FEATURE ANALYSIS FOR RECOMMENDER SYSTEMS USING TRANSFORMER-BASED ARCHITECTURES

by

Emre Boran

B.A., Economics, Boğaziçi University, 2019

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I express my deepest gratitude to my advisor, Professor Tunga Gungor, for his unwavering support throughout my master's studies. I am incredibly grateful to have had the opportunity to work with him, as he was always patient, enthusiastic, and an exceptional mentor during every stage of this study.

A big thank you to Telecommunications and Informatics Technologies Research Center (TETAM) for giving me the environment to run my codes.

I am incredibly grateful to my mother, Zeynep Boran, my father, Burhanettin Boran, and my brother Tahir Boran, who have always been there for me and supported me throughout my journey.

I want to express my sincere gratitude to Buğra Çil for introducing me to the field of computer science and for providing constant support.

I sincerely thank my dear wife, Ayça Kurt Boran, who has always supported me and is my constant source of encouragement and motivation.

ABSTRACT

FEATURE ANALYSIS FOR RECOMMENDER SYSTEMS USING TRANSFORMER-BASED ARCHITECTURES

Recommender systems are technology-based solutions that assist users by suggesting relevant items among millions of items. It could be anything like a movie, a meal, a vacation spot, shoes, or a piece of music. Unlike traditional recommender systems, sequential and session-based recommender systems make recommendations by paying attention to the order of items that users interact with. The advantage of such systems is that they take into account varying tastes. Additionally, due to some legal requirements, the users' data cannot be collected from some platforms, and the recommender system has to suggest the session's information without having any previous knowledge. It may only have to recommend products according to a few interactions in that session. These reasons constitute the importance of sequential and session-based recommender systems. In this thesis, we have experimented with sequential and session-based recommender systems using the Transformers4rec framework, which allows us to use transformer architectures in recommender systems. We observed that transformer architectures work better in short interaction sequences than long ones. We showed that additional features enhance the model's performance, particularly time-based features. Additionally, we examined and interpreted that the importance of features changes according to the size, shape, and type of data.

ÖZET

TRANSFORMATÖR TABANLI MİMARİLER KULLANAN TAVSİYE SİSTEMLERİ İÇİN ÖZNİTELİK ANALİZİ

Tavsiye sistemleri; milyarlarca öge arasından, onlar ile ilgili ögeleri kullanıcılara önermeye yardımcı olan teknoloji temelli çözümlerdir. Bu; bir film, yemek, tatil yeri, ayakkabı veya bir müzik parçası gibi herhangi bir şey olabilir. Sıralı ve oturum tabanlı tavsiye sistemleri, geleneksel tavsiye sistemlerinden farklı olarak kullanıcıların öğelerle etkileşim sırasına dikkat ederek öneriler yaparlar. Bu tür sistemlerin avantajı değişen zevkleri dikkate almalarıdır. Ayrıca, bazı yasal gereklilikler nedeniyle zaman zaman kullanıcıların verileri toplanamamaktadır ve tavsiye sistemi o oturumda elde edilen bilgilerle öneri yapmak zorunda kalmaktadır. Bu gibi nedenler, sıralı ve oturum tabanlı tavsiye sistemlerinin önemini oluşturur. Bu tezde, Transformers4rec framework kullanarak sıralı ve oturum tabanlı tavsiye sistemleriyle deneyler yapılmıştır. Transformatör mimarilerinin kısa etkileşim serilerinde daha iyi çalıştığı gözlemlenmiştir. Özellikle zaman tabanlı öznitelikler olmak üzere, ek özniteliklerin sonuçları iyileştirdiği gösterilmiştir. Ayrıca, sonuçların veri boyutu, şekli ve türüne göre değiştiği incelenmiş ve yorumlanmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ETL	Extract Transform Load
NLP	Natural Language Processing
RecSys	Recommender Systems

1. INTRODUCTION

The Internet is ubiquitous nowadays. We feel as if it always exists throughout history. We read the news on the Internet, are informed about the world, order food using the Internet, and socialize on it. That is, we meet almost all our human needs with it. As a result of the penetration of the Internet into all areas of our lives, there is too much information, products, or services on the Internet. When looking for something, we may need help choosing the information being serviced. For example, suppose we are writing a paper on a subject; we searched and clicked on a few of the 10-20 related results, but we still need to find what we are looking for. In another case, suppose we listened to some music that we always listen to repeatedly, and we wanted to hear something new and similar. However, we do not know how to find it. Alternatively, we went on a vacation last year and want to go somewhere similar. At this point, a solution could be to evaluate all possible options and choose the one that suits us. However, this is a challenging goal. Because there are so many options available for every field that might interest us, this is where recommender systems come into play. It can allow us to find what we are looking for in too many options. It recommends to us suitable paper, movie, music or holiday. A user can save time significantly by the chance of getting a good recommendation.

Also, since a user cannot evaluate every option, he/she would probably have decided on an option that is not the best. However, the recommender system can offer the best possible options if it works well. Alternatively, the system can introduce users to something new. In this way, it accelerates and improves the decision-making process. For example, there are millions of videos on Youtube. Sometimes people come to spend time, not because they are looking for something specific. If the videos randomly fall on the screen, users cannot find their favorite videos and leave the application, which may lead to less use of Youtube.

Making good recommender systems is not just beneficial for users. Moreover, the

people making the recommender system also benefit from this. Better recommendations will directly affect companies' income since they make users happy to use their service. For example, a company can increase sales by making it easy for customers to find and purchase the products they need. If it is a news provider, their usage will increase because people spend more time on the news website if they find more suitable content for their preferences. If we give an example from another business, 75% of Netflix's usage comes from a personalized recommender system [1]. It shows how vital the recommender system is. In addition to the financial returns, Greg Linden, who made the first recommender system on Amazon, also mentioned that he is happy to help users find the books they are looking for [2]. It shows that providing a better service to the customer also makes the people who make the recommender system both financially and morally satisfied.

Recommender systems are software tools and techniques that provide users with the correct items [3]. It makes life easier for users by providing items they will love. At this point, the item keyword is used generically in this thesis. This item can be news, a book, music, or a product. It represents the interacted things by the users.

Recommender systems are divided into three classical approaches. These are collaborative, content-based, and hybrid models. These approaches generally aim to recommend the correct item from user-item interactions. However, the current tastes of users may change, and these types of approaches evaluate all historical interactions at the same level. Therefore, the disadvantage of these models is that they cannot sufficiently consider the current tastes of the users. In order to solve this issue, session-based and sequential recommendation approaches have emerged. In these approaches, the order of items that users interact with becomes essential. In this way, it is possible to find up-to-date tastes. For example, consider a person in his twenties who likes action movies but has been liking drama movies for the past few years. If we design an engine with traditional recommendation methods, the model tends to recommend action movies because it interacts much more with action movies. However, since the sequential recommendation captures time-varying tastes, models based on it can also

capture current tastes and recommend drama-themed movies to the user.

In this study, we have used three datasets. The first one is the G1 news dataset published by G1 globo [4,5], a news website in Brazil. This dataset contains user-news interactions. In addition to user news interactions, it also includes information such as when it happened, from which country, and from which operating system. The other two datasets are from Movielens [6] published at different times and containing the movie ratings data by users. Our rating data consists of three critical components: movies, users who rate movies, and the value of ratings between 0.5 and 5. For example, we can read a line in this rating data as a user gave a movie 4 points. In addition to the user-item interactions, the datasets also include information about the movies such as their title and genre, as well as the timestamp of the interactions. We utilized this supplementary information to generate features in both datasets.

We will call the rating part of the Movielens dataset and user-news interactions in the G1 news dataset interactions of user-items. We will make models that recommend items to the user based on the user's interaction sequence. For example, we take a sequence of five movies that the user rates, and we will try to find the sixth movie that the user will rate. While doing this, we will make many recommendations. Then, for instance, one of the metrics we use will check if one of them hits it. We experimented with and interpreted how our models perform with their different features.

Our contribution to this thesis is to observe how transformers-based sequential recommender systems work with different features and then interpret them in a meaningful and consistent way. Our first contribution is to show that transformer models work best with short sequences. When we took long sequences, the model also learned that items that had been used in user-item interactions but interest got lost at one point, which is why prolonged sequence usage is causing worse results. Another is that time-based features contribute best to performance among the many features. We attribute both of these to user interactions changing over time. We show that the dataset confirms this change, and time-based features allow capturing tastes that change over

time. We will illustrate these results in detail.

This thesis is structured as follows: section two reviews the literature on recommender systems. In section three, we provide an in-depth examination of the datasets, including a general overview, a detailed analysis, the features generated from the datasets, and the evaluation metrics used. Section four explains the model architecture, data preparation, and training methodologies. The results of our experiments and the corresponding discussion are outlined in section five. Finally, section six includes a summary and our recommendations for future research.

