.3.1 Collaborative Filtering

The technique of filtering or assessing objects utilizing the views of others is known as collaborative filtering. While the phrase collaborative filtering (CF) has only been around for a little over a decade, it is based on exchanging thoughts that people have done for hundreds of years. Based on known user ratings of products, collaborative filtering (CF) predicts user preferences in item selection. CF successfully overcame the information overload problem as one of the most popular techniques for recommender systems. Memory-based and model-based CF are the two primary branches of CF. The majority of current research focuses on enhancing the similarity metrics to increase the accuracy of memory-based algorithms. However, only a few studies have focused on prediction score models, which we feel are more essential than similarity metrics.

2.3.2 Content Based Filtering

Content-based recommendation algorithms analyze item descriptions to identify products of particular interest to the user. Because the tiny print of different advising systems differed, the representation of objects was supported. Content-based recommendation systems strive to suggest products comparable to those that a user has previously enjoyed. Indeed, a content-based recommender's core method compares the properties of a user profile, which stores preferences and interests, with the attributes of a content object (item) to suggest new exciting things to the user. The content-based Recommendation method creates a User-based Profile by searching for users' likes and dislikes. The total of the item profiles makes up the user profile, with the combination indicating the customer or user's ratings. The recommender system will recommend new goods or anything that has not yet been reviewed; however, the recommender system will not recommend things that are not in the same category as the items for which the user has submitted ratings. Content-based filtering algorithms generally fail when the user profile or location descriptions are not sufficiently constructed.

2.3.3 Hybrid Methods

Single recommendation systems are combined as sub-components in hybrid recommendation systems. This hybrid technique was created to address the issue with traditional recommendation systems. This method combines two or more suggestion systems in various ways to maximize their benefits. Collaborative filtering is usually integrated with another approach in a weighted way in hybrid systems.As a result, a hybrid approach will provide users with better recommendations based on ratings and interests. Several well-known public databases have been frequently utilized to recommend movies and other forms of entertainment material. Experiments were carried out using public datasets such as MovieLens.

The MovieLens dataset is subjected to content-based and collaborative filtering in this proposed methodology. Content-based filtering generates suggestions by matching keywords and characteristics related to database items to a user profile. The user profile is built using information gathered from a user's behaviors, such as purchases, ratings (likes and dislikes), downloads, movie searches on a website, and movie link clicks. The engine grows increasingly accurate as the user offers more inputs or acts on the recommendations. Collaborative filtering is founded on the notion that individuals prefer things that are similar to other things they like and things that other people with similar interests like. It is not required.Anything else than a user's past preference for a set of options items. Because the assumption here is based on previous facts. Those users who have agreed in the past are

more likely to agree in the foreseeable future. This model accepts the user-id as an input and processes it. Information utilizing collaborative filtering and content-based filtering. The hybrid model will be based on the user's profile suggest the top ten films.

Chapter 3

Dataset and Software

3.1 Dataset

The Movielens-1m dataset is used in this project. This is a standard dataset that was maintained and collected by Group Lens. This dataset contains approximately 1,000,209 ratings given by the 6,040 anonymous users against 3,952 movies. It is the largest Movie-lens dataset that contains demographic data. The minimum number of movies rated by each user is close to 20. This dataset contains six attributes: movie id, movie title, movie genres, user id, user ratings, and timestamp. The ratings given by each user lie in the range of zero to five. The timestamp is denoted in seconds. The dataset contains various genres labeled such as Action, Adventure, Horror, and Romance. In addition to that, the dataset contains demographic information like gender, age, and occupation of the Users. We have adopted ratings, timestamps, movie IDs, and movie names in the training of our models, Matrix Factorization and BERT4REC.

3.2 Libraries Used

3.2.1 Pytorch

Pytorch is an open-source low-level API developed by Facebook for Natural Language Processing and Computer vision. It can be thought of as a more powerful version of Numpy. Pytorch aids in computing Tensors with the help of Graphical Processing Units. Pytorch contains modules like Autograd, optim, and nn modules, which help in defining various computational graphs, gradients, optimization algorithms, and Neural networks. We have adopted Pytorch for adopting embeddings, transformer encoder, and Adam optimizer modules.

3.2.2 Pytorch Lightning

Pytorch is an open-source low-level API developed by Facebook for Natural Language Processing and Computer vision. It can be thought of as a more powerful version of Numpy. Pytorch aids in computing Tensors with the help of Graphical Processing Units. Pytorch contains modules like Autograd, optim, and nn modules, which help in defining various computational graphs, gradients, optimization algorithms, and Neural networks. We have adopted Pytorch for adopting embeddings, transformer encoder, and Adam optimizer modules.

3.2.3 Numpy

NumPy is a Python package for the processing of arrays. It includes a high-performance multidimensional array object as well as utilities for manipulating them. It is the 13 most crucial Python package for scientific computing. It has a number of features, including the following:

- An N-dimensional array object with a lot of power.
- Advanced (broadcasting) capabilities • C/C++ and Fortran code integration tools
- Linear algebra, the Fourier transform, and random number skills are all useful.

3.3 Scikit-Learn

Scikit-learn is an open-source machine learning library accessible for the Python Programming Language. It contains different clustering, regression, classification, and dimensionality reduction algorithms like support vector machines, decision trees, random forests, K-means, Principal Component Analysis, and gradient boosting. Scikit-learn can work in conjunction with libraries Numpy and Scipy. We have used Scikit-learn for the train-validation-test methodology in implementing Matrix-Factorization.

3.4 Computational Platform

Google Colab is a open source python notebook that runs on google cloud. It allows us to write and use python code without any requirement of configuration and execute through the browser, well suited for implementation of machine learning algorithms. It makes computation easier and much faster. Colab supports both TPU, GPU version runtime which makes computations ever faster.

Chapter 4

Matrix Factorization

4.1 Introduction

Collaborative filtering (CF) recommender systems create recommendations based on user-item relationships, with no further knowledge about the users or objects required. Neighborhood-based CF and latent factor models are two typical ways of constructing CF recommender systems.

One of the approaches employed in collaborative model-based filtering is Matrix Factorization[1] , which is a well-known algorithm in the recommendation system field. It is a model that performs well in forecasting ratings, outperforming the Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) models. However, it turns out that the model is fundamentally linear, making it unable to capture significant nonlinear and nuanced interactions that might be used to forecast user preferences.

Neighborhood-based approaches calculate coincidence between items or users based on their user-item relationships to produce predictions. The user-based method looks at what different users with comparable interactions have done for their predictions. While, the item-based technique assesses item-to-item similarity and makes predictions based on the user's previous ratings.

The user-item rating matrix is used in latent factor models for movie recommendations to profile both users and objects using several latent variables. For each person and object, the number of factors might range from 10 to 100. A projected rating for an item may be calculated by multiplying the factor vectors for the user and the object together. These elements might be considered inferred properties of an object or user by a computer. A feature, often referred to as latent factor,

in the context of movies can be a timestamp, target age range, a genre, or even something utterly unintelligible. Matrix factorization is a practical latent factor approach for recommender systems, and it is the subject of this study.

