Master thesis

Fashion recommender systems with focus on time and seasonability.

Author:
Jaime Ferrando Huertas

Supervisors:
UPV: Roberto Paredes Palacios
H&M: Björn Hertzberg

School: **DSIC - Departamento de Sistemas Informáticos y Computación**
University: **UPV - Universitat Politècnica de València**
Course : **2020-2021**

Contents

Abstract

Recommender systems have a strong presence in today's web platforms, e-commerce, media content, news, or any platform that aims to boost user engagement through user personalized content offerings. These systems collect user information and use it to deliver personalized experiences, minimizing the overload of content while providing the most relevant content for the user. The state-of-the-art models for recommendation systems have varied during the last years. Models have transitioned from matrix factorization methods to neural-based models using large artificial neural networks.

This thesis aims to explore such neural-based models on a big fashion retailer, H&M. As one of the biggest fashion retailers in the world, we can benefit from user personalized recommendations to improve sales and user engagement with the brand. H&M has been using matrix factorization methods in the past. Still, these methods can not catch temporal relations such as trends or seasons, neither the time evolution in a customer fashion preferences. We will work with specific neural-based models that can capture such information.

The data comes from the Swedish market for H&M consists of web interactions. We will evaluate models with both traditional and diversity metrics. It is essential to H&M to assess elements such as user personalization, diversity and diversity metrics in the recommendations offered. It is not always best to recommend the bestseller article, but instead recommend articles that boost user engagement, which translate into better selling.

Results indicate neural models capture such temporal relations and benefit from them, performing better than the previous methods used at H&M.

Keywords— Fashion; Recommender system; Collaborative filtering; Neural Networks; Recurrent networks; Sequential Recommendation

Resumen

Los sistemas de recomendación tienen una fuerte presencia en plataformas web, el comercio electrónico, las plataformas de medios, las noticias o cualquier plataforma que tenga como objetivo impulsar la participación del usuario ofreciendo de contenido personalizado para el usuario. Estos sistemas recopilan información del usuario y la utilizan para ofrecer experiencias personalizadas, minimizando la sobrecarga de contenido y proporcionando el contenido más relevante para el usuario. Los modelos de estado del arte para los sistemas de recomendación han variado durante los últimos años. Los modelos han pasado de métodos de factorización matricial a grandes redes neuronales artificiales.

Esta tesis tiene como objetivo explorar estos modelos basados redes neuronales aplicados en una empresa de moda, H&M. Siendo una de las grandes marcas de moda, podemos beneficiarnos de las recomendaciones personalizadas de los usuarios para mejorar las ventas y el compromiso del usuario con la marca. H&M ha estado utilizando métodos de factorización matricial en el pasado. Estos métodos no pueden captar relaciones temporales como tendencias o temporadas, ni la evolución temporal en las preferencias de moda de un cliente. Trabajaremos con modelos específicos basados en redes que pueden capturar dicha información.

Los datos provienen del mercado sueco de H&M y consisten en interacciones web. Evaluaremos nuestros modelos con métricas tradicionales y de diversidad. Es fundamental para H&M evaluar elementos como la personalización del usuario y las métricas de diversidad en las recomendaciones ofrecidas. No siempre es mejor recomendar el artículo más vendido, sino recomendar artículos que aumenten la participación del usuario, lo que se traduce en mejores ventas.

Los resultados indican que los modelos neuronales capturan estas relaciones temporales y se benefician de ellas, con un rendimiento mejor que los métodos anteriores utilizados en H&M.

Keywords— Moda; Sistema de recomendación; Filtrado colaborativo; Redes neuronales; Redes recurrentes; Recomendación secuencial

Resum

Els sistemes de recomanació tenen una forta presència a les plataformes web, al comerç electrònic, a les plataformes multimèdia, a les notícies o a qualsevol plataforma que tingui com a objectiu augmentar la interacció dels usuaris mitjançant ofertes de contingut personalitzades pels usuaris. Aquests sistemes recopilen informació de l'usuari i l'utilitzen per oferir experiències personalitzades, minimitzant la sobrecàrrega de contingut alhora que proporcionen el contingut més rellevant per a l'usuari. Els models d'última generació dels sistemes de recomanacions han variat durant els darrers anys. Els models han passat de mètodes de factorització de matrius a models basats en grans xarxes neuronals artificials.

Aquesta tesi té com a objectiu explorar aquests models basats en xarxes artificials en una empresa moda, H&M. Com una dels empreses de moda més grans del món, podem beneficiar-nos de recomanacions personalitzades per millorar les vendes i el compromís dels usuaris amb la marca. H&M ha utilitzat mètodes de factorització de matrius en el passat. Aquests mètodes no poden captar relacions temporals com ara tendències o temporades, ni l'evolució temporal segons les preferències de moda d'un client. Treballarem amb models neuronals específics que puguin captar aquesta informació.

Les dades provenen del mercat suec de H&M i consisteixen en interaccions web. Avaluarem models amb mètriques tradicionals i de diversitat. A H&M és essencial avaluar elements com la personalització de l'usuari i la diversitat a les recomanacions ofertes. No sempre és millor recomanar l'article més venut, sinó recomanar articles que augmentin la interacció dels usuaris, que es tradueixen en una millor venda.

Els resultats indiquen que els models neuronals capturen aquestes relacions temporals i se'n beneficien, funcionant millor que els mètodes anteriors utilitzats a H&M.

Keywords— Moda; Sistema de recomanació; Filtratge col·laboratiu; Xarxes neuronals; Xarxes recurrents; Recomanació seqüencial

Acknowledgements

Chapter 1

Introduction

Recommendation systems are a subset of information filtering systems built to provide individualized and personalized recommendations to users in an ample space of possible options. These recommendations aim to deliver relevant offerings to the user. The systems are built with information from both the user's preferences and item space.

Thanks to the massive increase of both available data and tools to process it, these recommendation systems have gained popularity for the last few years. They saw a rise in popularity thanks to the Netflix price dataset [1] and have been used in many business use cases such as Spotify, Youtube, Amazon, Google, or Airbnb. These use cases work in a different business markets, yet they require a powerful recommender system; this is due to their large amount of offerings and the need to filter them to the user. Recommender systems will use both content and user data to filter the offerings and provide only the most relevant offerings to the user. A personalized and relevant offering of content to the user translates into more interactions.

Recommender systems can be used across different domains, but is it one type of recommender system used across all domains? or is it one type per domain? The answer to the question relies on what available data the domain has. There are different approaches to recommender systems based on the data they used. The main two types of recommendation system are **Content-based**, which examines the user's previous content interactions and the content attributes to recommend new content, and **Collaborative filtering**, which creates a user profile based on their content interactions and recommends content that similar profiles have interacted with. Content interactions can

be explicit or implicit and will determine the way we treat them.

At H&M, we have implicit data in the form of web-based article interactions or purchases. Previous methods used at H&M went with the collaborative filtering approach and created user profiles based on their interactions to recommend items that neighbor profiles had interacted with. With a new set of neural models, we will follow a content-based approach. We will provide user features and user-item interaction history to a model that will output a set of recommendations.

1.1   Problem Statement

In this project, we implement a new neural-based recommender system and compare its performance to the previous matrix factorization model currently in use at H&M. Current implementations at H&M do not capture time relations which are strongly present in the fashion domain, the new neural-based model will have a special architecture to facilitate learning from these relations. In particular, we will examine performance along the following points:

- Performance on next item to interact, traditional classification metrics such as Hit Ratio at K (HR@K).

- Diversity metrics such as Coverage, Overlap, Personalization, and novelty. These metrics allow us to evaluate recommendations from a closer perspective to the user to understand user engagement better.

H&M has a fast phased business case where clothing assortment is modified every season. The business at H&M focuses on fast fashion, a section of the fashion industry, selling high frequency trends with affordable clothes. A recommendation model capable of capturing such trends and adapting quickly to changes could provide performance gains.

1.2   Scope and objectives

This thesis scope is to compare different set of models for our recommendation systems at H&M with an extensive evaluation for both performance and diversity metrics. Our objective won't be to fine-tune

models to achieve their best performance, but rather provide an initial performance measurement to showcase their potential.