2. RELATED WORK

Recommender systems are tools and techniques that automatically suggest the right products to people. These systems not only simplify users' lives by saving them time but also increase profitability for service providers. This chapter will provide information from past work on how these systems work.

Recommender systems typically have three primary inputs: users, items, and the interactions between them. Users search for various items such as food, movies, music, vacations, and products. They possess distinct personalities and have varying preferences. Items, on the other hand, are the music, movies, food, or products suggested to users, and the type of items varies depending on the domain. These items can be characterized by attributes such as subject, type, color, or location, which make up the features of the items.

Interaction is the third input for recommender systems. It contains the relationship between the user and items. Again, this can be defined in different ways according to the domain. For example, a click, purchase, or reading are forms of user-item interaction. The interactions data type can be divided into two implicit and explicit. The implicit one is the feedback in which the users interact with items in actions such as clicks or purchases. For example, whether the user clicks on a product on an e-commerce website can be considered a binary relation. Explicit feedback is the explicit declaration of the user's likes or dislikes. For example, they were leaving a good review for a movie or a place, giving a good rating. It is a form of feedback that gives information on whether it likes it much more clearly.

The recommender systems should work reasonably because if a user sees an inappropriate item in the recommendation, the user can stop using the app. For this reason, those who designed the system may provide less benefit or even harm with the recommender system. That is why it is essential to do recommender systems well.

We will discuss general approaches to how recommender systems work. After doing this, we can grasp the problem the sequential & session-based recommender systems are solving, which is our focus. For a general summary, we can divide the general approaches into three. These are content-based, collaborative filtering, and hybrid methods.

2.1. General Recommender Systems

2.1.1. Content-based methods

Content-based methods assume that similar items will be rated similarly. In this model, considering the items that users give reasonable rates, the items closest to them in terms of features are recommended. We can explain this with an example. If a user reads sports news, the user may be offered sports news instead of political news because sports news features are similar to sports news rather than political news.

Content-based models give the best results where the features of items are rich such as web pages, publications, and news [7]. Based on the assumption that users prefer similar items more, it is necessary to find similar items better to improve this model. Therefore, finding which features and properties discriminators are in the items will be used to improve performance [1]. Whether an item is similar to another item in terms of its features can be determined through various techniques, such as machine-learning or deep-learning methods.

2.1.2. Collaborative Filtering Methods

Collaborative filtering methods assume people having similar tastes will rate similarly for the same items [1]. Therefore, items liked by similar users are enough to make recommendations that are recommended to a user. Therefore, it is optional to know the items' features in this recommender system because it is enough to find similar users without checking the features of items. It can be defined as people similar to you

who choose these items so that you might like them too.

A simple example of similar users is as follows. Suppose we want to make a recommendation to a user using the collaborative filtering method. First of all, we have an user-item matrix using all the data. Then other users close to a user's vector are found by a method such as a cosine similarity. We can also call them neighbors in the user space. Using similar users, we can suggest items to a user that did not interact with an item but similar users interacted with this item. As seen in this scenario, there is no need to know the features of the items. Similar users were enough to find appropriate items to recommend.

2.1.3. Hybrid Methods

It combines collaborative and content-based filtering methods. The dilemma is that one outperforms the other in some situations. In order to benefit from both, hybrid methods have been produced which would combine the good features of both. So using them together handles the low-performing cases.

2.2. Session Based & Sequential Recommender Engines

Recently, session-based and sequential recommendations have become popular. There are some reasons for this. One of them is that people change their tastes as time goes on. For example, a user used to like books on science fiction, but now the user may like history books. Alternatively, the user may have entered an e-commerce website to look for toothbrushes, and the same user may have logged in to look at trousers in another session. The motivation in these two sessions is very different from each other. Recent tastes should be more dominant than the past. Using all previous history in equal importance may lead to recommendations from the old taste.

The other is that user data may not be logged because of some legal requirements. In this situation, when a user comes to a website or an app, there is no previous in-

formation about him/her. Suppose there are only some click sequences in the given session. The recommender systems should work based on the session data and recommend items to the users.

In this section, we will mention some existing research about recommender systems that try to predict a sequence based on the previous sequence to solve the problems we discussed.

Using sequential data, which consists of previous events of the users, Markov chains are used in recommender systems (RecSys) to predict the following action based on previous action [8]. The transition matrix in MC gives the probability of buying an item considering the last purchases of the user. The problem with using MC is to consider only the last action of the users (or the last few).

If we can think of user-item interactions as sequences over time, the recommendations are the following possible sequence that the users may like, which resembles language modeling. There are many recommender systems implemented inspired by language modeling. For example, there are works in the recommendation systems domain that use word2vec [9] and doc2vec [10] embedding methods. In the well-known model named word2vec, the authors have developed new methods to produce continuous word representation. The meanings of the words are distributed into these representations in this model. Words that are close to each other are also close to each other in space, such as king and queen.

In doc2vec [10], a feature called paragraph vector has been developed, which eventually represents each document as a paragraph vector. Paragraph vector training differs from word vector training by feeding paragraph vectors to the training process. Also, in a Grbovic et al. [11] model, user2vec models are trained such that the items purchased by users are considered a word, and the users are considered a paragraph. While learning the vectors of the products, the vectors of the users were also learned.

Inspired by the word2vec, one model of Grbovic et al. [11] was trained as if the receipts were a sentence and the products written in receipts were a word. With this, representations of products in the same vector space are obtained. They can be used for product-to-product recommendations. The model will ensure that the products that are sold together with each other are close to each other.

In Gru4Rec [12], GRU-based RNN is used for session-based recommendations. The reason to use a gated recurrent unit (GRU) is that GRU solves the vanishing gradient problem in RNN. Because RecSys is not the main area to use RNNs, it is modified for this task considering the live production environment. The network's input is the actual state of the session, and the output is the next event for that session. In short, it tries to predict the next event of the users in a session.

In Moreira et al [13], a deep learning meta-architecture has been developed for news, called CHAMELEON. This model consists of two modules, the first is to learn the represetation of news. The second is to make a session-based recommendation using the Recurrent Neural Network. The problem it wants to solve is to suggest the most appropriate next article according to the interactions in the session of the user.

A sequential model named transformers is good at natural language processing tasks. The self-attention mechanism, the basis of transformers, can reveal syntactic and semantic patterns between words in a sentence [14]. This method has been applied to the sequential recommendation task and developed the model SASRec [14]. While training this model, it tries to generate the following item by using the previous steps at each step. For example, while at the nth step, it predicts the next using the previous n step. Another work is the use of transformers in Sun et al (2019) [15]. Moreover, in the 2022 ACM RecSys challenge [16], Lu, Y et al. [17] uses transformers ranked second in the competition.

In addition, Transformers4rec [18] is an end-to-end framework developed using Python and PyTorch. It encompasses data processing, model training, and evalua-

tion pipelines. It is built upon the HuggingFace Transformers library. The purpose of Transformers4rec is to use Natural Language Processing (NLP) advancements to sequential / session-based RecSys immediately and to bridge the gap between them. As a result, It makes available the latest transformers to RecSys at the production usage level, which is the environment where the end-user uses them.

3. DATASET

In this thesis, we used three datasets. These are called Movielens20, Movielens25m, and G1 news. The Movielens datasets [19] were published by the research lab, GroupLens. Users can vote for the movies they want on the website [6] by giving a score between 0.5-5. Then they can get a personalized movie recommendation based on the points they give. So they can both feed the dataset and get a movie recommendation as output from the recommender system. The other one, G1 news [5], was published by G1 globo, one of Brazil's most popular news websites. It includes user-news interactions within 16 days.

3.1. The Movielens Datasets

We want to provide more transparent information about how the Movielens datasets [19] are created. An essential piece of information contained in this dataset is these ratings. Users have the opportunity to give points to each movie on the Movielens website. They can give more points if they like the movies, and less if they don't. Each voting process is called an interaction in this thesis. These interactions consist of user id, movie id, rating, and timestamp as the date of this event. Additionally, our data includes the ratings of users who have given at least twenty votes.

We have used two Movielens datasets in this thesis. Both have the same structure, with 20m and 25m movie ratings released at different times. The 25m dataset is in the same format but contains slightly more than the 20m dataset. The information we give for datasets will be valid for both.

There is a general summary of our datasets in Table 3.1. As the names suggest, there are twenty million ratings in the Movielens20m whereas twenty-five million in the Movielens25m dataset. The table displays information on time intervals, movie counts, rating counts, and tag counts for the datasets.

Table 3.1. Movielens Dataset Statistics

	Movielens20m	**Movielens25m**
range	March 1995 - October 2016	March 1995 - November 2019
movie	27278	62423
user	138493	162541
rating	20000263	25000095
tag	465564	1093360

Users can give tags as they want in addition to giving the movies a rate. These tags consist of words or phrases they think to describe the movie. For example, one movie got "dark hero", and another got "1940s" as tags. Users can add new tags or vote for existing tags.