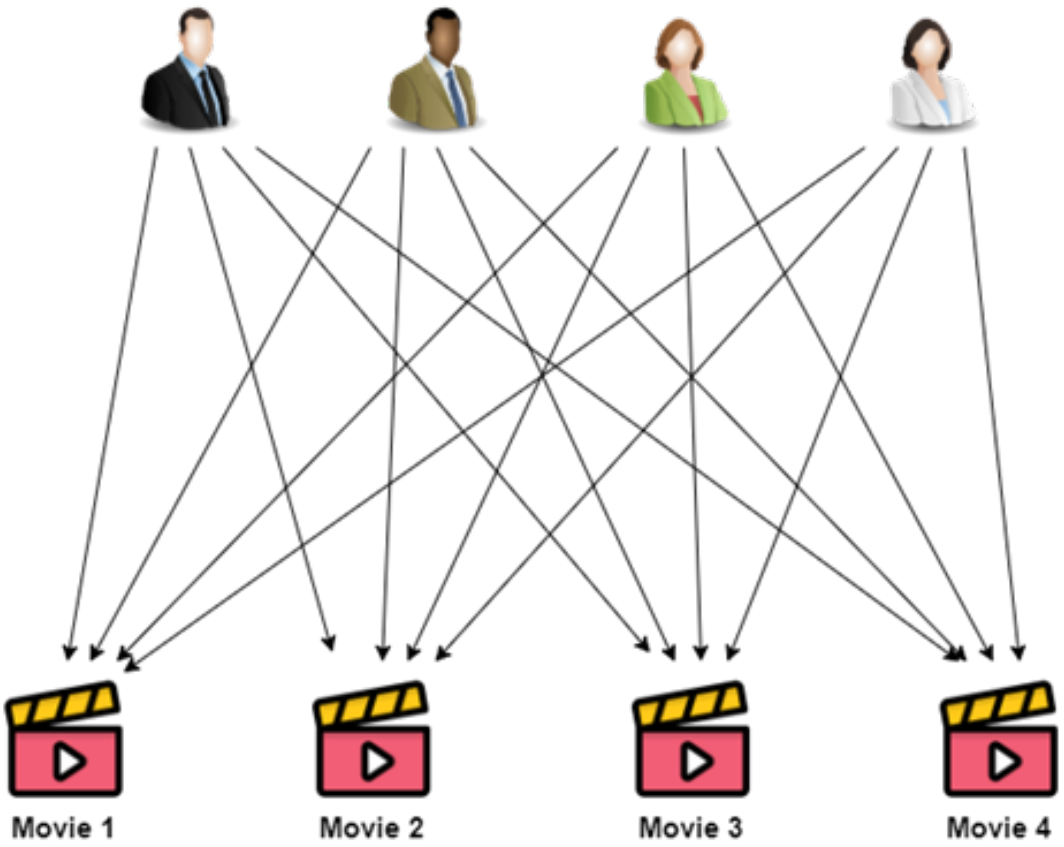

Figure 4.1: User-Movie Relation

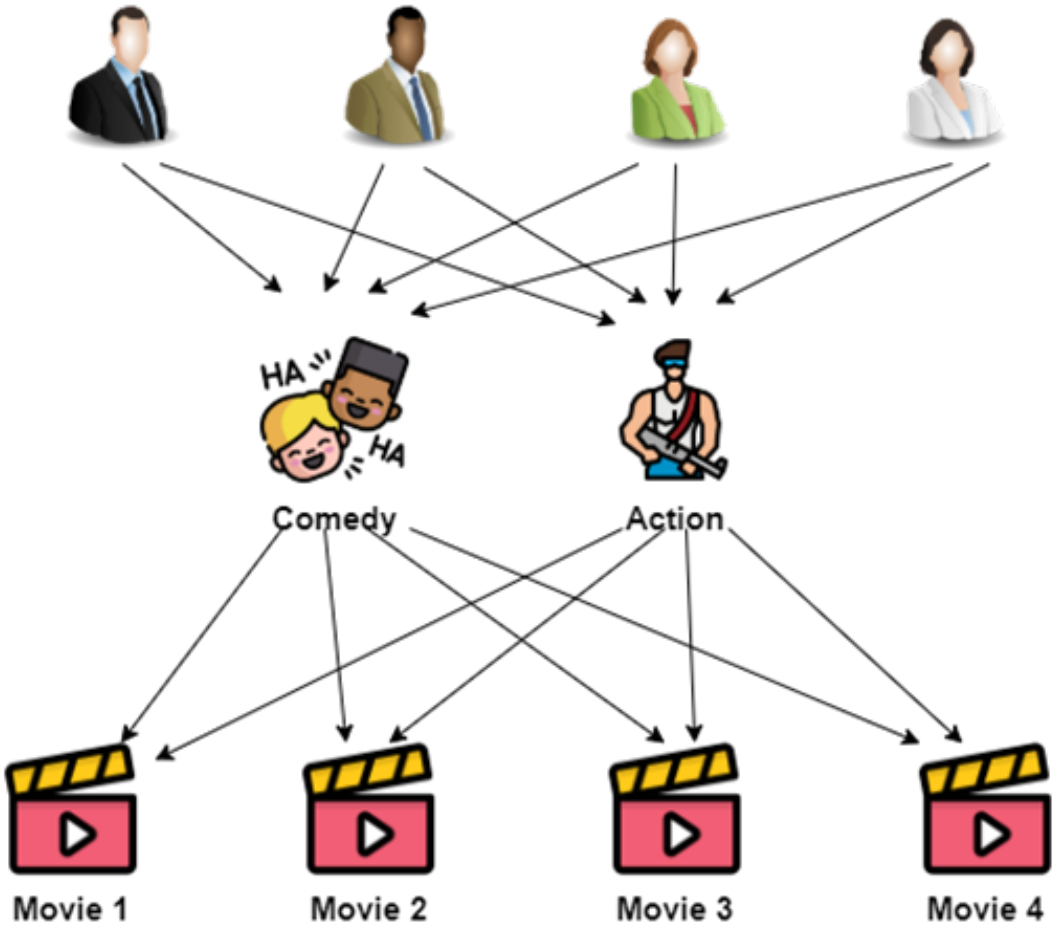

Figure 4.2: User-Movie Relation with latent Factors

Matrix factorization (MF)[1] is a method for calculating a system's latent component model based on user-item interaction. Users on one axis and things on the other, these user-item interactions can be represented as a matrix. Interactions in many of the movie recommendation systems are generally the user ratings data present for movies, although other data, such as temporal effects, explicit or implicit feedback, and confidence levels, can also be used. This matrix of ratings is sparse in real-scenario applications, since users give ratings to only a small percentage of movies when compared to all movies present. Even on relatively sparse matrices, MF can make robust predictions.

Figure 4.3: Matrix Factorization

4.2 Model Training

4.2.1 Matrix Factorization without bias

We decrease the dimensions of the rating matrix r by using the Matrix factorization approach, which is essentially representing the bigger rating matrix as a product of two latent component matrices u and v, for users and movies, respectively.

$$\begin{bmatrix} r_{11} & \dots & r_{1i} \\ \vdots & \ddots & \vdots \\ r_{n1} & \dots & r_{ni} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} m_1 & m_2 & \dots & m_i \end{bmatrix}$$

$$r = um^T \tag{4.1}$$

f is the extracted number of features, n is the total number of users and i is the number of items/movies.
Each row u_n is a feature vector for a user n and each row m_i is a feature vector for an item i. The product of these vectors produces an estimate of the original rating matrix.

$$r_{ni} = u_n m_i^T \tag{4.2}$$

4.2.2 Matrix Factorization with bias

To improve the predictions, we have used a bias for movie i called b_i and a bias for user u called b_u, and average of the global rating μ to model the rating r_{ui}.

$$r_{ni} = \mu + b_i + b_n + u_n m_i^T \tag{4.3}$$

The rating is divided down into components, with movie and user bias indicating how far the movie and user differ from the global average. So the things which are common in both movie and user(the collaborative part) will be $u_n m_i^T$.

$$\min \sum_{n,i} \left(r_{ni} - b_i - b_n - u_n m_i^T\right)^2 + \beta \left(\|u_n\|^2 + \|m_i\|^2 + b_i^2 + b_n^2\right) \tag{4.4}$$

In addition to learning the bias independently, We can solve this equation using a stochastic gradient descent approach.

$$b_i \leftarrow b_i + \alpha \left(e_{ui} - \beta b_i\right) \tag{4.5}$$

$$b_n \leftarrow b_n + \alpha \left(e_{ni} - \beta b_n\right) \tag{4.6}$$

$$m_i \leftarrow m_i + \alpha \left(e_{ni} u_n - \beta m_i\right) \tag{4.7}$$

$$u_n \leftarrow u_n + \alpha \left(e_{ni} m_i - \beta u_n\right) \tag{4.8}$$

4.2.3 Algorithm

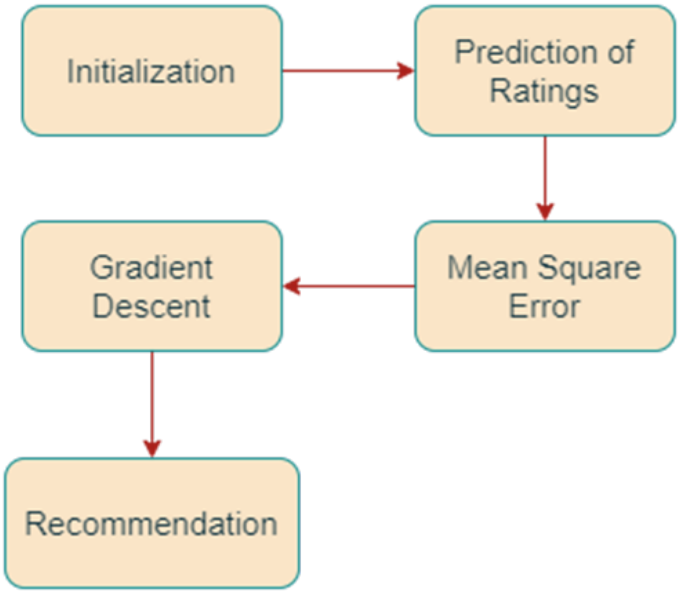

Figure 4.4: Block Diagram

- Initialize matrix u of size (users x K) and matrix m of size (Movies x K) with random values using a uniform distribution in [0,0.05]. Where, K is the number of features which are going to be extracted.

- Calculate predicted ratings by multiplying user and item matrices.
- Compute mean square error with predicted ratings and original ratings by the user.
- Using gradient descent update the weights of user matrix and item matrix.

The gradient descent algorithm was implemented in the following steps:

- Iterate over all the initially present ratings in training dataset.
- Calculate a predicted rating
- Calculate the error for predicted rating to that of initial rating using mean square error.
- Update the User and Item matrices according to error with the help of Adam Optimizer.
- After training, predict the ratings by taking dot product of weights of a user in user matrix with every movie in item matrix and recommend the top ratings.
- Recommended the top movies by calculating user learned ratings by taking dot product every movie in the movie matrix.

4.3 Evaluation

The most recent Movie-Lens dataset was used, which included 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 Movie-Lens users. The data for the ratings was split as 70:10:20 between training, validation and testing. The training and validation losses vs epochs graph is plotted. Refer Figure 4.5

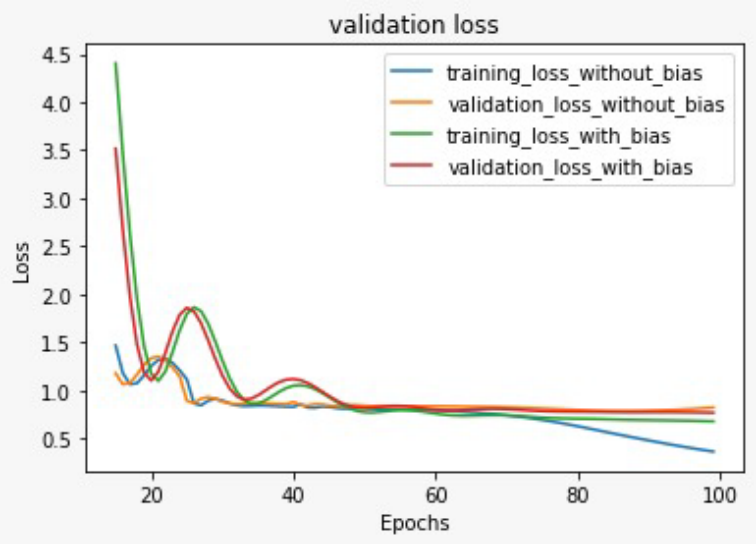

Figure 4.5: Training and Validation Loss Plots

The Metric used for Evaluation is the Mean Square Error.