Several metrics will evaluate each model's performance to examine both their accuracy and user engagement potential. We will provide conclusions on the results from the H&M point of view. We will perform additional experiments to understand our model's performance, change embedding dimensions for both item and user, study length of user history, and use randomized user histories.

1.3   Thesis outline

The rest of this report is organized as follows: In the second chapter, the main background needed for understanding the report will be given. This background comprehends from recommender systems terminology to an overview of artificial neural networks and the specifics of the models we used. The third chapter explains the specific architecture of the models tested. The fourth chapter gives an overview of the training setup and data processing, with a presentation of experiments results. The fifth section discusses the presented results and the limitations of the project, finishing with future work and conclusions.

1.4   Sustainability and ethics

Recommendation systems are built to interact with users by nature, learning from users interactions and providing them with content recommendations. Such nature raises concerns about their ethics and responsibilities. The initial problems starts with the data used to train such models; did the users provide consent for the usage of their data to train these models? Or consent for the usage of the required data to offer recommendations?. Companies will often collect as much data as possible and use them on their systems. Data collection policies have raised concerns on the European Union, and regulations for users' privacy have started to appear [2]. The company's responsibility is to let the user know what data is collected, how, for how long, and what it will be used to. This data will have to be stored appropriately to protect user privacy and follow necessary regulations.

The impact of the recommendation system can also raise concerns; recommendation systems can produce negative effects on users' life.

After all, the objective is to make the user interact with the offered content. We have seen this happening with social media, where companies changed their chronologically ordered feed for user personalized feeds. This change made it possible to improve user engagement and retain them in their platform for as much as possible, increasing chances for addiction or other unhealthy patterns.

Chapter 2

Background

2.1 Recommender Systems

This section will give an overview of what is needed to know about recommender systems, what types exist, and how they are built. In the introduction chapter, we mentioned how these systems can be applied to different use cases and how the data they are based on will decide the type of system to implement, now we will define the various taxonomies of recommender systems and what data they require. This will allow us to identify a fitting solution to our problem.

2.1.1 Definition

Recommendation systems are a subset of information filtering systems built to provide individualized and personalized recommendations to users in an ample space of possible options. These recommendations aim to provide a more relevant offering to the user.

To create personalized recommendations, the system needs to learn the user preferences while having all the available information on the items space. The system needs to anticipate the user's needs and present new and relevant items the user will find fitting but would not have consumed without the system intervention.

2.1.2 Recommender system types

We now present several types of recommender systems.

- **Collaborative systems:** One of the most common systems, these systems create user profiles based on their interactions with the content (ratings, purchases, clicks). The system will then use similarity methods to find user neighborhoods and create recommendations based on the user neighborhood. These systems perform at best when the user population is high as neighborhoods will be much more populated, and the system can cover all types of users. Collaborative filtering does not take time as input and will assume people who agreed in the past will agree in the future; users will like similar content as they liked before.

- **Content-based systems:** These systems focus on the attributes of the content to create the recommendations. Content such as a cloth garment would be tagged with different attributes such as style, color, size, sex, category. The systems will learn a user profile from the content the user has interacted with and provide recommendations based on it. Content labeling will play a significant role in the model performance, as it will determine its ability to capture inter-content relations. An example of this could be Spotify recommendations, where it will recommend songs within the same genre.

- **Demographic-based systems:** Demographic systems are very close to collaborative approaches in the sense that they will try to find similar users to find relevant recommendations. Still, now user similarity will not be computed on their interaction history but rather on their demographic attributes. Demographic attributes can include location, genre, profession, and other personal attributes. An example of these recommendations could be AirBnB recommending group activities for vacations based on the location and personal demographics of the group.

- **Knowledge-based systems:** Knowledge systems are based on underlying information on the relation between user needs and content. They use the explicit knowledge of the item space together with the user preferences to create recommendations. An example of such systems could be a tourism recommender system where there are no previous interactions of the user with the visiting place, but the system matches certain activities with the user thanks to its preferences and activities knowledge.

- **Utility-based systems:** Utility systems try to calculate the usefulness of the content to a specific user. This can be done by learning a user profile over time or based on user expressed preferences. An example of this recommender would be a clothing size recommender; the user will input desired fit and needs, and the system will find the fit with the most usefulness.

These are the main five types of recommender systems, each one with its advantages and disadvantages. When deciding what type of system we want to implement, we need to study the best fit. If we take the Utility-based approach, we can see how useful it can be when implemented in clothing companies. People have different clothing needs depending on the weather, but weather conditions may not play such a significant part if we use them in Spotify song recommender.

The available user data will play a huge role when deciding what system to use. A famous recommendation problem is the cold start [3], where a new user comes into the system with a blank profile. Systems like collaborative or content-based will not have enough information to provide relevant recommendations, but demographic will as long as the user provides enough demographic attributes when signing in the system.

The mentioned taxonomies are not exclusive; one could implement a mix of collaborative and content-based approaches by implementing content-based user-profiles and the collaborative approach to find inter-user content relevant to the user. One could also implement different recommender systems and then join recommendations with an ensemble approach. These join models are called hybrid systems.

2.1.3 Hybrid systems

We often see recommendation models made of joint systems that complement each other in strengths and weaknesses; these models are referred to as hybrid systems. There are multiple taxonomies to join recommender systems, but these are the most common [4].

- **Weighted recommender** Multiple recommender systems are implemented and joint with an assigned weight. This is the simplest hybrid system and reminds us of traditional machine learning ensemble techniques.

- **Switching recommender** Multiple systems are implemented and used at the same time; the master recommender will decide if to present all recommendations to the user or choose which system to display depending on user features. If the master presents all systems recommendations at the same time, it will be called a *mixed recommender*, but if it chooses one system to display, it will be a *switching recommender*.

- **Cascade recommender** Different systems are combined in a cascade architecture where the output of a system as input for the next system and so on. This system aims to optimize results by using multiple approaches.

2.1.4 Data sources

We have mentioned how the selection of the recommender system is highly affected by the available data sources. In this section, the most common data sources used across recommender systems are presented.

- **User-Content interactions**: Data for all content interactions the user has made. These interactions could be purchasing an item, listening to a song, or rating a film. There are many types of interactions, but they are usually divided into two groups: *Implicit* and *Explicit*. For example, purchasing an item will let us know the user's interest in the item. Still, we do not have any measure of user satisfaction with the item - implicit interaction. If the user rates a film within a scale, we have a metric to capture the user satisfaction with the film - explicit interaction. Often explicit interactions hold more value than implicit as they represent a more real metric of user satisfaction; there are data enrichment techniques for implicit interactions to come close to an explicit interaction, let's say you listen to a song in Spotify, you are not providing any rating, but the system could capture if you listen to the full song or if you listen to it multiple times to differentiate from other interactions.

- **Content attributes**: Contains all data attributes related to the content the system works on. In a Spotify recommender system those could be song genre, artist, publish date, and many more.

- **User attributes**: All user data from demographic attributes to preferences. This data will play a significant role in systems that try to model the users.

- **Functional data**: Data that could provide the system with functional requirements or information that escapes the rest of data sources. An example could be the weather forecast; a clothing recommendation system could boost waterproof items if the forecast is rainy.

2.2   Recommendation techniques

In this section, we describe the most common techniques used to create recommendation systems.

2.2.1   Collaborative filtering

Collaborative filtering approaches have been the most popular choice of recommender systems in the last decade. Thanks to the massive amount of data platforms started collecting; they found techniques to model the user and find similarities in a vector space. Traditional methods use factorization to create a latent space that captures interactions between users and items.

These systems require a historical list of content interactions per user to create a user-profile that will later be used to compute distances between users. The historical list of interactions is not required to be chronologically ordered as the order does not affect the model. The system is highly sensitive to the number of interactions and can be extremely affected by the cold-star problem with new users. Another caveat of these systems is the high sparsity in the user-item matrices.

Nearest neighbors

A technique used both on user-based and item-based neighborhood methods. The system creates a similarity between users and finds the nearest neighbors to the current user to find content neighbors interacted with. The utility of an item s for a user c in can be estimated as [5]:

$$\hat{u}_{c,s} = \frac{\bar{u}_c + \sum_{c' \in N_c} (u_{c',s} - \bar{u}'_c) \times \text{sim}(c, c')}{\sum_{c' \in N_c} \text{sim}(c, c')} \tag{2.1}$$

Where we have K nearest neighbors of user c in N_c, $\bar{u}_c$ the mean utility of previous items the user has interacted with and $sim(c_1, c_2)$ being a similarity measure between users such as cosine similarity, Pearson correlation or any dimensional similarity measure.