In addition to the rating dataset, the Movielens dataset contains information about the tag genome [20]. The process of applying tag genomes to the dataset is as follows. First of all, the tags that were given very few were eliminated. These may be misspelled tags, such as "sadfasda", or things that are very relevant to the user but not related to the movie, for example, the tag "I like it a lot". Such kind of tags does not indicate a property of the movie but rather a personal comment from the user. Also, tags containing actors and directors were eliminated. After that, 1128 tags remained. These tags are seen as valid. A similarity score is calculated between these tags and all movies based on a machine-learning technique. These similarity scores represent how much a tag is relevant to a movie.

We also have the genres of the movies. For example, the movie title is Toy Story (1995), and the genres are adventure, animation, children, comedy, and fantasy. It means a movie can have more than one genre. We also use this information while doing our experiments.

We said that we are using the rating dataset as the interaction dataset. When building a model, we also consider the side information, such as genres and tags. We

combine them around interactions. In the end, the following data has been obtained in the combined dataset: user id, movie id, rating, timestamp, genres, tag, genome relevance, and genome tag. In the next section, we will focus on features after getting to know the dataset.

3.1.1. Detailed Information About the Movielens Datasets

In the previous section, we introduced the Movielens data. This section will give detailed information and look at what the data we are working on looks like. We will give general information to get to know it and emphasize the parts affecting our results. While doing this, we used the Movielens25m dataset. We can consider the same observations to be valid for the other.

First, we can look at the rating counts over the years. The y-axis in Figure 3.2 represents the years, while the x-axis represents the number of ratings for that year. Since we train according to time, we wanted to check if we have enough data every year. As can be seen from the graph, we have enough data every year. Also note that we excluded 1995 from our data, as the dataset only includes three ratings.

Moreover, except for January 1996 and February 1996, we have at least 10k data every month. The reason for stating this is that we do incremental training based on months, as we will mention in the methodology section. It means that the data is enough for a monthly bases to work with it.

Table 3.2. Rating Counts Over Years

1996	1430049	**2002**	777297	**2008**	1018212	**2014**	478553
1997	626234	**2003**	920568	**2009**	810137	**2015**	1604752
1998	272042	**2004**	1048015	**2010**	792414	**2016**	1756826
1999	1058734	**2005**	1613384	**2011**	676434	**2017**	1690039
2000	1735624	**2006**	1038582	**2012**	635225	**2018**	1311014
2001	1057776	**2007**	931481	**2013**	515543	**2019**	1201147

Looking at the statistics based on movies will be helpful in terms of understanding the data. Table 3.3 shows the statistics about how many ratings movies got. As can be seen, although the average is 423.4, the standard deviation of our data is relatively high, 2477.9. In addition to these, the median is 6. In other words, half of the movies have a rating lower than 6. It shows that the rating counts are not balanced among the movies. Some popular and relatively old movies have received high ratings. For example, a movie named Forrest Gump received a maximum rating of 81,491. Movies with much interaction made the rating counts distribution unbalanced. In Table 3.4, we see the 15 movies with the most interaction.

Table 3.3. Movies Interaction Statistics

avg	423.4
stdev	2477.9
min	1
percentile 25	2
median	6
percentile 75	37
max	81491

In order to get a better understanding of user behavior on rating movies, we can look at some statistics about the rating scores. We have listed the movies with the best score that received at least 100 interactions in Table 3.5. On average, movies got 3.53 points. However, this, of course, varies according to the movies. Planet Earth movies were well-liked by the audience.

As we will explain in more detail in the methodology section, we train and evaluate the model in an incremental method. Therefore, the distribution of the release dates of the movies is essential. In the dataset, the titles of most movies have the year they were released. We extracted this information from there. Table 3.6 contains information about how many movies were released in which years and the percentage of the total number of movies. As can be seen from here, the Movies are not concentrated in specific years and are spread out evenly over all years.

Table 3.4. Most Interacted 15 Movies

Forrest Gump (1994)	81491
The Shawshank Redemption (1994)	81482
Pulp Fiction (1994)	79672
The Silence of the Lambs (1991)	74127
The Matrix (1999)	72674
Star Wars: Episode IV - A New Hope (1977)	68717
Jurassic Park (1993)	64144
Schindler's List (1993)	60411
Braveheart (1995)	59184
Fight Club (1999)	58773
Terminator 2: Judgment Day (1991)	57379
Star Wars: Episode V - The Empire Strikes Back (1980)	57361
Toy Story (1995)	57309
The Lord of the Rings: The Fellowship of the Ring (2001)	55736
The Usual Suspects (1995)	55366

Table 3.5. Top Rated 15 Movies

Planet Earth II (2016)	4,483
Planet Earth (2006)	4,464
Shawshank Redemption, The (1994)	4,413
Band of Brothers (2001)	4,398
Cosmos	4,326
Godfather, The (1972)	4,324
Blue Planet II (2017)	4,289
Usual Suspects, The (1995)	4,284
Twin Peaks (1989)	4,267
The Godfather: Part II (1974)	4,261
Over the Garden Wall (2013)	4,258
Black Mirror	4,256
Seven Samurai (Shichinin no samurai) (1954)	4,254
The Adventures of Sherlock Holmes and Doctor Wats	4,251
The Blue Planet (2001)	4,248

Table 3.6. Movies Release Dates Distributions

Year Range	**Movie Count**	**% of Total**
1870-1964	1529	%16.3
1965-1969	2049	%3.3
1970-1974	2601	%4.2
1975-1979	2347	%3.8
1980-1984	2319	%3.7
1985–1989	2851	%4.6
1990-1994	2836	%4.6
1995-1999	3805	%6.1
2000-2004	5118	%8.3
2005-2009	7530	%12.1
2010-2014	10020	%16.2
2015-2019	10403	%16.8

3.1.2. Fashion Effects on Ratings

We said that the advantage of sequence recommendations over classic recommendations is that they recognize and learn from changing tastes. We have given an example of books before. People's tastes in book genres may vary over the years, and the model may not be able to learn about current book tastes if the model cares equally about the entire past. On the other hand, sequential recommender systems catch up with current tastes. However, such a change of taste must also be present in the data for the model to learn this. Considering this, we want to see on the data whether the movies received more interaction in specific periods, that is, whether there has been a fashion over the years about interacted movies.

We have developed a method to check whether the dataset has a fashion over the years. Intuitively, movies usually get more interaction in the first period they are released. If interactions concentrate on certain movies in specific periods, i.e., fashion, this concentration may be the first period of their publication. We want to check this intuitive assumption by choosing two different years to test this hypothesis. These are

1997 and 2002. We compare the interaction counts of the movies released in these years according to the ages of those movies. We want to see if they get more interactions in their first year.

Table 3.7. Interaction Distrubition over Age of Movies

1997			**2002**		
Age	**% of cumulative**	**% of total**	**Age**	**% of cumulative**	**% of total**
0	%2.6	%2.6	0	%4	%4
1	%6.9	%4.3	1	%14.3	%10.3
2	%16	%9.1	2	%21.9	%7.6
3	%28.1	%12.1	3	%32.4	%10.3
4	%34.4	%6.3	4	%38.8	%6.4
5	%38.5	%4.1	5	%44.2	%5.4
6	%43	%4.5	6	%49.6	%5.4

Table 3.7 shows our analysis's results according to the hypothesis. In the % part of the "total" column, there is the interaction ratio of the movie at the relevant age to the total number of interactions. In the % part of the "cumulative" column, there is the ratio of the cumulative sum according to age to the total number of interactions the movie takes. This table shows us that movies take about one-third of their total interactions in the first three years, and almost half are in the first six years. This finding confirms the hypothesis that movies interact more in the early periods. From this point of view, we can say that in every period, the newly released movies of that period have more interaction. Therefore, we can claim that fashion affects the Movielens dataset.

3.1.3. Feature Definitions

While introducing the dataset, we said that the dataset we finally received contains the following information: userid, movieid, rating, timestamp, genres, tag, genome relevance, and genome tag. Of these, movie id is defined as the item in user-item interactions of recommender systems. We will also use the user id to divide into sequences.

We will explain the details of it in the methodology section. We get many features from other information such as the genre of the movie. We divide the features into three groups according to their types. These are categorical, continuous, and time-based features.

A movie can have more than one genre, which means a movie can become about comedy and drama. We had to choose one of the genres due to the limitation of our framework, Transformers4rec. So, we have chosen the first one from the genre list of movies. We use this genre as a categorical feature.

In the previous section, we talked about tags and genome tags. We had to choose one tag and one genome tag for a user-item interaction for the same reason as the genre. While making this selection, we took the most given tags. We got the highest relevance score for the movie among the genome tags. These became our other two categorical variables. We have three categorical features, tags, genome tags, and genres.