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.9}$$

The method used for the evaluation is that we have used the 20 % test dataset as the ground truth and compared them with the ratings predicted by the Matrix Factorization model. Also we have used Cosine Similarity as a sanity Check for the predictions predicted by the model by calculating the cosine similarity score between the recent most highest rated movie by the user in the dataset and the predictions.

$$\text{cosine similarity } = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{4.10}$$

Where,
A - Weight vector of recent highest rated movie by a user
B - Weight vector of the movie recommended to a user

4.4 Results

```
[50] print(mean_squared_error(model,test))

     0.7568132776446861
```

Figure 4.6: Mean Square Error

	userId	movieId	rating	timestamp	title
499942	270	374	5	972669952	Mr. Smith Goes to Washington (1939)
209370	270	634	5	972669914	Monty Python and the Holy Grail (1974)
80454	270	578	5	972669864	Mister Roberts (1955)
735445	270	427	5	972669786	Midnight Run (1988)
859527	270	151	5	972669786	Midnight Cowboy (1969)
...	...	...	...	...	...
925695	270	2498	2	968624228	Braddock: Missing in Action III (1988)
775416	270	1497	2	968624162	Bonfire of the Vanities (1990)
853193	270	3031	2	968623891	Beyond the Poseidon Adventure (1979)
540273	270	1452	2	968623809	Beneath the Planet of the Apes (1970)
659116	270	259	2	968623519	Avengers, The (1998)

379 rows × 5 columns

Most recent high rated movie by the user : Mr. Smith Goes to Washington (1939) 5

Figure 4.7: User Watch History

```
Godfather, The (1972) --> 0.84962887
Raiders of the Lost Ark (1981) --> 0.85307413
Casablanca (1942) --> 0.92780215
Rear Window (1954) --> 0.917664
Citizen Kane (1941) --> 0.86343676
Sanjuro (1962) --> 0.8830031
Wrong Trousers, The (1993) --> 0.9251889
One Flew Over the Cuckoo's Nest (1975) --> 0.9056849
Star Wars: Episode IV - A New Hope (1977) --> 0.8322398
To Kill a Mockingbird (1962) --> 0.9567722
```

Figure 4.8: Recommendations given by our MF model

Chapter 5

BERT4REC

5.1 Introduction

Content-based recommendation algorithms analyse item descriptions to identify products of particular interest to the user. Because the tiny print of different advising systems differed, the representation of objects was supported.

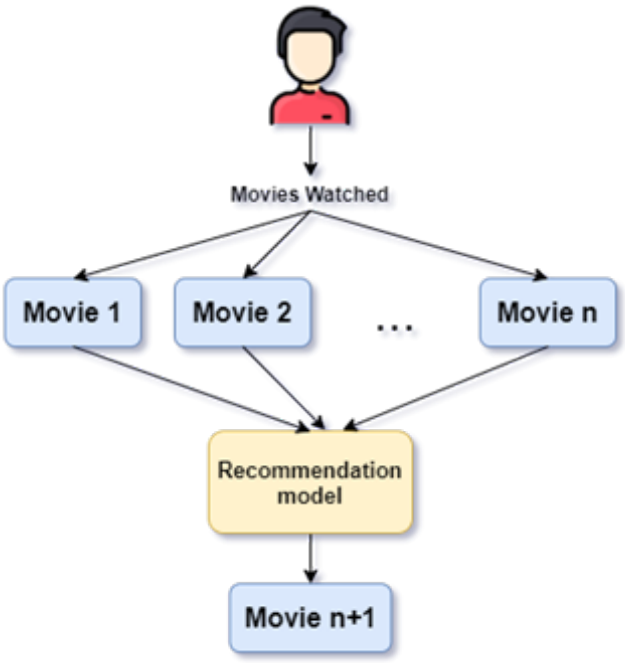

Figure 5.1: Content Based Filtering

Content-based recommendation systems strive to suggest products comparable to

those that a user has previously enjoyed. Indeed, a content-based recommender's core method compares the properties of a user profile, which stores preferences and interests, with the attributes of a content object (item) to suggest new exciting things to the user. The content-based Recommendation method creates a User-based Profile by searching for users' likes and dislikes. The total of the item profiles makes up the user profile, with the combination indicating the customer or user's ratings. The recommender system will recommend new goods or anything that has not yet been reviewed; however, the recommender system will not recommend things that are not in the same category as the items for which the user has submitted ratings. Content-based filtering algorithms generally fail when the user profile or location descriptions are not sufficiently constructed.

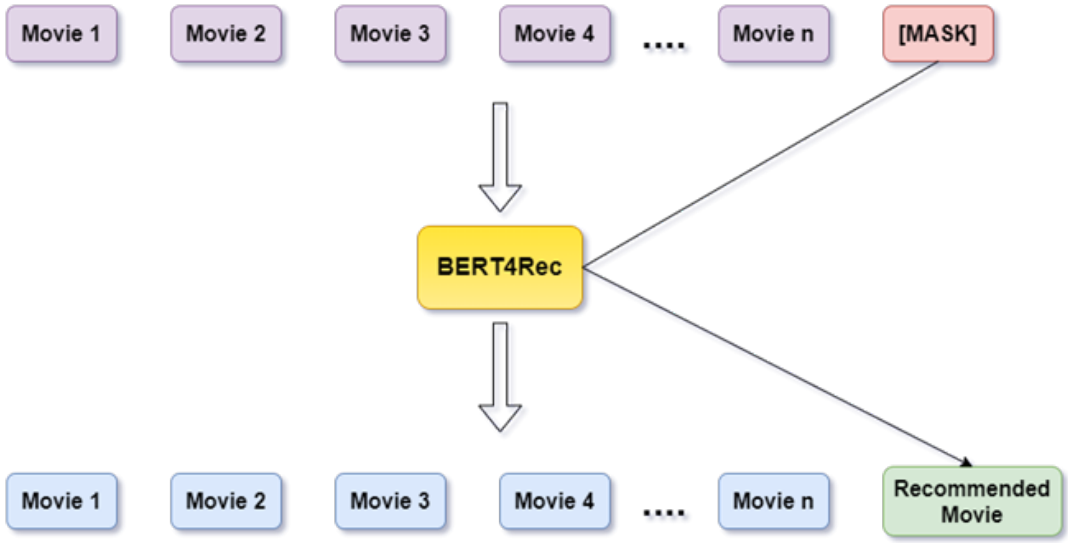

Figure 5.2: Abstract BERT4REC Model

Content-based filtering (CBF) is an approach for recommender systems that uses content about the objects in the system as the primary source of information. Tags, intrinsic properties such as size, length, colour, name, and so forth are examples. In the case of movies, the material might include meta-information such as the director, actors, genre, and release date. CB recommender systems usually offer recommendations by locating things similar to those that the user has previously enjoyed. Predictions are usually produced by constructing a profile of a person's interests based on the content of products that the user has previously liked, rated, or engaged with. When new items are compared to the user's profile, a relevance score is assigned to the user. While having a detailed user profile might be highly useful, this method has certain drawbacks if the item's content does

not include enough characteristics. Because these algorithms will only propose goods comparable to those that have already been rated. A user may miss out on items they may enjoy, even though items are relative to what they have previously engaged with.

5.2 Previous Literature

5.2.1 Recurrent Neural Networks

Recurrent Neural Networks, popularly known as RNNs, are introduced to solve the sequence learning problems. The input size is always fixed in the case of Convolutional Neural Networks (CNNs) and Feed-Forward Neural Networks. For example, the size of the input in the case of Alex-net architecture in Convolutional-Neural Networks is always fixed, equal to 227 x 227 x 3. Moreover, each input to the network is independent of future or previous inputs. Computations, Outputs, and Decisions for two successive inputs are entirely independent. Recurrent Neural Networks are typical architectures consisting of recurrence in their structure at each time step. They are to handle sequential data by considering the current and previously received inputs. The typical applications of RNNs include image captioning, Time series Prediction, Natural Language Processing, and Machine Translation.

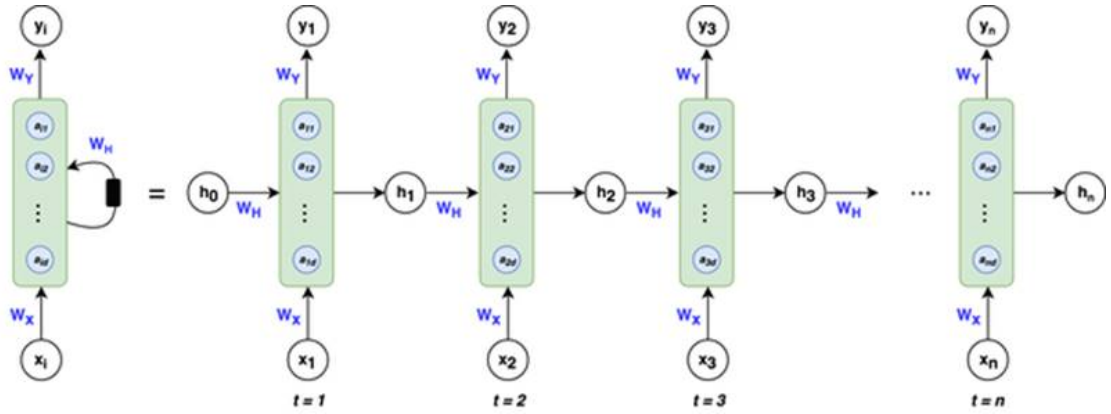

Figure 5.3: Recurrent Neural Networks

The recurrence relation of RNNs at each time step is given by the equation

$$\begin{aligned} h(t) &= f_{\text{ux}}\left(x(t), h(t-1)\right) \\ y(t) &= g_x(h(t), x(t)) \end{aligned} \tag{5.1}$$

Where $h(t)$ is the new state, $x(t)$ is the input at time step t, f_{yw} is the function with parameter c, and $h(t-1)$ is the old state. The various types of Recurrent

Neural Networks include Vanilla Networks, One to many Networks, many to one Networks and many to many Networks.