Matrix Factorization techniques

Techniques that become popular after the Netflix Price [1]. A competition on collaborative filtering methods won with such techniques [6]. Matrix factorization techniques present a new approach to solve the high sparsity in traditional collaborative methods. These methods split the user-item matrices into multiple smaller matrices. The user-item interaction matrix is split into two lower dimensional matrices, the first one has a row for each user, and the second has a column for each item. Each row or column corresponding to a user or item will be referred to as latent factors [7] that will create a latent space.
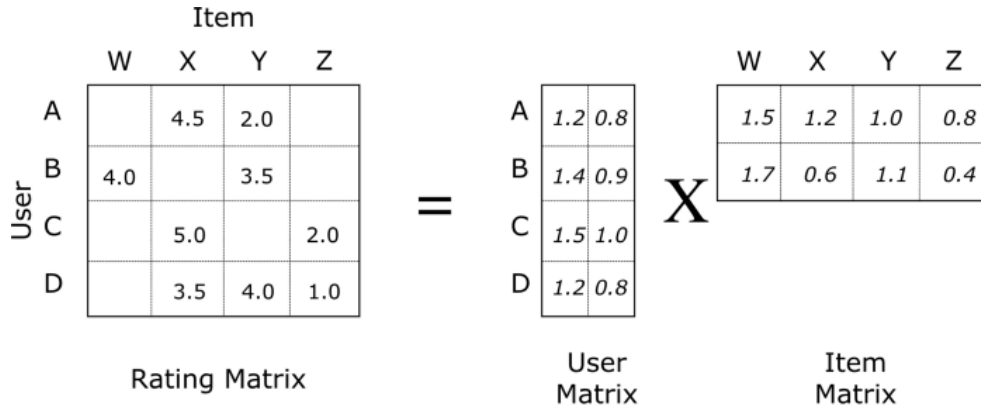

Figure 2.1: Matrix factorization example for a movie ratings dataset.

In the first implementation applied to the Netflix Price [8] we find the predicted rating per each user u for a given item i is defined as follows:

$$\tilde{r}_{ui} = \sum_{f=0}^{n \text{ factors}} H_{u,f} W_{f,i} \tag{2.2}$$

Ratings can be computed as:

$$\tilde{R} = HW \tag{2.3}$$

Where $\tilde{R}$ is the user-item matrix, H contains user latent factors, and W is the item latent factors. Such logic is referred to as the original matrix factorization method, but several other optimizations have been raised since then with SVD [9], and ALS [10]. Alternating Least Square (ALS) has been a dominant approach thanks to its parallel computation approach.

Neural based

With the rise of neural networks in different fields, they also reached recommendation systems [11]. Deep learning methods have been developed to tackle the collaborative filtering approach. There have been proposed methods based on denoising autoencoder networks [12]. Still, it was not until Deep Item-based Collaborative filtering [13] that deep learning methods beat traditional techniques. DeepICF made use of attention layers which already revolutionized the natural language field [14].

Deep learning methods performed well in recommendations because of their power to find good items and user representations in an embedding space. Embedding layers have been used previously in natural language processing and revolutionized the field due to the dramatic decrease in sparsity to represent words and their accurate representation.

2.2.2   Content based

Content-based recommender systems are built around item attributes and user profile preferences. The basic idea is to recommend items that are similar to what the user liked before [15]. These methods are suitable for situations with extensive data about the items and do not require user attributes such as demographic data. The model will build a user preference profile based on the history of its interactions and use it to recommend items the user will like.

Vector Space Model

The Vector Space Model [16] is the most famous method for content-based recommenders. The model inspires information retrieving systems to implement an item scorer. It treats item attributes as keywords and uses the TF-IDF [17] to calculate weights based on those keywords. Given an item, d_j, its content is defined by a set of weights w_ij where each weight corresponds to the ith keyword of the item d_j.

$$\text{Content}(d_j) = \{w_{1j}, w_{2j}, \ldots\} \tag{2.4}$$

Content-based systems will recommend items similar to what the user previously liked. To do so, it will create a user profile based on the history of items [18].

$$\text{ContentBasedProfile}(u) = \frac{1}{|N(u)|} \text{sum}_{d \in N(u)} \text{Content}(d) \tag{2.5}$$

Where $N(u)$ is what the user u previously liked. Once we calculate the *ContentBasedProfile* for all users, we can create a score $p(u, d)$ to predict what a user u will give to the item d.

$$p(u, d) = \text{sim}(\text{ContentBased Profile}(u), \text{Content}(d)) \tag{2.6}$$

Where sim is a similarity measure like Pearson correlation or cosine distance. These systems will be highly dependent on the keywords (item attributes) available for the items. The more accurate attributes to describe the item space, the better the models will perform.

Neural based

Neural-based models bring a new approach to item representations [19] thanks to their embedding layers. Item representations can be learned without specific attributes. In [20] Spotify learns songs embeddings from the sequence they are present without the need for song attributes such as genre or artist.

Neural-based approaches treat the problem as a regular classification situation and use artificial networks to predict the next item the user will like based on both user and item embeddings. In [21] we see a basic RNN approach to the YouTube recommendation system; the system receives the user video history as an ordered sequence and

predicts the next video to watch. Posterior improvements with additional context features were made [22]. Other approaches for click rate prediction with attention encoders have been implemented by Alibaba [23].

2.3   Artificial Neural Networks

Artificial Neural Networks (ANN) are computing models inspired by the biological neural network in animal brains. An ANN is based on a group of compute units called neurons, similarly to neurons in the biological brain. These systems are often applied in pattern recognition [24] problems such as data analysis, information retrieval, classification, regression, or computer vision. ANNs mimic the animal brain and learn from examples and do not need specific task programming, which is why they have seen a rise in popularity and the massive amount of data in the digital world. Here is an example of an ANN diagram
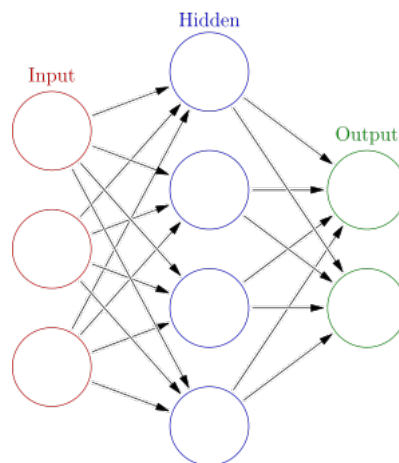

Figure 2.2: Example of a simple ANN.

The neurons building an ANN are compute units that receive one or more inputs, weightily sums them, and apply an activation function dependent on the sum to produce an output that other units can use. These units are connected in levels (layers) to form an ANN, also called fully connected networks.
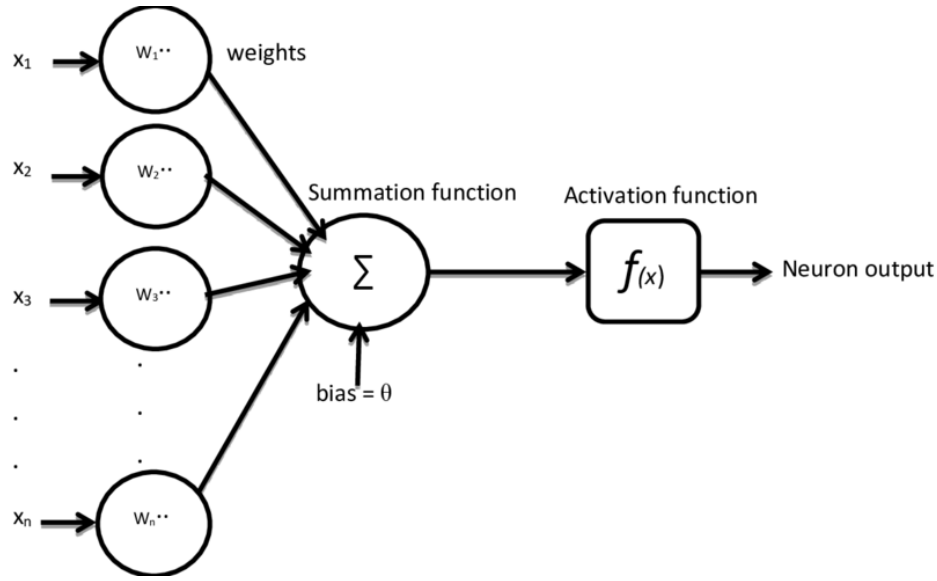
Figure 2.3: Artificial neuron diagram [25].