We have two continuous features. One of them is the rating score that a user gives a movie. The other one is genome relevance which is the relevance score of the genome tag.

We know the timestamp that represents when users give these ratings. Time information is essential because we have generated many features using this timestamp. We call this type of feature a time-based features.

The first time-based feature is the day of the date. It says on which day of the month the rating is given. It ranges from 1 to 31. Another one is the week of the year as a feature. It ranges from 1 to 52, and the second feature is the year, which starts from 1995 and goes up to the end of the dataset used. We similarly used weekday, the day of the week, and defined it as a feature that changes from 1 to 7.

Moreover, the day is a cyclical feature, which means that the first day of a month

is near the last day of the previous month's event. With this feature engineering, the day is mapped as a feature to a float ranging from -1 to 1. If they are only labeled 1 and 31, the numbers are far from each other. So, to get information on cyclicality, we have defined day sin features [21].

Finally, for Movielens datasets, we got the following features: day, weekday, year, and day sin in time-based features; genres, genome tags, and tags in categorical features; genome relevancy and rating in continuous features.

3.2. G1 news Dataset

G1 globo , a popular website in Brazil, published the G1 news dataset [5]. This website is so popular that there are more than 80m unique users per month, and more than 100k new content is uploaded monthly [13].

This dataset has user-news interactions from October 1 to October 16, 2017. More than 3 million interactions were made in 1.2 million sessions in this period. 330k users made these interactions, and there are 50k news articles [13]. This was the overall picture of the dataset. We use the preprocessed version of this dataset [22] shared on Transformers4rec [18]. In the preprocessing process, sessions having only one interaction were eliminated, and longer user-news sequences than 20 were cast to 20.

In this dataset, we have been provided with session-level information. Specifically, in addition to providing information about which users clicked on which news articles, it also includes information about the session to which these interactions belong.

Also, it would be beneficial to present statistics about the number of news clicks within each session for further analysis in this study. Table 3.8 illustrates that the G1 news dataset consists of short sequences. Even at Percentile 75, it is 3, and the standard deviation of the session lengths is relatively low.

Table 3.8. Session Length Statistics of G1 News

avg	2.7
stdev	1.3
min	2
percentile 25	2
median	2
percentile 75	3
max	13

We used this dataset in our experiments in two ways. The first takes 16 days, and the second uses the first 2 days. We will describe the results for both of them in experiments and results section.

3.2.1. Feature Definitions

Items in user-item interactions in this dataset are articles that users click. The goal is to recommend articles to users based on the sequence of articles that they have clicked on.

Similar to the MovieLens dataset, a wide range of additional information is available about the interactions, including categorical variables such as environment, device group, operating system, country, region, and referrer type. The platform [23] in which the dataset was released [5] does not provide specific definitions for these fields, but their meaning can be inferred from their names.

Additionally, time-based features have been generated for the dataset using the timestamp of the beginning of a session. The cyclicality effect has been incorporated using the method [21] used in the MovieLens dataset. Hour in, hour cos, weekday sin, and weekday cos have been created from the timestamp of the beginning of a session and the cyclical feature generation method.

In addition, the moment news receives interaction for the first time, we accept its birth date and calculate the age of that article while other interactions are taking place. We take the normalization of these and use it as a feature called item age hours norm.

Finally, for the G1 news dataset, we got the following features: hour sin, hour cos, weekday sin, weekday cos, item age hours norm in the time-based features; click environment, click device group, click os, click country, click region, click referrer type in categorical features.

3.3. Evaluation Metrics

Recommender systems help users find what they are looka¡ing for or discover new products through software. Monitoring the recommender system's performance is crucial to ensure that the software works effectively. In this thesis, we use several evaluation metrics to monitor the performance of the models we use.

In this thesis, we aim to predict the next item in a session based on the previous items. We treat the last item in the sequence as if it were the next item and attempt to predict it using the previous items as a reference. Consider a sequence of seven movies in the Movielens dataset. We aim to predict the last movie in the sequence by using the previous six movies as a reference.

We recommend the top K items out of all possible items and then evaluate whether the target item is included in our top K recommendations. Since we are only predicting the last item, it serves as our singular, definitive target. If we only consider whether the target is included in the top K recommendations, this measurement is called recall@k. For example, if we successfully predict the target item 20 times out of 100 attempts, our recall rate would be 0.20 (or 20%).

This thesis also uses the Normalized Discounted Cumulative Gain (NDCG) method

as a metric. This method considers the order of the target item among the top K recommendations.

To understand how NDCG considers orders in items, first, it is crucial to define discounted cumulative gain (DCG). DCG is a metric that rewards when the target is on the first lines; in other words, it punishes when the target is behind in the lines [24]. It is calculated by the formula

$$\mathbf{DCG} = \sum_{i=1}^{K} \frac{gain_i}{\log_b i} \tag{1}$$

where the term "gain" represents the rewards of hitting the recommended item, and the variable "b" can be chosen between 2 and 10 [25] in the equation. As seen, it is punished when the model hits on the higher i values by a logarithmic function, which ensures smooth punishing. Finally, we can define NDCG as DCG normalized between 0 and 1. It is a score between 0 and 1, similar to recall, and the closer the score is to 1, the better the metric performance.

In the experiments and results section of this study, we will utilize suffixes of "K" as "@10" and "@20" in conjunction with the metrics NDCG@10, NDCG@20, recall@10, and recall@20, which implies that we will be using "10" and "20" as our K values.

4. METHODOLOGY

The HuggingFace (HF) Transformers library [26] was built to open up developments in NLP to the broader machine-learning audience. It has an open-source repository and many contributors. This library contains many machine-learning models, especially transformers. In this thesis, we used Transformers4rec [18], an open-source library extended from the HF Transformers library. This library allows the implementation of sequential and session-based models at the production level. Transformers4rec takes transformers architecture from HF Transformers and writes a specialized head for the recommendation problems. Transformers4rec library has been used in two recommendation competitions, SIGIR eCommerce Workshop Data Challenge 2021, organized by Coveo [27] and WSDM WebTour Workshop Challenge 2021, organized by Booking.com [28] and got the best grades in both.

Sequential and session-based user interactions are similar to word sequences in language modeling. However, unlike language models, sequential and session-based recommendation models can receive side information. For example, we use genres of movies, time of interaction, and many other features. Ranking metrics are also different from language modeling. They require incremental training and evaluation. Sequence types in RecSys are much more time sensitive. Language changes over time, but it takes several years. Because of these differences, Transformers4rec has also modified the transformers library to make it suitable for RecSys.

4.1. Problem Definition

We can describe the recommendation problem we want to solve using "Rate More" section on the Movielens website [6]. Movielens website have a section similar to Figure 4.1. In this section, new movie suggestions have been made for users to rate based on movies that have been interacted with before. The task we want to solve here is similar to it. We have implemented this model to fine-tune it from month to

month for Movielens datasets, and we recommend the movies that are most suitable for users by looking at the past interactions among many movies. When it shows movies that may be closer to the user instead of random movies, the user can easily find the movies they want to rate. In this way, Users are provided to spend more productive time on the website. Similarly, using the G1 news dataset, a sequence of news clicks is used to suggest new news articles for the user to continue reading. In summary, we are attempting to create a recommendation sequence based on the input sequence of interactions.

Figure 4.1. Rate more

4.2. Transformers4rec Architecture

We provide information about the model's architecture shown in Figure 4.2. Transformers4rec's architecture consists of multiple modules. First, input features such as categorical and continuous prepared for models with Extract Transform Load Module (ETL) are given to the feature processing module. We use a library called NVTabular [29] to generate data suitable for sequential recommender systems from tabular data. After the input features are normalized, they are aggregated, then interaction embeddings are created. These interaction embeddings are given to the sequence masking module. Sequence masking module gives the embeddings to the sequence processing module after masking them according to the training method (e.g., Causal LM, Masked LM). The sequence processing module has transformer blocks (e.g., GPT-2, Transformer-XL, XLNet, Electra). Sequence embedding is achieved using these blocks. These transformer blocks feed the output layer, and the relevancy score is obtained for all items using the softmax method.

In summary, it is a library that allows us to try transformers models on recom-

mender systems. It provides it with an easy way to implement. It also has configurable inputs, and its output formats have a readily observable quality.