Back Propagation through Time: The most common loss function used in recurrent neural network architecture is cross-entropy loss given by:

$$E\left(y_t, \hat{y}_t\right) = -\sum_t y_t \log \hat{y}_t \tag{5.2}$$

Back Propagation through time is used to train various types RNNs, and it is based on Stochastic Gradient Descent. For one training example, all the gradients are summed up at each step. For any time step t, the equation is given by:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{5.3}$$

Vanishing Gradient Problem: A vanishing gradient problem is a situation in a neural network in which the gradients of deep neurons become too small to update the weights of that particular neuron. Vanishing Gradient Problem is usually caused by the activation functions like sigmoid and tan-h functions, whose derivative is small. The Vanishing Gradient Problem can be overcome using the ReLU activation function, Regularization, better initialization of weights, and using only short time sequences.

Exploding Gradient Problem: Exploding gradient problem is a situation in a neural network in which the gradients of deep neurons become too large to update the weights of that particular neuron. Exploding gradient problems can be overcome by clipping the gradients to a certain threshold. Exploding Gradient Problem will end up in resulting the NaN values during implementation. RNNs face long-term dependent problems as they connect previous information to the present task and face vanishing and exploding gradient problems. Long Short Term Memory and Gated Recurrent Units can be used to solve these issues.

5.2.2 Long-Short Term Memory (LSTMs)

LSTMs applications include image captioning, Time series Prediction, Natural Language Processing, and Machine Translation. LSTM Architecture typically consists of four gates that control the read, write and erase of information in a cell at time step t. The four different kinds of Gates are:

1. Cell state
2. Forget Gate

3. Input Gate

4. Output Gate

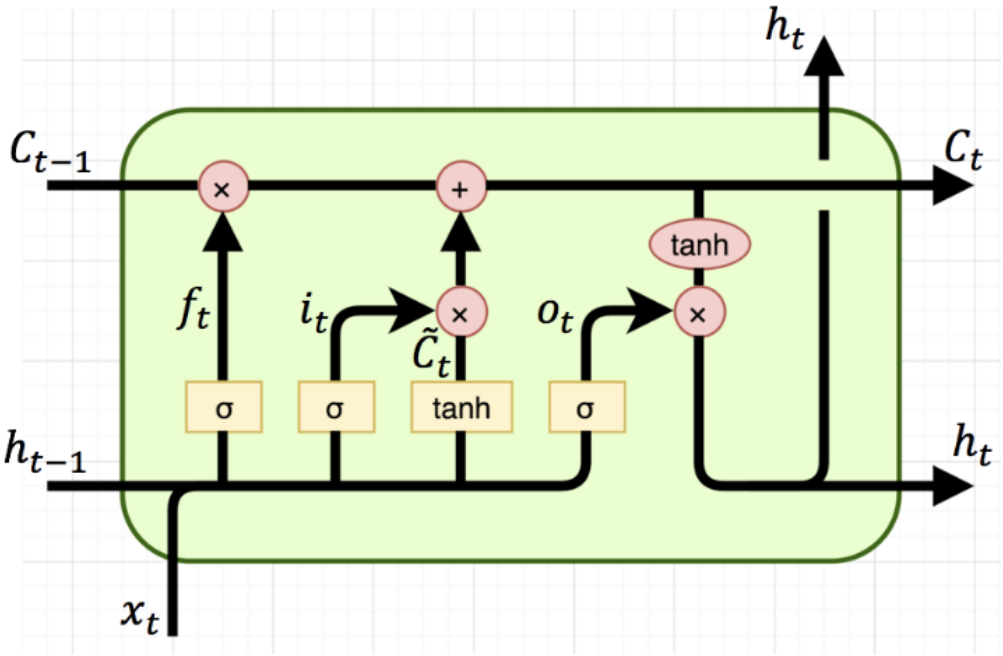

Figure 5.4: Long-Short Term Memory

Along with Cell State, Information can flow unchanged. It can add or remove the information in a cell state, regulated by the gates. As the name suggests, Forget gate keeps track of what information needs to be forgotten versus what information needs to remember across the various cells. Input Gate controls what information needs to be thrown out of the cell state. Output Gate controls what part of the information needs to be provided to the hidden state. LSTMs are trained using Back Propagation Through Time algorithm combined with gradient descent on the set of training sequences in a supervised fashion.

The various equations that represent cell state, forget gate, input gate, and output

gate are given by:

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ \tilde{c}_t &= \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ h_t &= o_t \circ \sigma_h\left(c_t\right) \end{aligned} \tag{5.4}$$

Where,
σ_g : sigmoid function.
σ_c : hyperbolic tangent function.
σ_h : hyperbolic tangent function or, as the peephole LSTM paper suggests, $\sigma_h(x) = x$.

The Vanishing Problem can be overcome by Gradient highway in the architecture. The gradient at Ct is passed on to Ct+1 unaffected except utilizing forget gate. Hence, the vanishing Gradient problem is minimal compared to Recurrent Neural Networks.

LSTM with Peephole Connections: The previous cell state information is provided as inputs in the Input Gate, Output Gate, and Forget gate computation. Here, The gates are allowed to peep into the cell state and then decide which dimension to cut down and which dimension to let through. The equations can be formulated as follows:

$$\begin{aligned} f_t &= \sigma\left(W_f \cdot [\boldsymbol{C}_{t-1}, h_{t-1}, x_t] + b_f\right) \\ i_t &= \sigma\left(W_i \cdot [\boldsymbol{C}_{t-1}, h_{t-1}, x_t] + b_i\right) \\ o_t &= \sigma\left(W_o \cdot [\boldsymbol{C}_t, h_{t-1}, x_t] + b_o\right) \end{aligned} \tag{5.5}$$

The Peephole connections in LSTM denote the memory cell activation contribution at time step t-1, which is nothing but the contribution of previous cell state C_{t-1}.

5.2.3 Gated Recurrent Units (GRUs)

Gated Recurrent was proposed in 2014 as an alternative to Long Short Term Memory Units. GRUs combine the forget gate and input gate into a single update gate. GRUs Merges cell state and a hidden state.

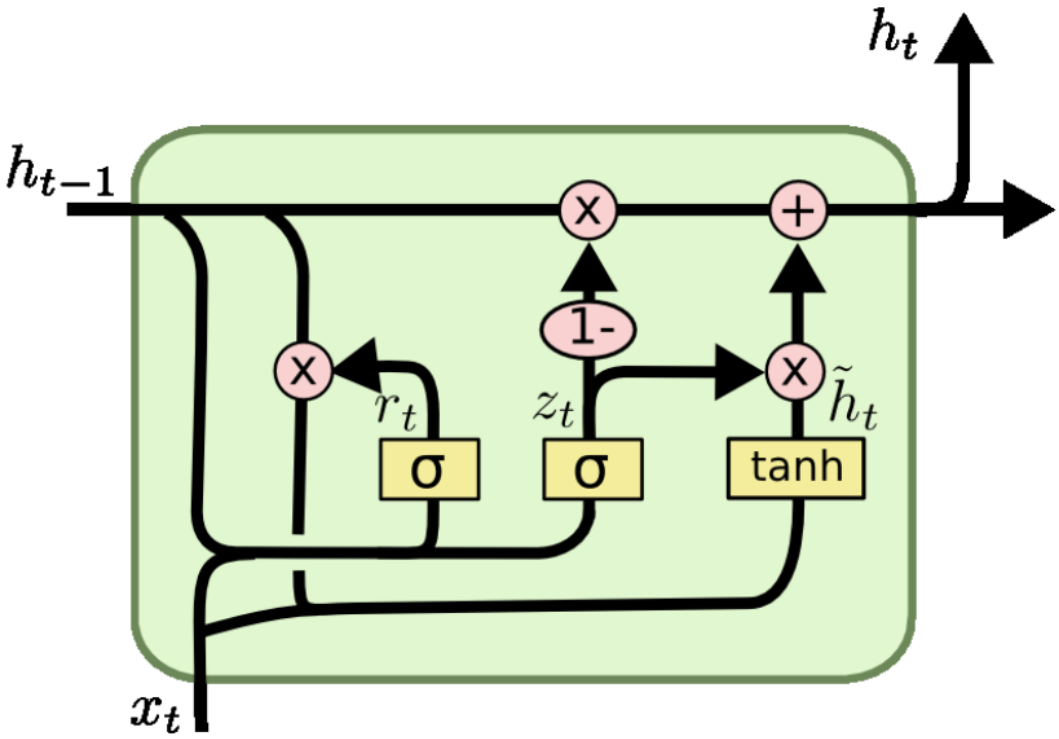

Figure 5.5: Gated-Recurrent Unit

Update Gate controls what part of hidden states are to be updated and preserved. The Reset gate controls what parts of previous hidden are used to compute the new content. The Hidden state updates gate simultaneously controls what is kept from the previous hidden state and what is updated to new hidden state content. The new hidden gate reset gate selects functional parts of previous hidden states.

$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \\ \tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \end{aligned} \tag{5.6}$$

Z_t is the update gate, r_t is the reset gate, h_t is the new hidden state content, and h_t is the hidden state.Gated Recurrent Units are helpful when many sequential units need to be trained as they have only two parameters at each time step compared to Long Short Term Memory, which has three parameters needed to be trained.

5.2.4 Transformers

The Transformer is a novel architecture that exhibits feed-forward neural network behavior and adopts a self-attention mechanism. Sequential computation prevents parallelization. Recurrent Neural Networks are feedback networks that require temporal dependency on weights. RNNs, LSTMs, and GRUs still need prominent attention mechanisms to address long-range dependencies. Transformers paved the way for modern pre-trained models like BERT and Open-AI GPT. The Transformer is an encoder-decoder architecture consisting of a stack of encoders and decoders. Each encoder and decoder consists of Feed-forward Neural networks and Self-attention layers. Self-attention layers help to solve long-range dependencies in Sequential problems.