When creating an ANN, the activation functions of the neurons will play an important role, many mathematical functions have been used in the field, but not all will fit the requirements for each use-case. The most common activation functions are RELU, Leaky-RELU, Identity or Sigmoid [26] but there are others such as Gaussian, Hyperbolic tangent, Binary, or GELU. Here are the most common functions

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \qquad \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

(a) Rectified linear unit (ReLU)[27] (b) Leaky RELU [28]

$$f(x) = x \qquad\qquad f(x) = \frac{e^x}{e^x + 1}$$

(c) Identity (d) Sigmoid function

Figure 2.4: Most common neuron activation functions

Every set of weights in a neuron layer are learnable during training. Weight learning is the foundation of neural networks training,

and while there are multiple mathematical techniques to learn them, the most common one is Backpropagation [29]. In Backpropagation we need an error function to evaluate how well the current weights are performing, for example, the RMSE error for a regression problem. We will derive our learnable weights against that error function to travel within the hyperplane of weight values, finding the ones with the lower error function. This algorithm is computed inversely; the output layer errors are calculated first and then propagated to the previous layer to compute their errors. The process is repeated until reaching the first layer.

2.3.1   Feedforward Neural Network

Feedforward neural networks (FFNN) are the basic iteration of ANN where neurons can only be connected forward, not establishing a connection with themselves or previous neurons. Information will only move forward, starting from the input layer to the hidden layers and finally the output layer. A diagram of FFNN can be seen here:

Input Values   Input Layer   Hidden Layer 1   Hidden Layer 2   Output Layer
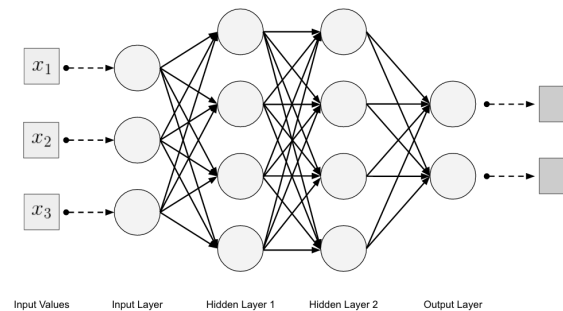
Figure 2.5: Example of a FFNN [30].

These networks are often referred to as multi-layer perceptrons and created the foundation of neural networks today.

2.3.2   Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a class of ANN where connections between neurons can form a directed graph. Cycles are formed in the neuron connections, where a neuron receives its own output. RNNs are commonly used to process sequences of inputs with variable length, as they create an internal state(memory) and process each

input sequentially, capturing sequence relations. This makes them specially useful for task as handwriting recognition [31] and speech recognition [32]. Here we can see a diagram of an RNN cell.
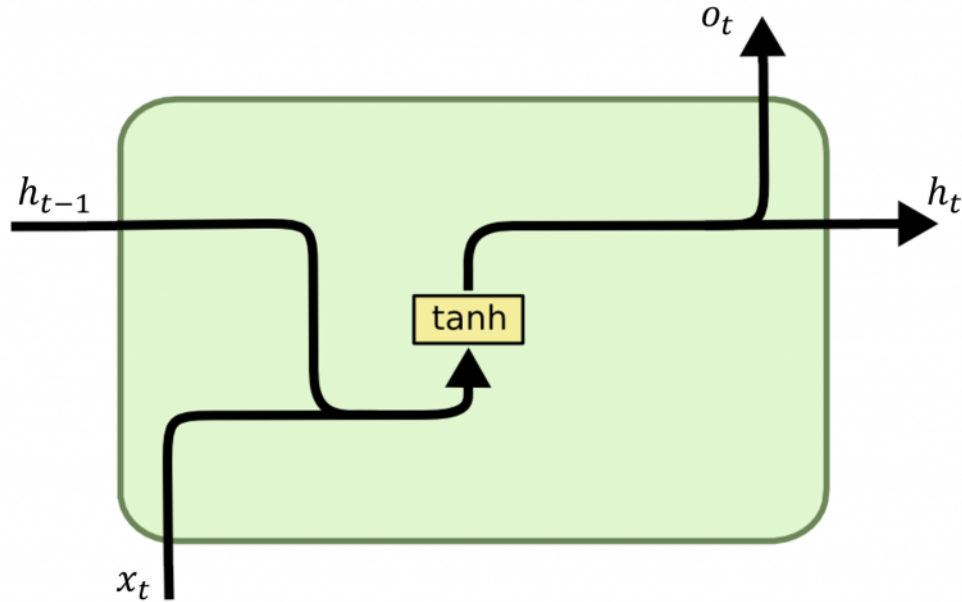
Figure 2.6: RNN cell [33]

The cell receives both the input vector x_t and the previous hidden layer vector h_{t-1} : and behaves as a standard neuron.

2.3.3   Long short-term memory - LSTM

Long short term memory(LSTM) ANN are a special RNN architecture [34]. LSTMs replace RNN neurons by LSTM cells where that are is composed of three gates:

- Input gate

- Output gate

- Forget gate

Each of the gates can be treated as a single neuron. Thanks to the more elaborate architecture, LSTM cells can learn time dependencies and makes them especially useful for time series, machine translation,

or any use case where time plays a role. Generally, RNNs are outperformed by LSTMs. Here we can see a diagram of an LSTM cell.
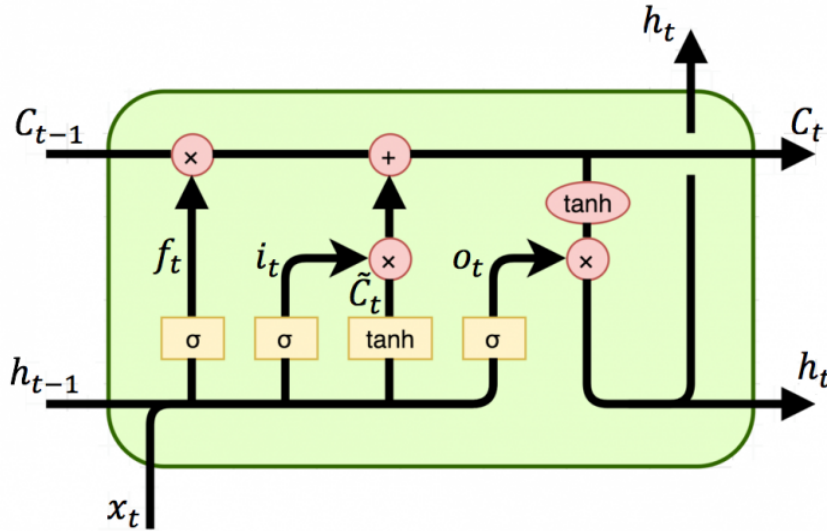
Figure 2.7: LSTM cell diagram [33].

The cell receives the input vector x_t, previous hidden layer vector h_{t-1} : and the previous cell state c_{t-1}. From left to right, forget gate f_t, input gate i_t, cell state candidate values $\bar{C}^{(t)}$ and output gate o_t. All the elements will control the flow of the input data through the cell by adding or removing information from the cell state $C^{(t)}$.

2.3.4   Gated recurrent unit - GRU

A Gated recurrent unit (GRU) [35] is another type of RNN architecture similar to LSTMs. GRU cells have two gates, a reset gate [36] and an update gate. The reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to maintain. If the reset gate is set to 1 and the update gate to 0, it will be equivalent to a plain RNN.