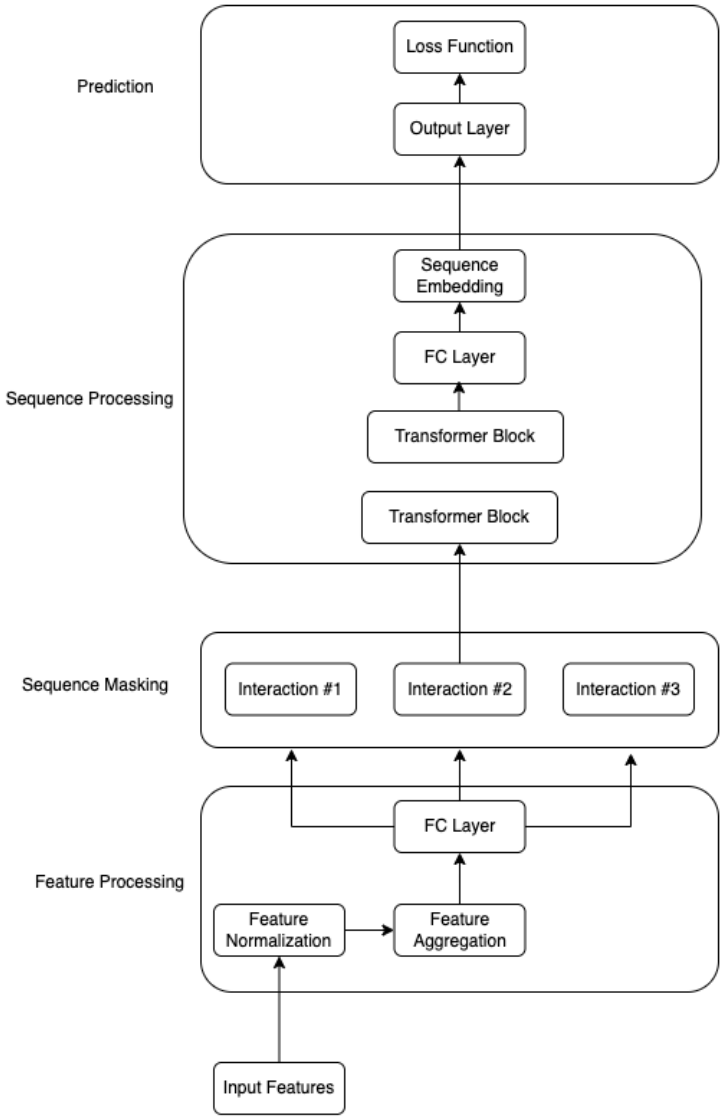

Figure 4.2. Transformers4rec architecture

4.3. Extract Transform Load

The Trasformers4rec framework requires data to be in a specific format, and this requirement necessitates the use of an Extract, Transform, Load (ETL) process.

In this section, we will explain the process of dividing the datasets used in this thesis into sequences and preparing them for the recommendation model.

In this thesis, we call the movie rating data, and the user clicks on the news as "interactions". Each row in this tabular dataset includes the user id, item id, timestamp, and information about the interaction. Each row gives information about when the user interacted with which item. In addition, we brought information about the items, such as the movie's subject, in Movielens datasets and clicked country in G1 news, which is also available in the datasets. We need to adapt this raw data to the recommender systems. While doing this, we use the NVTablular [29] library. Its task is to prepare the preprocessed data required by the training pipeline.

The raw data with a few columns is similar to Table 4.1 in Movielens datasets. One row contains the movie that a user interacts with and its features. We put one continuous feature as an example. In the experiments using Movielens datasets, we created sequences of user interactions to provide the data to the model on a per-user basis. In other words, we grouped them according to the users and extract lists of data from the other columns.

Table 4.1. Movielens Raw Data

User Id	**Item Id**	**A Feature**	**Timestamp**
user_0	movie_0	1.5	1120105552000000000
user_0	movie_1	4.5	1183626018000000000
user_1	movie_2	3.0	1214626018000000000
user_1	movie_3	5.0	1259325028000000000
user_1	movie_4	1.5	1331925008000000000
user_2	movie_5	0.5	1528213801000000000

We grouped the data in Table 4.1 by user id and converted it to the format in Table 4.2. There are different users in each row in the grouped one. Columns other than user id include sequences. The length of these sequences is equal to the number of items that users interact with. For us, a sequence means an item set with which a user

interacts. For example, if a user has interacted with 100 items, it results in 100 items in one cell of the sequence columns for that user. Other features we mentioned in the dataset, not in this table, have been added here as sequences, such as genre sequence and month of timestamp. Table 4.2 shows the shape of the data format we gave to the model for the Movielens dataset.

Table 4.2. Movielens Preprocessed Data

User Id	**Item Id Sequence**	**A Feature Sequence**	**Timestamp Sequence**
user_0	movie_0, movie_1	1.5, 4.5	1120..., 1183...
user_1	movie_2, movie_3, movie_4	3.0, 5.0, 1.5	1214..., 1259..., 1331...
user_2	movie_5	0.5	1528...

As previously mentioned, we create our sequences on a user basis in the Movielens dataset. Our task will be to predict the next movie interaction of users based on their previous movie interactions. In contrast, with the G1 news dataset, we work with sessions, which is slightly different from the approach used in the Movielens data. The raw version of G1 news data, as seen in Table 4.3, contains a log of each click event of users. For example, in session 0, article 0 was clicked, and an example feature of this interaction is 1. Additionally, this click event has a timestamp representing when the event occurred.

Table 4.3. G1 News Raw Data

Session Id	**Item Id**	**A Feature**	**Timestamp**
session_0	article_0	1	1219105440000000000
session_1	article_1	2	1232944001000000000
session_1	article_2	5	1113315018000000000
session_1	article_3	2	1115119018000000000
session_2	article_4	1	1212793008000000000
session_2	article_5	5	1819453902000000000

Using this raw data, we create sequences for the recommendation model similar to how we did it with the Movielens dataset. The main difference is that while we

created user-based sequences in Movielens, we created session-based sequences here. By grouping this data similarly to how we did it with Movielens, we obtain Table 4.4. We will give this preprocessed data to the recommendation models and recommend articles for each session that may be suitable for that session.

Table 4.4. G1 News Preprocessed Data

Session Id	**Item Id Sequence**	**A Feature Sequence**	**Timestamp Sequence**
session_0	article_0	1	1219...
session_1	article_1, article_2, article_3	2, 5, 2	1232...,1113...,1115...
session_2	article_4, article_5	1, 5	1212..., 1819...

4.4. Incremental Training and Evaluation

In the preceding section, we examined how we generate sequences from sessions in the G1 news dataset and user interactions in the Movielens dataset. It would be beneficial to clarify what we are training and what we aim to predict. Our objective is to identify patterns within item sequences. Specifically, we aim to uncover the pattern of news sequences within a session in the G1 news dataset, and the pattern of movie sequences among the movies that users have rated in the Movielens dataset.

As the model learns these patterns, it attempts to predict the last item in the sequence. For example, given a sequence of five items (item_1, item_2, item_3, item_4, and item_5), the model attempts to predict the fifth item by analyzing the first four items. In the case of the Movielens dataset, the model attempts to predict the nth movie a user will interact with by using the information from the previous n-1 movies. Similarly, for the G1 news dataset, the model attempts to predict the last news article by learning the pattern of news sequences.

While identifying patterns in the item sequences, we provide the Transformers4rec framework with additional features related to the interacted items, referred to as feature sequences. These sequences are not necessary for the models to learn, but they often

improve the results. Similar to the item sequences provided, feature sequences are added to the framework similarly. Moreover, we will delve into the feature importance topic in the experiments and results section.

Previously, we mentioned that sequential models capture changes in people's behavior over time. In daily life, recommendation models that work in production are updated daily, hourly or monthly according to their domain. For example, if a website offers up-to-date news with much traffic, it is reasonable to update it hourly. Or, if an e-commerce application sells only a few particular products, it is appropriate to update the model daily. In this section, we will talk about how we update the model over time and how we evaluate it.

Suppose there is a recommendation model that is fine-tuned daily. In this case, also assume that we use the data of n-1 days in the past to train the model and test this model in production with end-users for today. If we say it mathematically, suppose the current day is the T_n'th day. We obtained the model we use today using user-item interactions on days $T_1, T_2...T_{n-1}$. We also did the test on the current day, T_n. When we fine-tune the model again for the next day, we will have trained the model in the sequence $T_1, T_2...T_n$. When we test it, we will have tested it on the day T_{n+1}.

At this point, we have to choose a time window while training the datasets by time. We need to decide whether we should train and evaluate it day by day or weekly. We chose the windows month-to-month for Movielens because when we look at the data from month to month, we observe that we have enough data for each month, and there is a pattern that can be learned from month to month. Therefore, we implemented the model to be fine-tuned for each month. Our model starts to train as of the first month of data. It is fine-tuned over and over again for each subsequent month.

As in the example about daily fine-tuning and evaluating the next day, in these experiments, the model is evaluated with the test data of the i+1 month when it is fine-tuned by using i'th month, and it is evaluated with the data of the i+2 month

when it is retrained for the i+1 month. For example, after being fine-tuned in April 2001, the model was evaluated in May 2001. In this way, we get an evaluation result for each month.