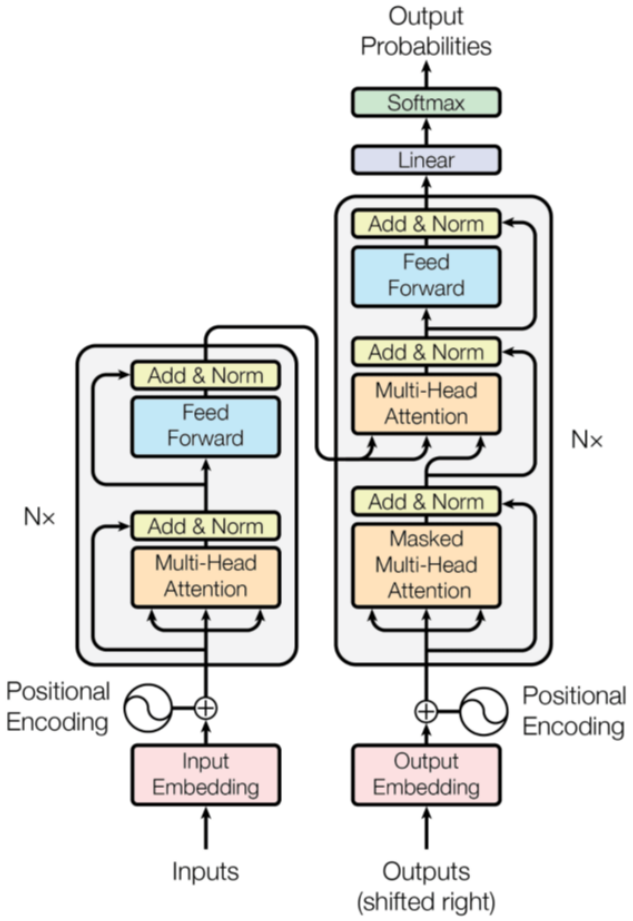

Figure 5.6: Transformer[8]

Self-attention Mechanism: Three matrices lie at the core of the self-attention

mechanism. They are Query, Key, and Vector. The term Query refers to an act of inquiry and calculates the influence of every other embedding vector on the given embedding vector. The term Key refers to the location where the inquiry will happen.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.7}$$

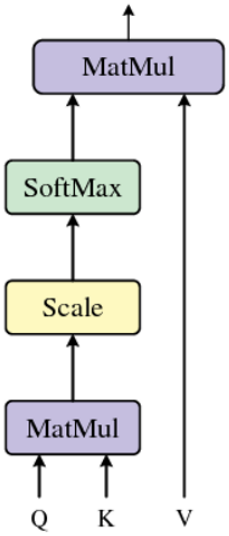

Figure 5.7: Self Attention Mechanism

Multi-head attention: Multi-head attention is the concatenation of several multi-head attention layers. Multi-head attention layers help to explore a wide variety of sub-spaces and dimensions.

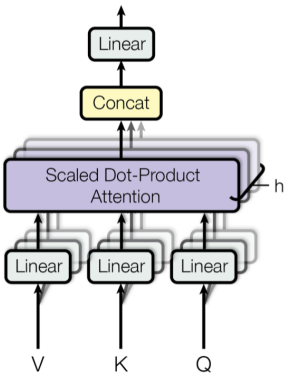

Figure 5.8: Multi Head Attention Mechanism

$$\begin{aligned} \text{MultiHead}\left(Q, K, V\right) &= \text{Concat}\left(\text{ head }_1, \ldots, \text{ head }_h\right) \mathrm{W}^0 \\ \text{where } head_i &= \text{Attention}\left(\mathrm{W}_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{5.8}$$

Transformers are quite advantageous over Recurrent Neural Networks, Long-short Term Memory, and Gated Recurrent Units in terms of Parallel computation and exploiting long-range dependencies in the sequential learning problems.

5.2.5 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers is a stack of transformer encoders pre-trained on two tasks, namely, Masked Language Modelling and Next Sentence Prediction.

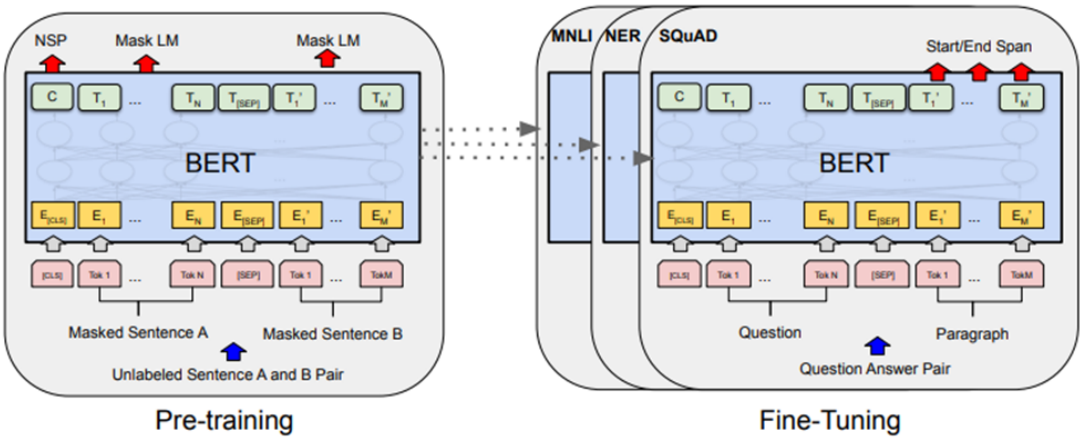

Figure 5.9: BERT[11]

The term "Masked Language Modeling"[11] refers to the Masking of tokens during the initial phase, and the model is made to predict them. The term "Next Sentence Prediction"[11] refers to the model trained whether a given sentence is following the given first sentence probabilistically.

BERT is intended to condition both left and the right context to pre-train deep bidirectional representations from the unlabeled text. As a result, using just one additional output layer, the pre-trained BERT model may be fine-tuned to generate state-of-the-art models for a wide range of NLP tasks.

5.2.6 Sequential Recommendation

The previous works on the sequential recommendations based on user behavioral sequence used Markov Chains (MCs) and a few organized recommendation problems as sequential optimality problems. They adopted Markov Decision Processes (MDPs) to solve the above optimization problems. Recently, For modeling, Sequential recommendations, RNNs[9], LSTMs[9], and GRUs have proven prominent. RNN and its Variants adopt left-to-right unidirectional mechanisms to model the user behavioral Sequence. The main idea of the models is to encode user behavior sequences into a vector and process them with different architectures and loss functions.

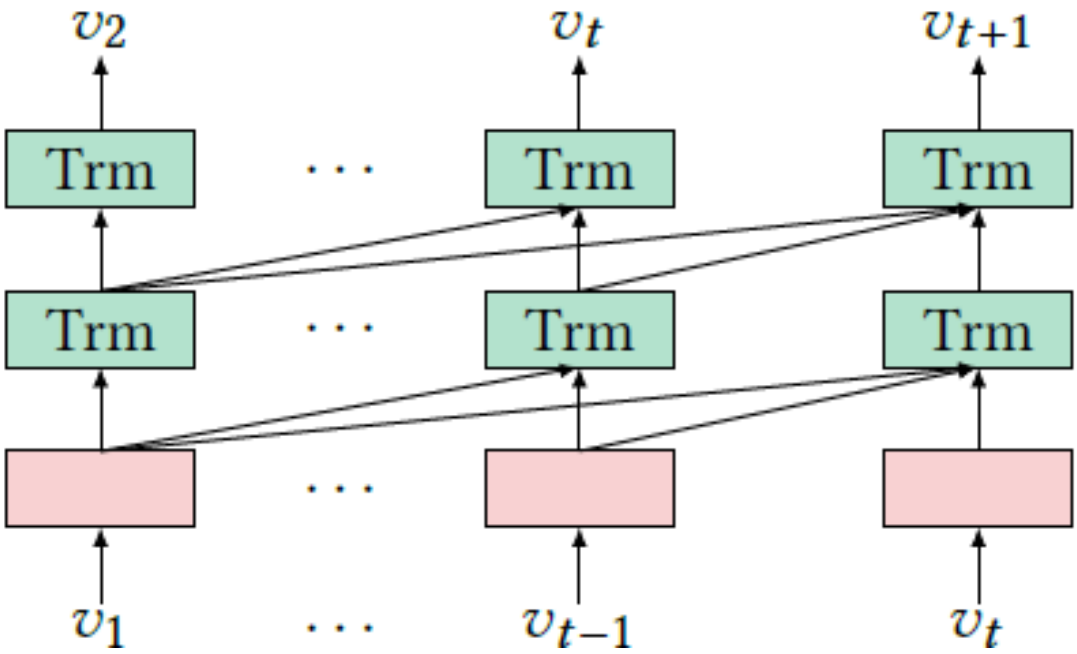

Figure 5.10: Architecture of RNN[9] and its variants

Self-Attentive Sequential Recommendation (SASREC)[10] uses a self-attention mechanism unidirectionally to model sequential recommendations for the user's dynamic preference. The model uses binary cross-entropy as the loss function, and the output is the predicted masked token. The difference between the SASREC[10] and BERT4REC is that SASREC[10] is a unidirectional model, whereas BERT4REC is a bidirectional model.