GRUs have seen better performance than LSTMs in speech recognition task [37], but it is still unclear which cell performs best. Here we can see a diagram of an LSTM cell.
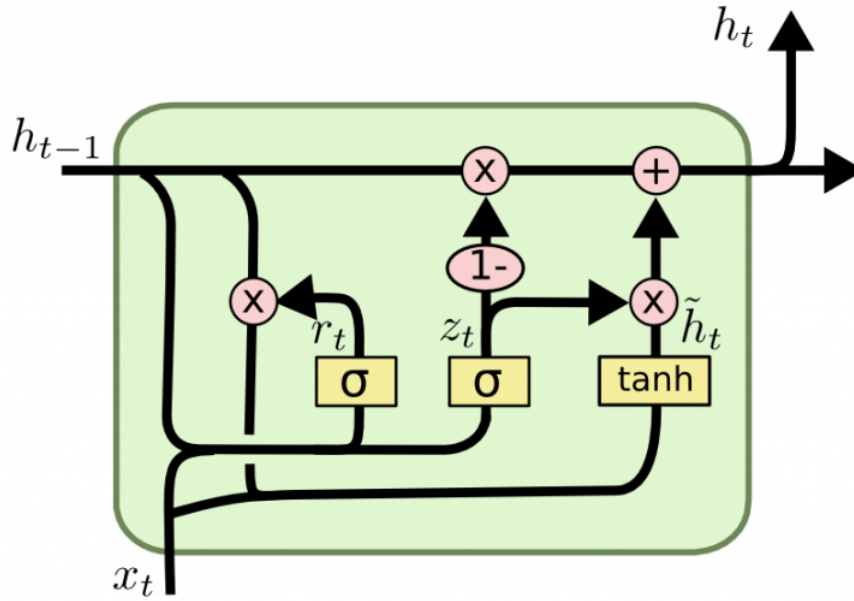
Figure 2.8: GRU cell diagram [33].

The cell receives the input vector x_t and previous hidden layer vector h_{t-1} :. Reset gate r_t and update gate z_t will determine the flow of input data that the cell receives.

2.3.5 Encoder-decoder architectures

The encoder-decoder (enc-dec) architecture is an ANN architecture widely used for tasks where sequences are mapped to another sequence, such as machine translation or speech recognition [38]. The network is composed of one encoder and one decoder. The encoder transforms each item into a corresponding hidden vector containing the item and its context. The decoder reverses the process, turning the vector into an output item, using the previous output as the input context.
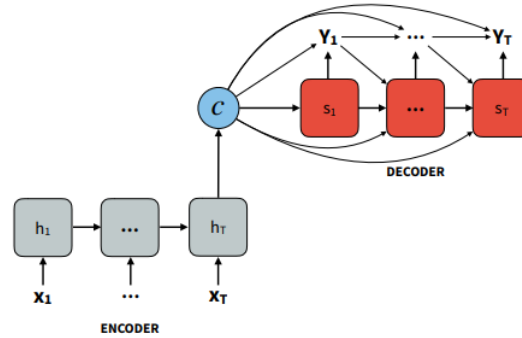
Figure 2.9: Encoder-decoder diagram [39].

These architectures have received many optimizations such as attention layers, beam search for decoders. They were the bases for transformers models, the current state of the art for deep learning machine translation and speech recognition models.

2.3.6 Attention

The attention mechanism was introduced in machine translation by [14] and iterates on the encoder-decoder architecture. While in the original enc-dec implementation, the decoder will transform the entire input sequence into one fixed vector, which is then passed to the decoder. The attention architecture allows the model to store all hidden states of the decoder that are then passed to the decoder. The decoder will use all previous hidden state vectors in the form of a weighted sum that passes into each decoder step. The weights of each decoder step are decided by a trainable layer referred to as the alignment model. Other suggestions to modify the alignment model have been made, with the dot product attention of the decoder hidden states and decoder states being the most popular.

Figure 2.10: Encoder-decoder with attention diagram [39].

2.3.7 Transformers

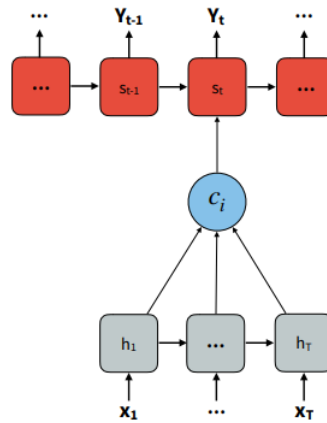Transformer models are based on the encoder-decoder architecture and add multi-head attention mechanisms in both encoder and decoder. With transformers, both encoder and decoder can consist of multiple sequentially connected blocks. Transformer models are the current state of the art in natural language processing.
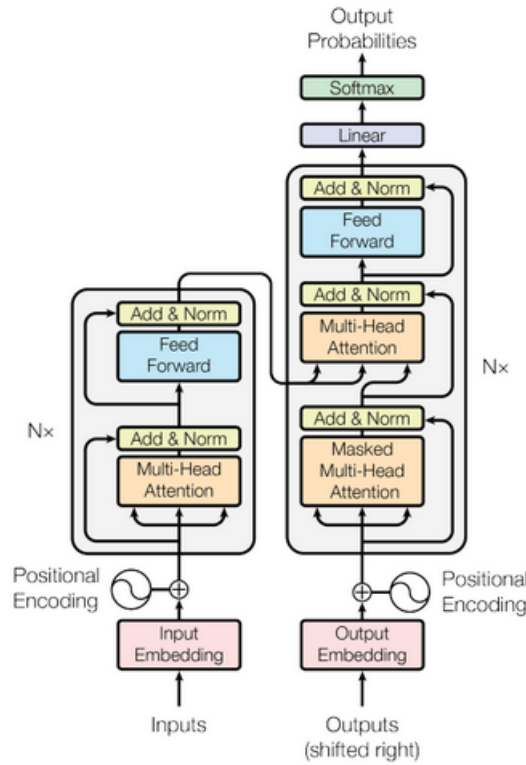
Figure 2.11: Transformer model architecture [40].

The encoder has N sequentially connected blocks. The first block receives an input vector along with its positional encoding to form the encoder input. The encoder input goes through multi head attention and a feed forward layer. Output will be sent to the next encoder block or decoder if it's the last encoder block.

Each transformer encoder consists of two parts: multi-head attention and a feed-forward layer. The attention mechanism receives inputs from the previous encoder and processes them in a feed-forward layer to create the output encodings. Each output is individually processed by the feed forward layer and is sent to the next encoder and all decoders. The first encoder also receives positional encodings to respect each input position in the sequence; this is necessary as there is no other information about the sequence order.

The transformer decoders have three main parts: masked multi-head attention, multi-head attention, and a feed-forward layer. The masked attention will process the decoder inputs and send them to the next multi-head attention with all encodings. Last attention outputs

will be processed by a feed-forward layer similarly to the encoder. The first decoder will also receive positional encoding to respect each input in the sequence.

After the last decoder block, the original transformer implementation had a set of linear layers and a softmax layer to produce output probabilities over the vocabulary.

2.4   Natural language processing

We have provided an overview of neural-based models used in Natural Language Processing (NLP), in which we will base our experimentation to create recommender systems. These models also used techniques from traditional NLP that allow them to generate word representations, embeddings. This technique is especially important as reducing input sparsity will generally improve the model's performance or ability to learn from such input. For our proposed models, we will also use such techniques to learn item representations. Better content representations will come in handy for content base models, highly dependent on content attributes.

2.4.1   Word Embeddings

Traditional methods to represent words such as one-hot encoding suffered from high dimensionality and sparsity problems. Word embeddings arrive to solve those problems in the form of a dense real-valued vector to represent words. Embeddings should encode the meaning of the word based on their meaning, words with similar meanings should be closer in space.

Word embeddings were originated before neural networks [41]–[43]. Still, it was not until Word2vec [44] that they become popular as they enabled neural networks to process natural language datasets and exploded their potential. Word embeddings can be learned in two different ways, both based on context information.
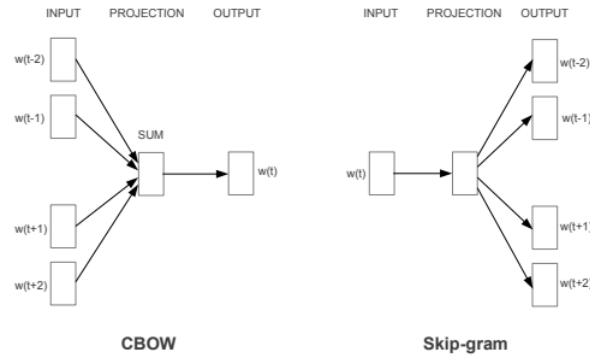
Figure 2.12: CBOW and Skip-gram diagrams [44].