On the other hand, news is inherently more dynamic than movies. In particular, the news presented on a website or application is highly current and up-to-date. We used an hourly time window to effectively capture this dynamic nature in our G1 news dataset, which includes news interactions. This approach enabled us to fine-tune and evaluate the recommendation models for each hour. For example, when the model is fine-tuned for the 6th hour of the day, it is evaluated with the 7th hour, and when it is fine-tuned with the 7th, it is evaluated with the 8th hour. We call this type of task incremental training and evaluation.

The overall success of the model is obtained by taking the average of the evaluation results of time windows. In other words, the results of each time window are recorded, and then the average of the results of all time windows is taken. This averaging method is employed because it closely mirrors how models are evaluated in production environments. For instance, the success of a model at the production level consists of the period of interest. Today's success is vital if we are to evaluate the model daily. When we look at the model's success tomorrow, we will consider only tomorrow again. Looking at the overall past, we take the average of all periods to see how successful it was. We can closely mimic how models are evaluated in real-world scenarios by designing such an evaluation process.

5. EXPERIMENTS AND RESULTS

In this thesis, we utilized three distinct datasets. The first is from G1 globo [5], a news website in Brazil. The other two datasets are from Movielens [6]. We name them Movielens20m and Movielens25m. These Movielens datasets contain similar information but were published at different times and contain approximately 20m and 25m interactions, respectively. We conducted our experiments on these three datasets separately. In this section, we will present the results of our experiments.

As the methodology section explains, we use user-item interactions as a sequence in this thesis. Interaction refers to a rating given to a movie for the Movielens datasets or a click on a news article for the G1 news dataset. Therefore, a sequence size is the number of movies the users have interacted with so far for Movielens datasets and the number of interacted news for the G1 news dataset in a session.

In our experiments, the initial parameter we have to select was the length of the sequences for each user. Our first inquiry is to determine the significance of sequence length. To make this decision, we conduct experiments using the Movielens dataset. However, for the G1 news dataset, we use fixed values of a minimum of 2 sessions and a maximum of 20. As stated in the dataset section, the sessions in the G1 news dataset are relatively brief, thus we deem it unnecessary to include this dataset in our investigation of sequence length.

For the Movielens data, we chose the length by experimenting. The Movielens data was published with a minimum sequence of 20. We also have some users with approximately 10k sequences, meaning some users rated approximately 10k movies. Due to the framework we use, it is not possible to use all interactions of users with long sequences. We had to choose some of the latest interactions instead of all. For example, in one experiment, we used the last ten movies. We trained the recommendation models using the ten latest sequences. In other words, we predict the next film interaction

based on the ten latest interactions of users in the Movielens dataset.

5.1. Length

In order to determine the importance of the sequence length of the features for the Movielens datasets, we conducted experiments using various lengths. During these experiments, we utilized all the features available to us. As the model, we used XLNet for all of them. In all experiments, everything was the same, and only the sequence length we used was different. To answer the question of how many latest interactions we should obtain, we trained the model with different sequence lengths of 3, 5, 10, 25, 50, and 100.

Table 5.1. Length Results - Movielens20m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
3	0.207	0.230	0.317	0.409
5	**0.227**	**0.250**	**0.348**	**0.437**
10	0.207	0.231	0.318	0.410
25	0.207	0.230	0.318	0.410
50	0.207	0.231	0.317	0.409
100	0.208	0.230	0.328	0.417

Table 5.2. Length Results - Movielens25m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
3	0.157	0.182	0.279	0.380
5	**0.197**	**0.224**	**0.334**	**0.444**
10	0.182	0.206	0.303	0.399
25	0.179	0.201	0.297	0.383
50	0.158	0.184	0.290	0.393
100	0.144	0.168	0.248	0.344

Tables 5.1 and 5.2 indicate that the best results were obtained from sequences with a length of 5, possibly because transformers perform well on short sequences.

The intuitive interpretation of the model results is that we predict the next movie based on the previous four movies. In other words, the model predicts the next movie well using the previous four movies instead of long sequences. We used length 5 in other experiments with Movielens datasets based on these results.

5.2. Different Model Results

In the next phase, we tried different models: Albert, Electra, and XLNet. After finding the model that gives the best result according to the experiment results, we have chosen the best. In these experiments, we have used all features we have, just as we did when choosing the length.

Table 5.3. Different Model Results - Movielens20m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
Albert	0.143	0.160	0.246	0.315
Electra	**0.229**	**0.252**	**0.353**	**0.443**
XLNet	0.228	0.252	0.347	0.439

Table 5.4. Different Model Results - Movielens25m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
Albert	0.137	0.160	0.240	0.332
Electra	**0.195**	**0.223**	**0.332**	**0.442**
XLNet	0.192	0.219	0.329	0.438

Table 5.3 and Table 5.4 show that the Electra model gives the best results in all metrics in both Movielens datasets. Therefore, we used the Electra model in our subsequent experiments with Movielens datasets.

Tables 5.5 and 5.6 demonstrate that the XLNet model outperforms other models when used for 2 and 16 days on the G1 news dataset. Thus, using XLNet for future experiments on this dataset would be good idea for further experiments.

Table 5.5. Different Model Results - G1 News - 2 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
Albert	0.138	0.162	0.264	0.356
Electra	0.090	0.105	0.158	0.216
XLNet	**0.201**	**0.229**	**0.356**	**0.468**

Table 5.6. Different Model Results - G1 News - 16 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
Albert	0.041	0.048	0.082	0.111
Electra	0.039	0.047	0.075	0.108
XLNet	**0.095**	**0.113**	**0.176**	**0.247**

Up to this point, we have decided on the size of the experiment and which models we will try. We said length is the fix for G1 news with a minimum of 2 and a maximum of 20. We said that for Movielens, the sequence length of 5 works well. Moreover, the Electra model will be used for the Movielens dataset, and XLNet will be used for the G1 news dataset.

5.3. Feature Grouping Experiment Results

We have chosen the model and length we will use. We used all features while doing it. In this part, we investigate the importance of the features.

In our experiments, we utilize three groups of features for the Movielens dataset: categorical, continuous, and time-based. For the G1 news dataset, we have two groups of features: categorical and time-based.

We conducted a group-based feature analysis. We used features in groups and observed their performance. When we selected a specific group of features, we used all the features within that group. For instance, if we decide to include the continuous

feature group for the Movielens dataset, we utilize all the continuous features, such as ratings and genome scores.

Table 5.7. Feature Grouping Experiment Results - Movielens20m

	ndcg @10	**ndcg @20**	**recall @10**	**recall @20**
-	0.138	0.164	0.246	0.349
Cat	0.134	0.160	0.246	0.350
Cont	0.201	0.222	0.307	0.391
Time	0.230	**0.254**	0.349	0.442
Cat-Cont	0.205	0.228	0.313	0.402
Cat-Time	0.230	**0.254**	0.350	0.441
Cont-Time	0.228	0.252	0.347	0.439
All	**0.232**	**0.254**	**0.356**	**0.443**

Table 5.8. Feature Grouping Experiment Results - Movielens25m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
-	0.177	0.202	0.296	0.392
Cat	0.175	0.201	0.289	0.391
Cont	0.173	0.199	0.290	0.395
Time	0.195	0.221	0.332	0.437
Cat-Cont	0.172	0.198	0.289	0.393
Cat-Time	0.195	0.221	0.332	0.437
Cont-Time	**0.198**	**0.225**	0.336	**0.445**
All	0.197	0.224	**0.337**	0.443

The results are in Table 5.7 and Table 5.8 for the Movielens datasets. The naming convention in this table is that the cat-cont model only includes categorical and continuous features. Also, as we mentioned, if an experiment includes a feature group, it means that all features in that feature group have been used. Considering that we have two continuous and three categorical features, the cat-cont model has five features.

We experiment with all possible combinations of using or not using feature groups

for the Movielens dataset. Not using any feature at all is an option. The model that does not use any features does not mean that it does not receive any input. It shows that the model only learns depending on the item sequences while being trained. When we use features, we provide additional information while learning the item sequence.

Furthermore, using each feature group individually creates three options, using two feature groups creates three options (cat-cont, cat-time based, cont-time based), and using all features together creates an option. In this experiment, there are 8 cases for the feature grouping experiment. In Table 5.7 and 5.8, we have indicated the status where no feature is used with a "-" sign.

When we look at the Movielens20m datasets in Table 5.7, we can see that the model using all features is the best. For Movielens25m in Table 5.8, the result of the model with cont-time features is slightly better than the model having all features on recall@20.

Additionally, our experiment results have revealed that time-based features consistently achieve the highest performance among the experiments having only one feature group when used alone. For example, when evaluating the recall@20 metric for Movielens25m, we observed a result of 0.437 when only time-based features were utilized, compared to 0.391 when only categorical features were used and 0.395 when only continuous features were used.