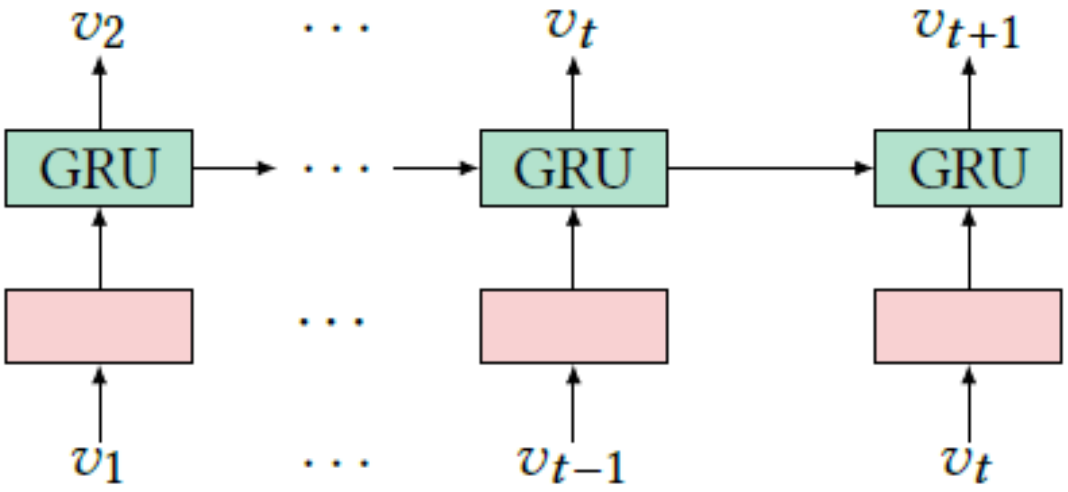

Figure 5.11: SASREC Model Architecture[10]

5.3 Model Training

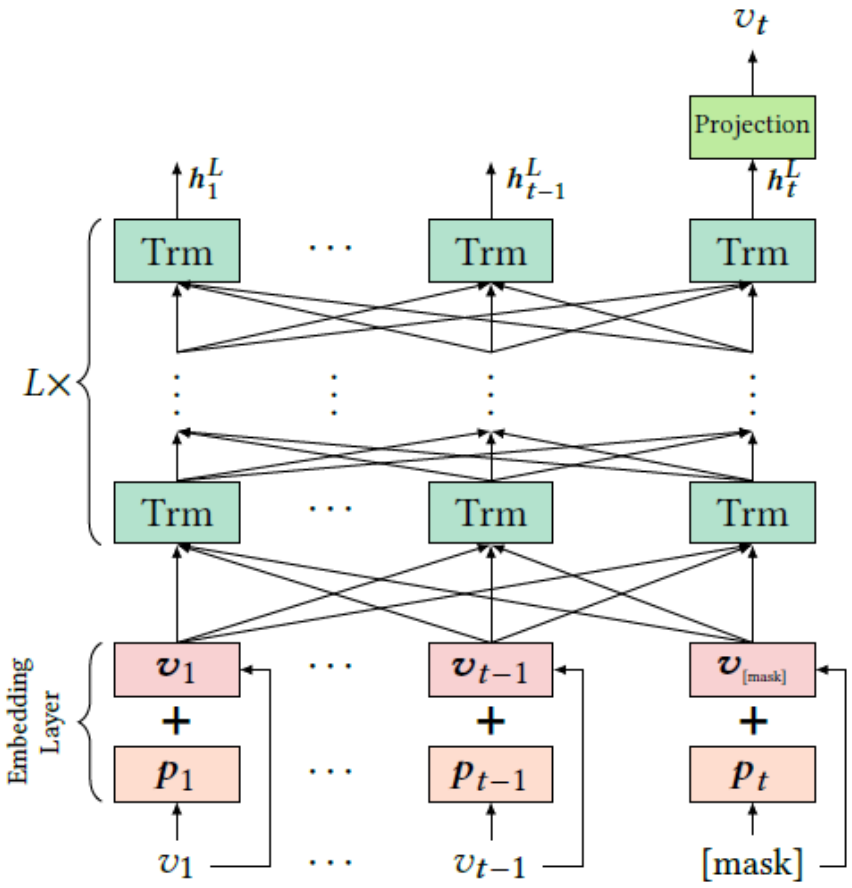

Figure 5.12: BERT4REC Model Architecture[2]

During the training phase of BERT4REC[2], a random proportion of movies are chosen, and then BERT4REC is made to predict the masked tokens.

$$\text{Input: } [v_1, v_2, v_3, v_4, v_5] \xrightarrow{\text{randomly mask}} [v_1, [\text{mask}]_1, v_3, [\text{mask}]_2, v_5]$$
$$\text{Labels: } [\text{ mask }]_1 = v_2, \quad [\text{ mask }]_2 = v_4$$

The final hidden representations of the masked tokens are fed to the output, and finally, the loss function is defined as a negative log-likelihood as follows:

$$\mathcal{L} = \frac{1}{|\mathcal{S}_u^m|} \sum_{v_m \in \mathcal{S}_u^m} -\log P\left(v_m = v_m^* \mid \mathcal{S}_u'\right) \tag{5.9}$$

where $\mathcal{S}'_u$ is the masked version for user behavior history $\mathcal{S}_u, \mathcal{S}^m_u$ is the random masked items in it, v^*_m is the true item for the masked item v_m, and the probability $P(\cdot)$ is defined in Equation 6.9 . As BERT4REC is similar to BERT, the output of self-attention[8] layers is then passed to a position-wise feed-forward neural network and then to a stack of transformer encoders. The equations can give the above formulation:

$$\begin{aligned} \mathrm{PFFN}\left(H^l\right) &= \left[\mathrm{FFN}\left(h_1^l\right)^{\mathrm{T}}; \ldots; \mathrm{FFN}\left(h_l^l\right)^{\top}\right]^{\mathrm{T}} \\ \mathrm{FFN}(x) &= \mathrm{GELU}\left(xW^{(1)} + b^{(1)}\right) W^{(2)} + b^{(2)} \end{aligned} \tag{5.10}$$

PPFN is a position-wise feed-forward Neural network, GELU is Gaussian Error Linear Unit, and FFN is Feed-forward Neural Network.

$$\begin{aligned} H^l &= \mathrm{Trm}\left(H^{l-1}\right), \quad \forall i \in [1, \ldots, L] \\ \mathrm{Trm}\left(H^{l-1}\right) &= \mathrm{LN}\left(A^{l-1} + \mathrm{Dropout}\left(\mathrm{PFFN}\left(A^{l-1}\right)\right)\right) \\ A^{l-1} &= \mathrm{LN}\left(H^{l-1} + \mathrm{Dropout}\left(\mathrm{MH}\left(H^{l-1}\right)\right)\right) \end{aligned} \tag{5.11}$$

Where H is the hidden layer output which is at depth l,LN is the layer Normalization, The final output of the BERT4REC is given by the equation by

$$P(v) = \mathrm{sotmax}\left(GEEL\left(h_t^L W^P + b^p\right) E^{\top} + b^0\right) \tag{5.12}$$

where $\boldsymbol{W}^P$ is the learnable projection matrix, $\boldsymbol{b}^P$, and $\boldsymbol{b}^O$ are bias terms, $E \in R^{|\cdot V| \times d}$ is the embedding matrix for the item set $\mathcal{V}$. We use the shared item embedding matrix in the input and output layer for alleviating overfitting and reducing model size. Testing of BERT4REC is done by appending a masked token to the end of the user's behavioral sequence, and then the masked token is made to predict by the model.

5.4 Evaluation

The most recent Movie-Lens dataset was used, which included 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 Movie-Lens users. The data for the ratings was split as 70:15:15 between training, validation and testing. The training and validation losses vs epochs graph is plotted. Refer Figure 5.15

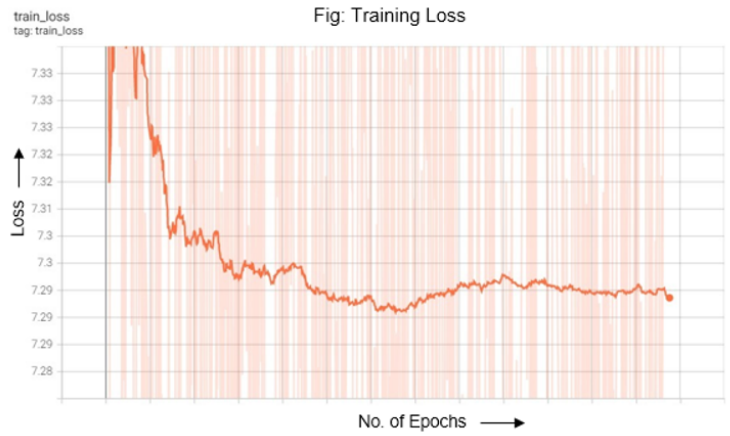

Figure 5.13: Training Loss

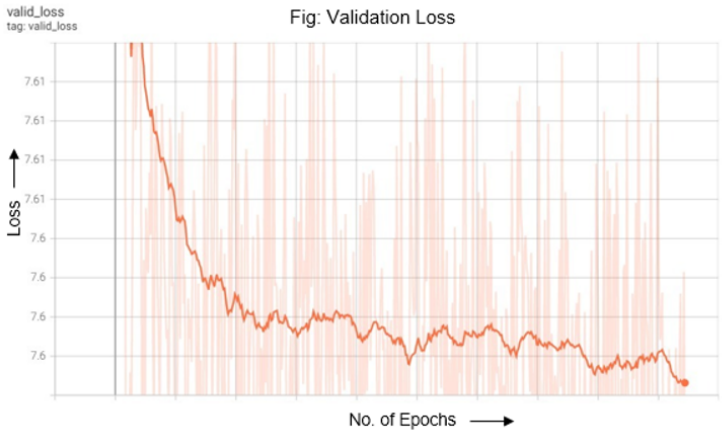

Figure 5.14: Validation Loss

The metric we have used for BERT4REC Evaluation is Mean Average Precision. Average precision measures relevant products recommended in the first K results. Average precision is calculated using precision and recall in the ranked sequence of recommendations at every position.

$$AP = \sum_{1}^{x} (\text{ precision at } i) \times (\text{ change in recall at } i) \tag{5.13}$$

The mean of the average precision measured across all the instances in the dataset of users is called the Mean Average Precision. Mean average precision is a rankless metric used for evaluating recommendation systems. We have adopted Mean

average precision to evaluate BERT4REC. Mean average precision is given as:

$$MAP = \frac{\sum_{u=1}^{N} AP_u}{M} \tag{5.14}$$

The method we have used for evaluation of BERT4REC is that we used the 20% test data set and sorted the watched movies based on the timestamp for all the users present in the test dataset and taken the top ten highest rated movies for all the users individually and kept the last one as the ground truth and sent the first nine movies as input to the BERT4RC model and matched the ground truth movie with the top ten recommendations predicted by the model for all the users and calculating the average precision value and finally taking mean to get Mean Average Precision value.