The first model, continuous bag of words (CBOW) receives a window of the surrounding context's word to predict the current word. Skip-gram in the other hand uses the current word to predict the surrounding window of context words.

2.5   Evaluation

This section provides a brief overview of what metrics are commonly used when evaluating recommender systems. We provide two sets of metrics to evaluate both the performance and diversity rate of a recommendation system.

2.5.1   Performance metrics

Performance metrics measure how good our recommender system is at predicting the content the user will consume. They provide an objective measurement of how accurate offering the recommendations are against the test data. We will be using precision at K for our evaluation, but there are other relevant metrics as NDCG, also known to be used in recommendation systems evaluation.

Hit Ratio at K

Hit Ratio at K measures the precision of the recommendation up to K recommendations. Hit Ratio is defined as the number of correctly

predicted items divided by the number of predicted items, which is K. It's calculated as follows

$$\text{Hit Ratio@k} \ = \frac{\text{true positives } @k}{(\text{ true positives } @k) + (\text{ false positives } @k)} \quad (2.7)$$

This metric does not measure how accurate recommendations are ranked; it will not matter whether a recommendation is ranked first or third as long as it is within the top K. This measure is commonly referred to as precision at K (P@K)

2.5.2 Diversity metrics

Recommendation systems have been traditionally evaluated with performance metrics as the ones explained previously. Still, there is an increasing realization that accuracy alone might be a suboptimal strategy for a successful user experience [45]. Diversity metrics complement performance metrics and provide a deeper view of how recommendations distribute. A recommendation model could achieve good performance by simply recommending the top consumed content, drifting away from tailored user experiences. In the case of H&M, recommending top-selling articles such as white shirts, white socks may show good performance metrics on paper, but it won't translate into greater user engagement. Was the user going to buy white shirts anyway, as they are the most popular item? How does the user feel when receiving non-personalized recommendations? These metrics allow us to evaluate closer to the user perspective and achieve a better user experience.

Overlap at K

Measures the percentage of unique recommendation sets in the full recommendation list. If we had a thousand users with their own recommendation set, the value would be one if all users got a distinct recommendation set and zero if they all received the same recommendation set. We will use $1 - Overlap@k$, and it's calculated as follows.

$$\text{Overlap@K} \ = 1 - \frac{\text{N}^\circ \text{ of distinct of recommendation sets}}{\text{N}^\circ \text{ of recommendation sets}} \quad (2.8)$$

This metric is crucial to measure how *personalized* our recommender is and detect when it is overfitting to a particular set of recommendations.

Personalization at K

Personalization at K measures uniqueness from all recommendations sets the system creates. While Overlap@K measures uniqueness in binary (unique or not unique) Personalization@K measures a more complex similarity, we calculate the dissimilarity (1- cosine similarity) across all user's recommendations. User recommendations are encoded in a vector to help measure the distance between two recommendation sets. We calculate the Personalization@K as follows:

$$
\begin{aligned}
\text{Personalization@K} &= 1 - \text{mean cosine similarity} \\
&= 1 - mean(\sum_{i \in U} \sum_{j \in U} c(i,j))
\end{aligned}
\tag{2.9}
$$

Being U the full set of users recommendations and $c(i,j)$ the cosine similarity between vectors i and j. This metric provides a view of how distant are the unique sets of recommendations between them.

Novelty at K

Novel recommendations are referred to as content the user did not know about [45],Novelty at K aims to measure the novelty score of a recommendation list. Recommender systems can not accurately know if the content was unknown to the user, so they assume the less popular an item is, the more novel to the user it will be [46]. We can calculate novelty at top-K recommendations by summing the novelty of each recommendation based on its popularity.

$$
\text{Novelty@K} = \frac{1}{|R|} \sum_{i \in R} POP_i = \frac{1}{|R|} \sum_{i \in R} -\log(p(i))
\tag{2.10}
$$

With R being the set of top-K recommendations and $p(i)$ a function to determine the popularity of a recommendation item. at H&M we calculate popularity as the number of sales the item has.

Coverage at K

Coverage at K measures the percentage of items that the model covers in its recommendations. It's often measured across all user recommendations the model can create, as measuring it in a single user recommendation result would not have much sense. We can calculate it as follows:

$$\text{Coverage@K} = \frac{\text{N}° \text{ of distinct content in recommendation}}{\text{Total N}° \text{ of distinct content}} \quad (2.11)$$

All these metrics can also be calculated on content attribute level for more detailed analysis. At H&M we often calculate these metrics on a category level to understand what categories our customers are receiving in their recommendations and intervene if they have non-desirable results, for example, only home items.

Chapter 3

Models

In this section, we will go over the different model architectures evaluated. We start with the current model at H&M and add new neural architectures based on NLP models. NLP Neural-based architectures do a fantastic job capturing time and order relations within a sequence. We expect the same when moving them in the recommendation system usage.

3.1 Baseline - AALS

AnnoyAlternatingLeastSquares (AALS) is a matrix factorization method. Following the algorithm described at 2.2.1 we will build matrices for both user and items containing the latent factors. We will use Approximate Nearest Neighbour Search [47] (ANNOY) within the matrices to find the approximate closest neighbors to create recommendations. We use ANNOY and not the correct Neighbours to save computational cost, these models are trained daily, and computational time plays a significant role.

3.2 Neural Sequencer RNN

Basing on Google's work with Youtube recommendations [21] we implemented a similar model adapted to H&M use case. This Neural Sequencer RNN (N.S RNN) model consists of two embedding layers, a multi-layer RNN, and a final classification linear layer. The model takes user id and user web browsing history as a sequence to predict

the next item the user will interact with. Embeddings have a 256 size, RNN layers have 512 neurons and the final linear layer has 512 (twice the embedding size due to the concat layer).
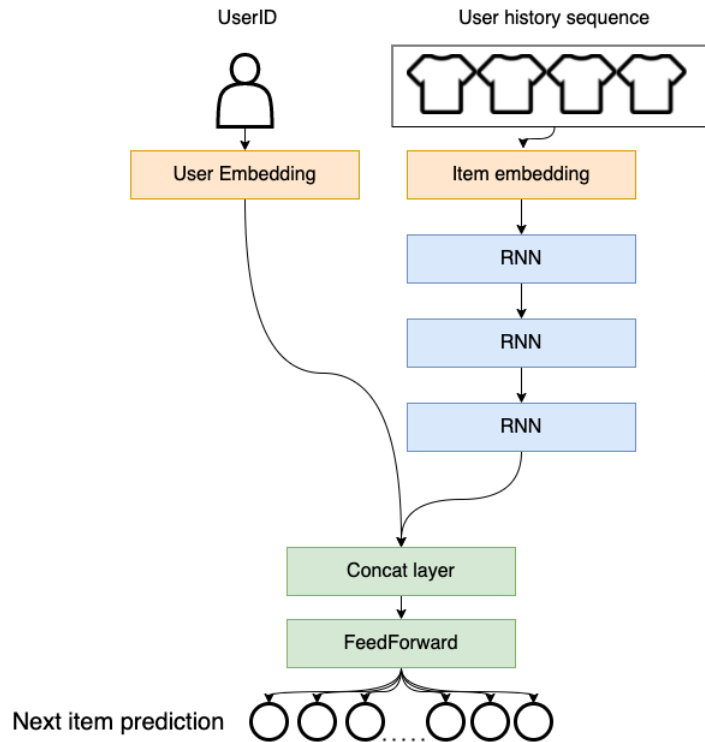
Figure 3.1: Neural sequencer RNN schema.

The user id and user sequence of item interactions will pass through their respective embedding layers. The sequence of item representations will then pass through a three-layer LSTM which latest state will be concatenated to the user id representation and finally sent to a final classification layer predicting the next item the user will interact with. RNN gates are known not to be the most efficient gate for recurrent networks. We will also try another two variations with both LSTM and GRU cells.