Our detailed analysis results of the Movielens dataset revealed that movies tend to receive more engagement in their early years of release. For instance, movies released in 2011 interacted significantly between 2012 and 2013. This suggests that as the model is trained every month, it learns about a specific period and then forgets the past in its memory. The model finds time-based information valuable as consecutive years tend to be similar, such as the years 2011, 2012, and 2013, as well as 2015, 2016, and 2017. We came to the same conclusion by analysis of the Movielens dataset and evaluating the results of our experiments.

Table 5.9. Feature Grouping Experiment Results - G1 News - 2 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
-	0.190	0.215	0.354	0.454
Cat	0.192	0.215	0.361	0.452
Time	0.191	0.217	0.354	0.455
All	**0.203**	**0.229**	**0.374**	**0.474**

Table 5.10. Feature Grouping Experiment Results - G1 News - 16 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
-	0.079	0.097	0.142	0.211
Cat	0.078	0.095	0.144	0.214
Time	**0.096**	**0.114**	**0.182**	**0.256**
All	0.087	0.105	0.164	0.234

Table 5.9 and Table 5.10 present the results of the experiments conducted on the G1 news dataset. Our experiments involving a 16-day timeframe reveal that time-based features are compelling, similar to the findings in the Movielens dataset. This suggests that time-based features perform well when working with a long-term dataset that encompasses different periods. However, when we consider a shorter timeframe of 2 days, we observe that the best results are achieved when all features are utilized together. Additionally, the categorical group features hurt the long-term experiments' results. However, when used in conjunction with time-based features in short-term experiments, they positively impact the results.

5.4. Impacts of Each Feature

In addition to evaluating the features as groups, we also examined which individual features were the most effective. For example, in the Movielens dataset, we have nine features: four time-based, two continuous, and three categorical. We conducted nine experiments to determine the essential feature, testing each feature individually.

Table 5.11. Impacts of Each Feature - Movielens20m

	ndcg @10	**ndcg @20**	**recall @10**	**recall @20**
day	0.202	0.223	0.315	0.398
sin	0.204	0.225	0.308	0.393
week	0.195	0.215	0.298	0.378
year	**0.229**	**0.252**	**0.349**	**0.439**
genome relevance	0.192	0.212	0.288	0.370
rating	0.195	0.218	0.305	0.396
genome	0.140	0.166	0.248	0.351
genre	0.182	0.206	0.329	0.424
tag	0.150	0.177	0.246	0.353

Table 5.12. Impacts of Each Feature - Movielens25m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
day	0.164	0.189	0.297	0.399
sin	0.173	0.196	0.295	0.385
week	0.171	0.201	0.284	0.404
year	**0.192**	**0.218**	**0.329**	**0.430**
genome relevance	0.140	0.165	0.231	0.329
rating	0.158	0.185	0.254	0.362
genome	0.178	0.203	0.296	0.395
genre	0.162	0.189	0.296	0.402
tag	0.174	0.200	0.286	0.388

Table 5.11 and Table 5.12 indicate that time-based features are the most effective across both datasets. The "year" feature is the most impactful among these time-based features, which suggests that the model learns better based on the years of the interactions and that the user-item sequences exhibit similarities within given years. As previously discussed, our dataset analysis revealed that movies tend to receive more engagement in their early years of their release. For example, interactions in 2013 primarily consisted of movies released a few years ago. It creates a pattern in the data over the periods. The results of our experiment align with this, showing that the model could learn the structure of the data by giving more importance to the year feature. The ability to capture this temporally changing pattern of users is a critical factor in the success of sequence-based models.

Another feature that stands out as being important in Table 5.11 and Table 5.12 is the "genre" feature. This feature significantly impacts the models' success, which is not surprising as people tend to interact with movies that belong to genres they enjoy. It can be intuitively understood because people have particular preferences for movie genres. For example, a person who likes action movies is likely to rate action movies highly. Because of the similarity in the type of movie sequences, a similarity is obtained between the sequences in terms of genre. The models are learning these sequences.

Additionally, we made two experiments, 2 days and 16 days, in the G1 news dataset, with each feature alone, and added the results to Table 5.13 and Table 5.14. The results in Table 5.13, based on a 2-day data model, indicate that hour-based features perform the best when used alone. It is likely due to the rapid changes in news on an hourly basis, which is reflected in the effectiveness of hour-based features in the model.

Table 5.14 shows the results of a 16-day model, and the impact of features differs in this case. The table illustrates that when trained over a longer term of 16 days, the results tend to converge, and the age hour norm feature proves to be more effective

when used alone compared to other features. This can be attributed to the fact that as news ages, it loses its relevance and importance. Therefore, it makes sense for the model to consider this feature necessary.

Table 5.13. Impacts of Each Feature - G1 News - 2 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
country	0.197	0.224	0.352	0.458
device group	0.198	0.226	0.352	0.464
environment	0.185	0.214	0.337	0.449
os	0.197	0.224	0.352	0.459
referrer type	0.199	0.225	0.357	0.461
region	0.195	0.224	0.350	0.466
age hour norm	0.192	0.217	0.356	0.453
hour sin&cos	**0.202**	**0.228**	**0.362**	0.463
week sin&cos	0.199	0.225	**0.362**	**0.467**

Table 5.14. Impacts of Each Feature - G1 News - 16 Days

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
country	0.088	0.103	0.168	0.230
device group	0.085	0.101	0.166	0.231
environment	0.086	0.103	0.164	0.234
os	0.089	0.105	0.171	0.234
referrer type	0.091	0.109	0.176	0.248
region	0.084	0.099	0.165	0.224
age hour norm	**0.105**	**0.122**	**0.191**	**0.259**
hour sin&cos	0.094	0.110	0.174	0.241
week sin&cos	0.098	0.116	0.184	0.254

5.5. General Interpretation

First, we started by choosing the size of the interactions, then determined the model, and then showed which features were more effective. At the end of these, we observed that the model learned well in short sequences, and the most excellent point

was the importance of the time-based features that affect the sequential pattern.

For the Movielens dataset, time features work well because movie ratings change over time. Movies get more interaction in the early years after their release. For example, the movie Corruptor, released in 1999, had the most interaction between 1999-2002. After 2002, the movie received little interaction. In addition, when the model uses short sequences while training, it forgets what was learned at that time in the following years. It starts to learn the new pattern for the following years. So it is no coincidence that the model learns well in short sequences and time-based features. Although the movie ratings change over time, users are slightly more conservative on movie subjects. Because we observed that the genre feature is effective in our model results, there must be a pattern in the genre among sequences, and the model tends to make decisions based on its genre feature.

The results of the G1 news dataset show that when the model is trained with a short interval, it performs better. Specifically, the recall@20 score is 0.468 when trained for 2 days but drops to 0.247 when trained for 16 days. This suggests that the model may not forget old data when it is trained for a more extended period of time. Additionally, time-based features, such as those found in the Movielens dataset, were influential in both 2-day and 16-day training intervals. This supports the idea that time-based features are practical and that users' preferences change over time. Furthermore, while hour-based features perform well in the short term, the age feature appears to be more beneficial for achieving good performance in the long term.

5.6. Baseline Methods

First, we will present the results of the baseline models we implemented to compare against our Transformers4rec-based models.

In this thesis, we were inspired by the work of Grbovic et al. in NLP, and used word2vec [9] as a model similar to the study in [11]. We attempted to learn vector

representations of products using word2vec in our baseline model.

We used the following approach: we trained a word2vec model using all items except the last one. Next, we determined the most similar item vectors for the next item by calculating the weighted average of these remaining item vectors. These were the predictions for the last item. Afterward, we evaluated the performance of our model using various metrics. Additionally, we trained our model for Movielens datasets using sequences of varying lengths to compare it with transformer-based models effectively.

The results of our experiments can be seen in Table 5.15 and 5.16 for the Movielens dataset. The models trained with five different sequence lengths, as well as the full-size model (all), performed well. The Movielens20m dataset showed slightly better results with the full-size model, while the Movielens25m dataset had slightly better results with the 5-length sequence models. However, the performance of baseline method was significantly lower compared to transformer-based models.

Table 5.15. Results of the Baseline Model - Movielens20m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
5	**0.0296**	0.0406	0.0613	0.105
25	0.0161	0.0227	0.0336	0.06
100	0.0217	0.0298	0.0444	0.0770
All	**0.0297**	**0.0407**	0.0617	**0.1053**

Table 5.16. Results of the Baseline Model - Movielens20m

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
5	**0.0317**	**0.0427**	**0.0646**	**0.1084**
25	0.0152	0.0212	0.0319	0.0557
100	0.0237	0.0327	0.0493	0.0852
All	**0.0317**	0.0426	0.0645	0.1080

We also applied the same methodology to the G1 news dataset. However, since the sequence lengths in this dataset are relatively short, we did not train models with

different sequence lengths as we did for the transformer-based models. Instead, we used the entire sequence, similar to the "all" model in the experiments with the Movielens dataset. We utilized the entire dataset (16 days) during the conduct of these experiments.