5.5 Results

We got the Mean Average Precision Value of 0.277 for the BERT4REC Model.

Fig: BERT4Rec Input

```
['Mummy, The (1959)',
 "Mummy's Curse, The (1944)",
 'Message in a Bottle (1999)',
 'McCabe & Mrs. Miller (1971)',
 'Mary Poppins (1964)',
 'Man with the Golden Gun, The (1974)',
 'Man Who Would Be King, The (1975)',
 'Maltese Falcon, The (1941)',
 'Mad Max (1979)',
 'Lolita (1997)',
 'Lethal Weapon (1987)',
 'League of Their Own, A (1992)',
 'Goldfinger (1964)',
 'Ghostbusters (1984)',
 'Getaway, The (1994)',
 'Gaslight (1944)',
 'From Dusk Till Dawn (1996)',
 'Friday the 13th Part VII: The New Blood (1988)',
 'Friday the 13th Part V: A New Beginning (1985)',
 'French Connection, The (1971)',
 'Frankenstein (1931)',
 'Foxfire (1996)',
 'Foreign Correspondent (1940)',
 'For a Few Dollars More (1965)',
 'Force 10 from Navarone (1978)',
 'Fight Club (1999)',
 'Fighting Seabees, The (1944)',
 'Falcon and the Snowman, The (1984)',
 'Face/Off (1997)',
 'Exorcist II: The Heretic (1977)']
```

BERT4Rec

Fig: BERT4Rec Output

```
["One Flew Over the Cuckoo's Nest (1975)",
 'Aliens (1986)',
 'Alien (1979)',
 'Terminator, The (1984)',
 'Gladiator (2000)',
 'Stand by Me (1986)',
 'Wizard of Oz, The (1939)',
 'Shakespeare in Love (1998)',
 'Star Wars: Episode I - The Phantom Menace (1999)',
 'Total Recall (1990)',
 'Toy Story 2 (1999)',
 'Fugitive, The (1993)',
 'Men in Black (1997)',
 'Titanic (1997)',
 'Airplane! (1980)',
 'Babe (1995)',
 'Usual Suspects, The (1995)',
 'Galaxy Quest (1999)',
 'Sixth Sense, The (1999)',
 'Breakfast Club, The (1985)',
 'Blade Runner (1982)',
 'Election (1999)',
 'Groundhog Day (1993)',
 '2001: A Space Odyssey (1968)',
 'Casablanca (1942)',
 "Bug's Life, A (1998)",
 'Pulp Fiction (1994)',
 'Godfather: Part II, The (1974)',
 'Abyss, The (1989)',
 'Forrest Gump (1994)']
```

Fig: MAP Score for BERT4Rec

```
print("Mean Average Precision of BERT4Rec model = ", sum1/count1)

Mean Average Precision of BERT4Rec model =  0.2778572935105253
```

Figure 5.15: BERT4REC Results

Chapter 6

Hybrid Approach

6.1 Introduction

A hybrid recommender system is one that generates an output by combining different recommendation algorithms. When comparing collaborative or content-based recommender systems to hybrid recommender systems, the suggestion in hybrid systems, accuracy is higher. The cause for this is a lack of knowledge about the situation domain dependencies in collaborative filtering, as well as people's preferences in collaborative filtering a system focused on content The combination of the two results in an increase in common knowledge, which is beneficial. improves the quality of recommendations. It is especially important because of the increase of knowledge promising to investigate new methods for extending the core collaborative filtering algorithms using user behavior data with content data and content-based algorithms.

For proposing products or objects to the user, recommendation systems are widely utilized in a number of applications. Content-based and collaborative filtering are two prevalent ways for filtering recommendations. When there isn't enough data to learn the relationship between the user and the items, these solutions run into problems. In such instances, the Hybrid Recommendation System, a third type of technique, is utilized to develop recommendation systems. The disadvantages of both content-based and collaborative filtering methods are overcome with this approach.

6.2 Research Problems

Cold Start: When the system is unable to develop any relationship between people and products for which it lacks sufficient data, it is said to be experiencing a

cold start. Cold-start issues can be divided into two categories.Problems with user cold-starting: The user cold-start problem occurs when there is almost no information about the user available. Product cold-start issues: A product cold-start issue occurs when there is almost no knowledge about the product. The system fails to deliver recommendations for three key reasons: Systematic Bootstrapping, Low Interaction (Few instances available), and the introduction of new users.

Data Sparsity: This issue arises from the fact that, in most cases, consumers only rate a small portion of the available items, especially when the catalog is enormous. As a result, the user-item rating matrix is sparse, with insufficient data to identify related individuals or products, lowering the quality of the suggestions. In CF recommender systems that rely on peer feedback for recommendations, data scarcity is common. A factorization model of the triadic relation user-item-domain is used to solve the data sparsity of cross-domain recommendations. The problem of data sparsity can also be overcome by treating each user-item rating as a predictor of missing ratings.

Accuracy: The capacity of a recommendation system to correctly forecast each user's item preferences is known as recommendation accuracy. Since the beginning of recommendation systems, there has been a lot of focus on improving recommendation accuracy. A recommender system for online material is created. Based on the user's navigation history, the writers construct the user's long-term interest. The user's profile is then compared against the content of the website to determine whether or not to recommend it.

Scalability: This is a tough attribute to achieve because it is linked to the amount of users and things the system is designed to handle. Unless it is designed to be very scalable, a system meant to propose a few goods to hundreds of users will most likely fail to recommend hundreds of items to millions of individuals. The usage of this compressed dataset enhances scalability, reduces sparsity, and reduces the system's computing time by a little amount.

Diversity: This is a desirable trait that has recently gotten a lot of attention. It's crucial to have a variety of suggestions since it helps to minimize popularity bias. Having a recommendation list containing items that are quite similar to each other (for example, showcasing all the episodes of a popular series) is the latter. A person who isn't interested in one of them isn't likely to be interested in any of them, so the recommendation list is useless.

Others: These are some of the additional issues that have been raised in a few

research. Lack of Personalization, Privacy Preservation, Noise Reduction, Data Source Integration, Lack of Novelty, and User Preference Adaptiveness are just a few of them.

6.3 Types of Hybrid Systems

1. Weighted

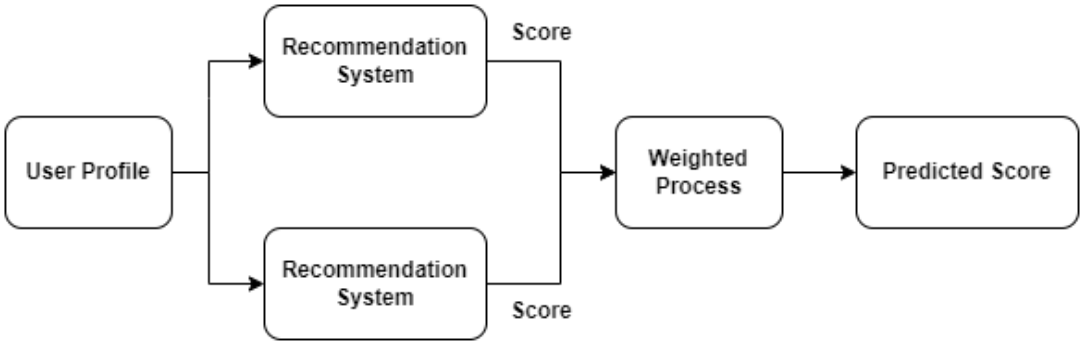

Figure 6.1: Weighted Hybrid Recommendation System

We can define a few models in the weighted recommendation system that can read the dataset well. The weighted recommendation system will integrate the outputs from each of the models into static weightings that will remain constant across the train and test set. For example, we can combine a content-based model with an item-by-item collaborative filtering model, with each contributing 50% to the final prediction.
The weighted hybrid has the advantage of integrating different models to support the dataset on a linear recommendation process.

2. Switching

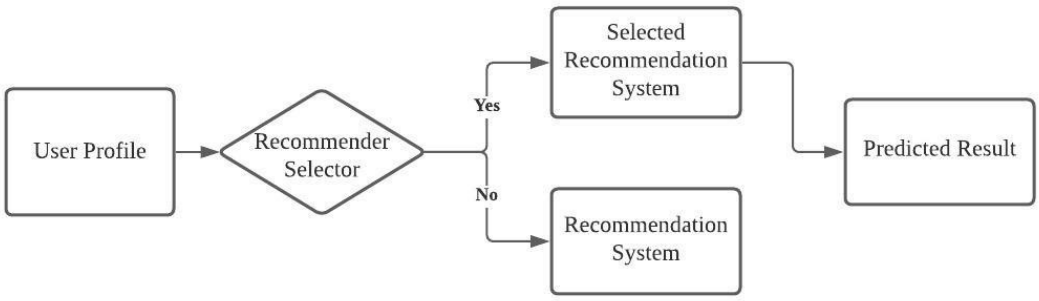

Figure 6.2: Switching Hybrid Recommendation System

Depending on the situation, the switching hybrid selects a single recommendation system. We should establish the recommender selector criteria based on the user profile or other features while building the model for the item-level sensitive dataset. On top of the recommendation model, the switching hybrid technique adds an additional layer that selects the best model to utilize. The constituent recommendation model's strengths and weaknesses are taken into account by the recommender system.