3.3   Neural Sequencer with attention

Following NLP research, our next experiment consists of the addition of an encoder-decoder architecture with attention. The attention layer

was a breakthrough on machine translation [14]. Schema of the model.
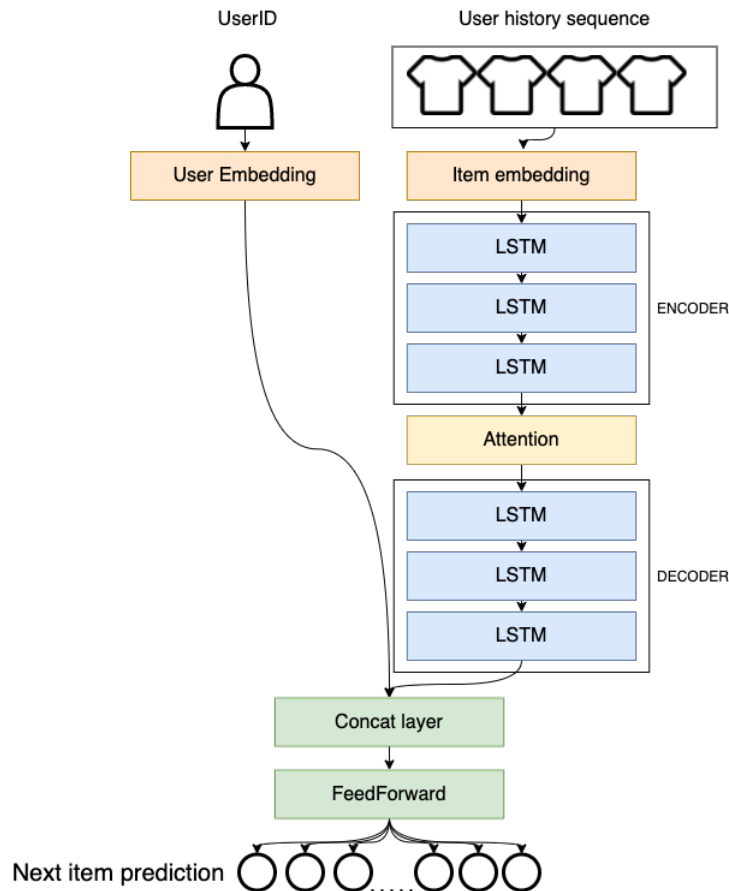
Figure 3.2: Neural sequencer Transformer schema.

In the attention model, we expand upon the N.S RNN. We use the three initial layers as an encoder that passes its hidden states to an attention layer connected to a three-layer LSTM decoder which is connected to the initial concat layer. All LSTM layers have 512 neurons

3.4   Neural Sequencer with transformers

Transformers have been state of the art in NLP during the last years since [40] and will be evaluated for our use case as well. We will implement a transformer recommendation architecture similar to how Alibaba created the Behavior-Sequence-Transformer model [23] based

on user history sequences. The implementation only uses transformer multi-head attention encoders and replaces the transformer decoders with feed-forward layers without attention.

Figure 3.3: Neural sequencer RNN schema.

The transformer encoder block consists of 8 attention heads with 2048 neurons on feed-forward layers (Alibaba implementation). Following leaky Relu feed-forward layers also follow Alibaba implementation with a 1024-512-256 neuron setup.

Chapter 4

Experimentation and results

This section will go over multiple experiments to understand each recommendation model performance and how learned representations, user history length, and sequence order play a role in the model's performance.

4.1 Training setup
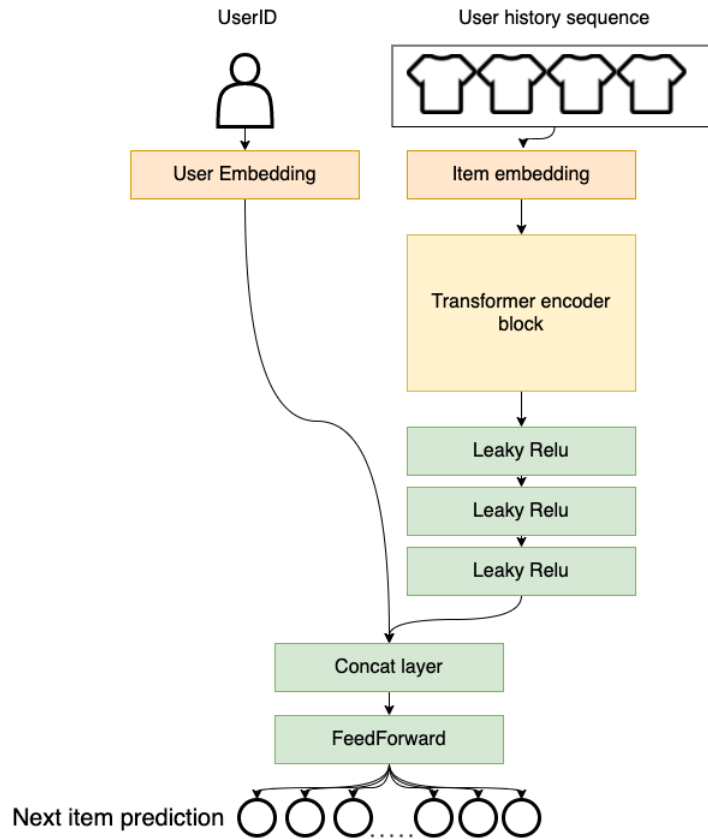
The main objective for the thesis has been evaluating different neural-based models. To keep a consistent training environment, we decided on some common training practices across all experiments. Batch size has been set for 2048, Adam [48] as an optimizer and OneCycle learning rate for fast convergence [49]. Such settings can play a significant role in model performance and are often a full research subject. To avoid going in an extensive singular model study, we went with the most common practices. We trained all experiments for an unlimited number of epochs with a patient policy of three, meaning that we will stop training when three consecutive epochs show decreasing model performance. Reported results are on the best performing epoch at test set, looking at the top fourteen recommendations.

4.2 Data

4.2.1 Source

Models have been trained with web interaction data from the H&M Swedish website for one month. We used web interactions data and

not other sources such as transactions because of the higher popula-
tion on web interactions. Customers browse the website much more
often than they buy items. A populated data source will help our mod-
els generalize better, learn better representations, and increase perfor-
mance overall. Deep Learning models are known to be highly depen-
dent on the number of data being trained on.

Web interactions collect the history of items the user has seen on
H&M website. Building a recommender capable of predicting the next
item to browse will help users find valuable items based on their his-
tory, increasing user satisfaction and sales.

4.2.2   Preprocessing

Neural models will be trained to predict the next user interaction based
on past user history. A user history of interactions can be used to create
multiple samples that train the network. Given a user history H con-
taining $h_1, h_2, ..h_n$ interactions, a training sample can be created from
each interaction, h_t where the past interactions $h_1, .., h_t - 1$ are used as
past history to predict interaction h_t. A representation of how prepro-
cessing would affect a user history is shown in the following figures.

Figure 4.1: User interaction history.

Figure 4.2: User interaction samples.

When a user interacts with an item consecutively, $h_t - 1$ will be the same as h_t. Consecutive duplicates interactions have been removed to prevent models from falling in local minimum scenarios where they predict the next interaction's latest known interaction.

4.2.3 Split

The training set contains all samples created from users' web history up to the second latest known interaction $h_n - 1$. A test set for web history is created, where the latest interaction h_n is predicted based on the previous history. We limited the test set to the latest interaction to evaluate the model at the current time and not when the model will not ever be used on. Following the preprocessing example, we can see a schema on which samples will go to training or test for the user history.
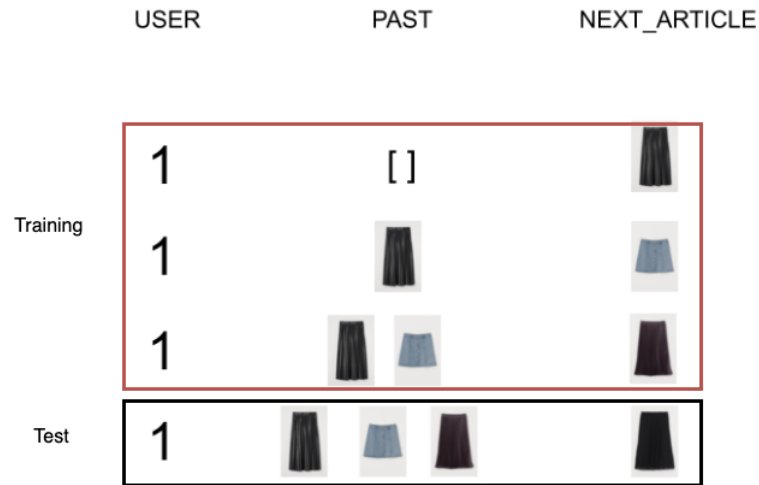
Figure 4.3: User interaction training-test split.