Table 5.17. Results of the Baseline Model - G1 News

	ndcg@10	**ndcg@20**	**recall@10**	**recall@20**
All	0.076	0.092	0.144	0.208

The best score we obtained using transformer-based models in the G1 news dataset was around 0.25 for recall@20. These results indicate that the transformer-based models performed well when compared to our baseline model. As shown in Table 5.17, the baseline model achieved an approximate score of 0.21 for recall@20. Although there is a significant difference in performance between the Movielens and G1 news datasets, the word2vec model performed relatively well on the G1 news dataset. We attribute this to the nature of the G1 news dataset, which consists of item ids that frequently change hourly. In other words, new news articles are published every hour, and older articles are not included in subsequent interactions. Therefore, article ids in our data are often accompanied by articles published simultaneously, resulting in a significant difference in article ids between time windows. The model also understands that certain article ids are closely associated at specific periods, resulting in similar vectors. However, in the Movielens dataset, a movie can be rated for 25 years, which might cause the model to consider it similar to all other movies during training. This difference between datasets leads to the observed discrepancy in performance between the two models.

5.7. Comparisons with Other Research

This section will provide information about other studies that have conducted experiments using our datasets or that have similar results to ours using different datasets.

In the 2022 ACM RecSys challenge [16], one of the results in Lu, Y et al. [17], ranked second in the competition, is that the experiment results get worse as session length increases. It is stated that the reason is that as the length increases, the model learns old fashion and cannot keep the current situations in the memory well. Our experiments with the Movielens datasets also support this claim because we get negatively affected results when we increase length. Moreover, the session lengths of the datasets used in the [18] Transformers4rec paper are also short.

With the G1 news dataset [5], Moreira et al. [13] developed a model called CHAMELEON. In this study, one of the experiments was conducted by fine-tuning and evaluating for every hour as we did. They did this experiment using the last day of the data. In other words, it is the last 24 hours in the G1 news dataset. The Recall@5 technique has been used for evaluation. Although it varies from hour to hour, the results between hours vary from 65% to 75%. Also, experiments with this dataset in Transformers4rec's paper [18] were made using 16 days. The results were around 67%. We couldn't implement the same experiments since the code used while doing experiments here is a bit old. We think that they get better results than us because of hyperparameters they use.

Experiments were performed using the Movielens20m dataset in BERT4Rec [15]. In this study, unlike us, they took all their interactions instead of taking the last n interactions of the users. This approach is very different from ours. Recall@10 results are approximately 75% in their experiment.

6. CONCLUSION AND FUTURE WORK

Recommender systems are computer programs that aid us in making important decisions regarding the movies we watch, clothing, vacation destinations, and even food choices. These systems play a crucial role in our daily lives, and it's essential that they function optimally for both the users and the businesses that utilize them.

In this study, we thoroughly investigated the effectiveness of various features in sequential and session-based recommender systems. Through a series of comprehensive experiments, we aimed to gain a deeper understanding of the behavior of these features and to interpret the results in the context of the dataset's patterns.

During our research, we utilized the cutting-edge library, Transformer4Rec, which facilitates the effortless implementation and deployment of state-of-the-art transformer models in recommendation tasks at the production level. This library is favored among the community for its ease of use and versatility in recommender systems.

We have demonstrated that Transformer-based recommender systems improve performance when the sequence length is kept short. Additionally, our data analysis revealed that seasonal user preferences exhibit variability over time. Our models can effectively learn and adapt to these periodically changing customer preferences, as evidenced by the results. Furthermore, we observed that incorporating time-based features outperforms other features in our models.

As this study utilizes experimental methods, it would be beneficial to offer recommendations for implementation in production settings in future work. One approach could be to develop an automated feature analysis process similar to the one used in this thesis before deploying a model in production. This could be achieved by designing a pipeline that utilizes a library for this purpose and allowing users to select which features they would like to include or exclude. Additionally, it would be helpful

to provide users with a visual representation of the essential features, similar to the experiment where we tested individual features. This would enable users to quickly understand the impact of feature engineering on the model's performance.

REFERENCES

1. Schrage, M., *Recommendation Engines*, The MIT Press Essential Knowledge Series, 2020.

2. Schrage, M., "Observations", 2008, `www.technologyreview.com/2008/04/22/220919/recommendation-nation`, accessed on March 11, 2023.

3. Ricci, F., L. Rokach and B. Shapira, *Recommender Systems: Introduction and Challengesg*, Springer US, Boston, MA, 2015.

4. G1 News, `https://g1.globo.com`, accessed on March 11, 2023.

5. G1 News Dataset, 2019, `https://www.kaggle.com/datasets/gspmoreira/news-portal-user-interactions-by-globocom/`, accessed on March 11, 2023.

6. Movielens, 1997, `https://movielens.org/home`, accessed on March 11, 2023.

7. Isinkaye, F., Y. Folajimi and B. Ojokoh, "Recommendation Systems: Principles, Methods and Evaluation", *Egyptian Informatics Journal*, Vol. 16, No. 3, pp. 261–273, 2015.

8. Rendle, S., C. Freudenthaler and L. Schmidt-Thieme, "Factorizing Personalized Markov Chains for Next-Basket Recommendation", *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, 2010.

9. Mikolov, T., K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space", *1st International Conference on Learning Representations*, Scottsdale, Arizona, USA, 2013.

10. Le, Q. and T. Mikolov, "Distributed Representations of Sentences and Documents", *Proceedings of the 31st International Conference on International Conference on*

Machine Learning, Vol. 32, Beijing, China, 2014.

11. Grbovic, M., V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan and D. Sharp, "E-Commerce in Your Inbox: Product Recommendations at Scale", *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2015.

12. Hidasi, B., A. Karatzoglou, L. Baltrunas and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks", Arxiv:1511.06939 [cs], 2016.

13. de Souza Pereira Moreira, G., F. Ferreira and A. M. da Cunha, "News Session-Based Recommendations Using Deep Neural Networks", *DLRS 2018: 3rd Workshop on Deep Learning for Recommender Systems*, Vancouver, BC, Canada, 2018.

14. Kang, W.-C. and J. McAuley, "Self-Attentive Sequential Recommendation", *IEEE International Conference on Data Mining (ICDM)*, Singapore, 2018.

15. Sun, F., J. Liu, J. Wu, C. Pei, X. Lin, W. Ou and P. Jiang, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer", *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2019.

16. Landia, N., R. Mcalister, D. North, S. Kalloori, A. Srivastava and B. Ferwerda, "RecSys Challenge 2022 Dataset: Dressipi 1M Fashion Sessions", *Proceedings of the Recommender Systems Challenge 2022*, New York, NY, USA, 2022.

17. Lu, Y., Z. Gao, Z. Cheng, J. Sun, B. Brown, G. Yu, A. Wong, F. Pérez and M. Volkovs, "Session-Based Recommendation with Transformers", *Proceedings of the Recommender Systems Challenge 2022*, Seattle, WA, USA, 2022.

18. de Souza Pereira Moreira, G., S. Rabhi, J. M. Lee, R. Ak and E. Oldridge, "Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation", *Fifteenth ACM Conference on Recommender Systems*, New York,

NY, USA, 2021.

19. Harper, F. M. and J. A. Konstan, "The MovieLens Datasets: History and Context", *ACM Transactions on Interactive Intelligent Systems*, Vol. 5, No. 4, pp. 1–19, 2015.

20. Vig, J., S. Sen and J. Riedl, "The Tag Genome: Encoding Community Knowledge to Support Novel Interaction", *Association for Computing Machinery*, Vol. 2, No. 13, pp. 1–44, 2012.

21. London, I., "Encoding cyclical continuous features - 24-hour time", `https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/`, accessed on March 11, 2023.

22. Preprocessed G1 News dataset, 2021, `https://drive.google.com/drive/folders/1qSUWRqBflR8EvMKoyLIkSyB3JlwNPNTT`, accessed on December 10, 2022.

23. Kaggle, `https://www.kaggle.com`, accessed on March 11, 2023.

24. Jarvelin, K. and J. Kekalainen, "IR Evaluation Methods for Retrieving Highly Relevant Documents", *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, 2000.

25. Shani, G. and A. Gunawardana, *Evaluating Recommendation Systems*, Springer US, Boston, MA, 2011.

26. Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, "Transformers: State-of-the-Art Natural Language Processing", *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, 2020.

27. SIGIR, "SIGIR eCommerce Workshop Data Challenge 2021, organized by Coveo", https://sigir-ecom.github.io/ecom2021/data-task.html, accessed on March 11, 2023.

28. WSDM, "WSDM WebTour Workshop Challenge 2021, organized by Booking.com", https://www.bookingchallenge.com.

29. NVTablular, https://github.com/NVIDIA/NVTabular/, accessed on March 11, 2023.