3. Feature Combination

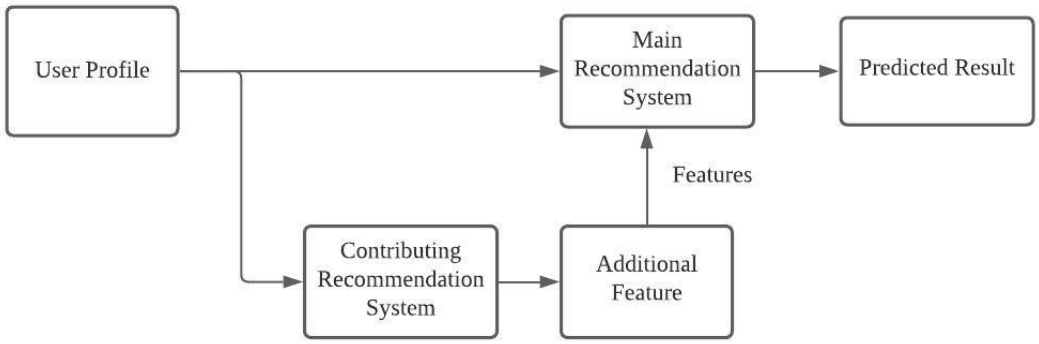

Figure 6.3: Feature Combination Hybrid Recommendation System

We add a virtual contributing recommendation model to the system in feature combination hybrid, which acts as feature engineering for the original user profile dataset. We can infuse collaborative recommendation model characteristics into a content-based recommendation model. The hybrid model can take into account collaborative data from the subsystem without relying just on one model.

4. Cascade

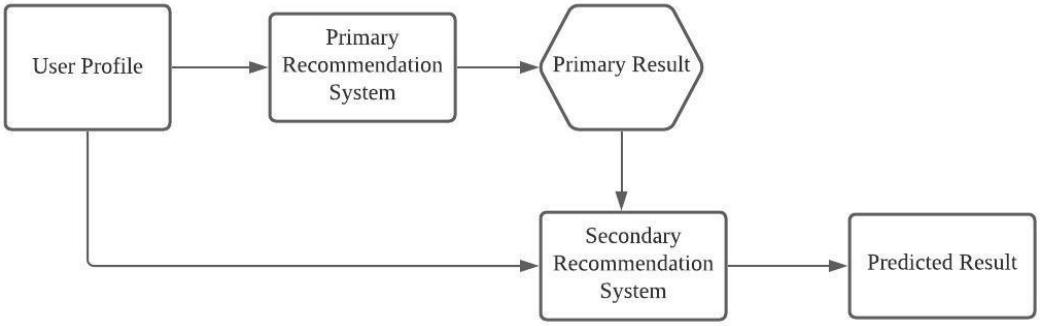

Figure 6.4: Cascade Hybrid Recommendation System

Cascade hybrid is a rigorous hierarchical structure recommendation system in which the primary recommendation system produces the primary result and the secondary model is used to handle small concerns with the primary result, such as breaking a tie in score. Because most datasets are scarce in practice, the secondary recommendation model can help with issues like equal scoring or missing data.

6.4 Our Approach

Cascade Hybrid Model: Pipe-lining both the collaborative filtering and content based filtering models in a cascade fashion to design a novel hybrid recommendation system.

A phased recommendation process is exemplified by cascade hybrid models. A first strategy is used to establish a rough ranking of candidate items, and then a second technique is used to refine the list based on the preliminary candidate set. Order of the recommendation systems matters in cascades. We have considered a random User ID as input to the pipe-line model and the output is a list of predictions to that user. We have considered a random User ID as input to the pipeline model and the output is a list of predictions to that user.

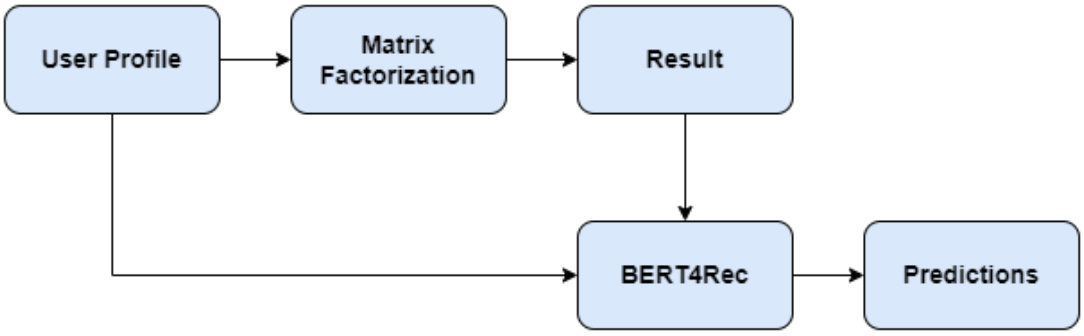

Figure 6.5: Cascade hybrid approach

We have extended the same idea of the Mean Average Precision evaluation implemented for the BERT4Rec model to the hybrid model and observed a significant increase in the MAP score. We have observed approximately an 18% rise in the MAP score, which indicates that the recommendation system's hybrid approach is enhancing the user's experience by generating more relevant recommendations.

6.5 Results

The recommendations of one technique are refined by another recommendation technique.

The outputs from MF model based on ratings are cascaded to the BERT4Rec model along with the user profile, which in turn produces recommendations specific to that user.

We have taken 20% of our dataset as test data and sorted the list according to the ratings given and corresponding timestamp. Taking the top 10 movies occured for some user and setting aside the last movie as ground truth, we have calculated the Mean Average Precision (same as mentioned earlier for BERT4REC evaluation) and we have observed a significant improvement in the MAP Score when compared to the content based filtering alone.

Fig: Input is test dataset (Example user ID =270)

	userId	movieId	rating	timestamp	title
573101	270	1639	5	972669546	Man Who Would Be King, The (1975)
296897	270	1276	5	972669521	Maltese Falcon, The (1941)
215948	270	627	5	972669474	Mad Max (1979)
507133	270	384	5	972669159	Lethal Weapon (1987)
221497	270	955	5	968626656	Goldfinger (1964)
...	...	...	...	...	...
987784	270	3238	3	968622946	Adventures of Elmo in Grouchland, The (1999)
819337	270	1615	3	968622902	Aces: Iron Eagle III (1992)
975735	270	2057	3	968622839	52 Pick-Up (1986)
967469	270	2748	2	968625819	Exorcist II: The Heretic (1977)
948243	270	404	2	968625819	Exorcist III, The (1990)

90 rows × 5 columns

Fig: MAP Score for Hybrid model

```
print("Mean Average Precision for Hybrid model = ",sum1/count1)
```

```
Mean Average Precision for Hybrid model =  0.32803170385133684
```

Fig: Hybrid model output

```
['Braveheart (1995)',
 'Being John Malkovich (1999)',
 'Men in Black (1997)',
 'Fargo (1996)',
 'L.A. Confidential (1997)',
 'Princess Bride, The (1987)',
 'Shakespeare in Love (1998)',
 'Star Wars: Episode I - The Phantom Menace (1999)',
 'E.T. the Extra-Terrestrial (1982)',
 'Groundhog Day (1993)',
 'Alien (1979)',
 'Pulp Fiction (1994)',
 'Forrest Gump (1994)',
 'Terminator, The (1984)',
 'Fugitive, The (1993)',
 'Gladiator (2000)',
 'Ghostbusters (1984)',
 'Godfather: Part II, The (1974)',
 'Blade Runner (1982)',
 "Bug's Life, A (1998)",
 'Usual Suspects, The (1995)',
 'Abyss, The (1989)',
 'Aliens (1986)',
 'Galaxy Quest (1999)',
 'Total Recall (1990)',
 '2001: A Space Odyssey (1968)',
 'Babe (1995)',
 'Stand by Me (1986)',
 'GoodFellas (1990)',
 'Jaws (1975)']
```

Figure 6.6: Results of the hybrid recommendation system approach

Chapter 7

Conclusion

To implement the system, a hybrid technique of content-based filtering and collaborative filtering is used. This method solves the shortcomings of each individual algorithm while also improving the overall system performance. We designed and assessed a novel hybrid movie recommendation system that utilizes a cascade approach and applies the BERT4Rec framework along with Matrix Factorization. We evaluated the collaborative and Content based models individually and designed the hybrid model. The hybrid system significantly improved the results.

The hybrid approach enabled us to investigate the performance of BERT4Rec combined with collaborative filtering, and avoid disadvantages of pure approaches such as the lack of information about the domain dependencies in collaborative filtering and people's preferences in content-based systems. Future prospects include more options for embedding keywords in BERT4Rec model, that allows users to add items such as genres of a movie, plot, cast, etc. to further enhance the model's performance.

Further work can be done on a hybrid recommender that uses clustering and similarity to improve performance. Our technique of Hybrid recommendation can also be used to implement recommendation systems for songs, videos, venues, news, books, tourism, e-commerce, etc.

Bibliography

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems,"Computer, vol. 42, no. 8, pp. 30–37, 2009

[2] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019.BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3357384.3357895

[3] Manoj Kumar, D K Yadav, Ankur Singh and Vijay Kr. Gupta. Article: A Movie Recommender System: MOVREC. International Journal of Computer Applications 124(3):7-11, August 2015

[4] Luis M. de Campos, Juan M. Fernández-Luna *, Juan F. Huete, Miguel A. Rueda-Morales;"Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks", International Journal of Approximate Reasoning, revised 2010.

[5] Harpreet Kaur Virk, Er. Maninder Singh," Analysis and Design of Hybrid Online Movie Recommender System "International Journal of Innovations in Engineering and Technology (IJIET) Volume 5 Issue 2,April 2015.

[6] S. Agrawal and P. Jain, "An improved approach for movie recommendation system", 2017 International Conference on I-SMAC (IoT in Social Mobile Analytics and Cloud) (I-SMAC), pp. 336-342, 2017.

[7] Costin-Gabriel Chiru, Vladimir-Nicolae Dinu , Ctlina Preda, Matei Macri ; "Movie Recommender System Using the User's Psychological Profile" in IEEE International Conference on ICCP, 2015

[8] Vaswani, Ashish, Noam, Shazeer, Niki, Parmar, Jakob, Uszkoreit, Llion, Jones, Aidan N., Gomez, Lukasz, Kaiser, and Illia, Polosukhin. "Attention Is All You Need." (2017).

[9] Alex Sherstinsky, . "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network".Physica D: Nonlinear Phenomena 404 (2020): 132306.

[10] Kang, Wang-Cheng, and Julian, McAuley. "Self-Attentive Sequential Recommendation." (2018).

[11] Devlin, Jacob, Ming-Wei, Chang, Kenton, Lee, and Kristina, Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." (2018).