4.3 Results

The result of the different experiments of this project are reported below.

4.3.1 Model comparisons

Best results achieved with each of the models can be found in the following table.

Experiments	HR @14	Coverage @14	Overlap @14	Personalization @14	Novelty @14
AALS	4.42%	0.24%	**14.45%**	56.81%	7.6
N.S LSTM	**50.32%**	**8.75%**	21.54%	**90.57%**	**9.88**
N.S RNN	43.94%	7.68%	19.84%	89.87%	9.51
N.S GRU	49.00%	8.04%	25.94%	88.03%	9.45
N.S Attention	47.46%	7.61%	19.84%	89.55%	9.44
N.S Transformer	44.33%	7.67%	21.97%	86.98%	9.36

Table 4.1: Model results overview

The neural sequencer models obtain a much better result than the baseline model AALS in traditional metrics, around ten times the performance. An improvement was expected, but not so high. In terms of diversity metrics, neural-based models do come ahead in three out of four. They all have much higher Coverage and personalization with a slight improvement in novelty. In the overlap metric, the baseline beats the neural-based models. Overlap measures the raw uniqueness of recommendation sets, while personalization captures the distance between sets. If AALS has a lower overlap and lower personalization than neural based models, this means the users have higher number of unique recommendation sets with AALS than any neural-based model but that the unique sets are not as different between them as they are in neural-based models.

When it comes to neural-based models, the Neural Sequencer with LSTM cells pulls ahead in HR with the GRU cell version being very close. More sophisticated NLP models such as attention and transformers present lower results, but this could be due to their higher complexity and need for extensive fine-tuning. As expected, the original N.S with RNN has the worst performance. RNN was a major improvement in NLP but has been replaced for more complex cells like LSTM or GRU for some years now. Regarding diversity metrics, the N.S with LSTM cells pulls ahead again but not with a big margin, metrics within neural-based models stay fairly consistent, and we believe it is the change to neural architecture that is responsible for the improvements, not the specific tweaking to the neural architectures.

4.3.2 Impact of user and item embedding sizes

To further extend our analysis in model performance and result explainability, we will create several experiments where embedding size for both items and users is modified. We have been training with an embedding size of 256 for both. When reducing the embedding size, we reduce the model's ability to learn representations and stop using them as before. The experiment will provide us insights into how dependent the model is on those representations, helping with the explainability of the results. We will use the Neural Sequencer LSTM architecture across all tests.

We first evaluate how reducing the user embedding dimensions will affect the model. We reduced up to one dimension to push the

limit on how it affects the model.

User Size	Item Size	HR @14	Coverage @14	Overlap @14	Personalization @14	Novelty @14
1	256	49.35%	8.67%	23.88%	89.84%	9.87
10	256	49.92%	8.72%	23.62%	90.32%	**9.88**
20	256	50.28%	8.74%	22.53%	89.92%	**9.88**
128	256	**50.38%**	**8.79%**	21.58%	90.44%	9.87
256	256	50.32%	8.75%	**21.54%**	**90.57%**	**9.88**

Table 4.2: User embedding size study on N.S LSTM

User embedding dimension affects does not affect performance metric HR as different neural-based architectures did, but it does play a significant role in the Overlap diversity metric where we see a direct correlation when increase from 1 to 128 dimensions with 256 being similar to 128. Such behavior is understandable as overlap measures user's unique set of recommendations. With worse user representations, the model will not produce the same number of unique sets per user.
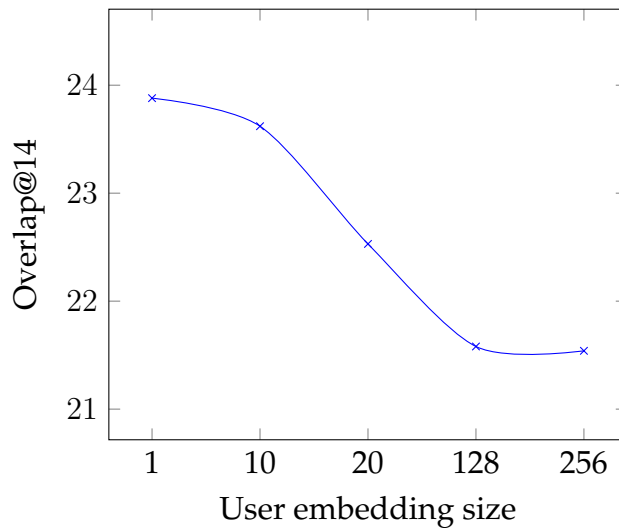

Figure 4.4: User embedding size length against Overlap@14

Overlap measures unique user recommendation sets, so the more information the model has about distinct users, the more unique the produced sets would be. For a second test, we changed the item embedding dimension:

User Size	Item Size	HR @14	Coverage @14	Overlap @14	Personalization @14	Novelty @14
256	1	35.54%	6.20%	22.80%	89.40%	9.8
256	10	44.32%	7.80%	22.52%	90.53%	9.82
256	20	46.75%	8.12%	22.34%	**90.76%**	**9.91**
256	128	**50.87%**	8.34%	**21.32%**	90.32%	9.8
256	256	50.32%	**8.75%**	21.54%	90.57%	9.88

Table 4.3: Item embedding size study on N.S LSTM

We now find a significant impact on the HR metric; there is a direct correlation between item embedding dimension HR. Performance drops to two-thirds when items have one dimension only. Diversity metrics do see a change as well, but now in Coverage, the metric measuring the percentage of items covered in the recommendations. This drop in Coverage performance can be explained as we are making it harder to learn item representations to the model, making it harder to distinguish them.

4.3.3   Impact of sequence order

One of the main hypotheses for this thesis has been to transition our recommendation model at H&M to one that can capture time relations. After seeing an improvement in results, it is clear that the new neural-based models translate in better performance, but is it because of their ability to capture time and order relations? In this experiment, we will randomize the order of the training sequences and keep a consistent test set with the other experiments. Suppose model performance would stay similar to the original experiments. In that case, this means it is not the time and order relations but the raw neural-based model's performance responsible for the increase in performance.

To study it, we tried three different orders within the user history on the N.S LSTM.

- **Correct order**: The original order in which interactions were made.

- **Day level order**: Randomize interactions within the same day, meaning that two interactions will only be correctly ordered in the user history sequence if they happened in different days.

- **Random**: User history order is completely randomized.

User history Sequence order	HR @14	Coverage @14	Overlap @14	Personalization @14	Novelty @14
Correct order	**50.32%**	**8.75%**	**21.54%**	**90.57%**	**9.88**
Day level order	49.37%	8.04%	23.39%	90.46%	9.55
Random	43.37%	7.73%	22.89%	85.41%	8.89

Table 4.4: User history sequence order study on N.S LSTM

User history sequence order does play an essential role in our recommendation model, affecting HR performance when having a total randomized order and with almost similar performance with a half randomize logic as the day level ordering. We also see a correlation between order and diversity metrics, with the correct order obtaining the best diversity metrics and dropping as we increase randomness in the history order.

4.3.4   Impact of user history length

How far back should we look in a user history is a big question when offering recommendations. Pasts experiments have been done with a max length of user history of twenty interactions. In this experiment, we evaluate different history lengths and how they affect our model's performance. We tried different values on the N.S LSTM and obtained the following results.

User history Sequence order	HR @14	Coverage @14	Overlap @14	Personalization @14	Novelty @14
5	45.00%	**8.77%**	25.19%	92.05%	9.41
10	48.00%	8.61%	26.60%	89.50%	9.25
Default value - 20	50.32%	8.75%	21.54%	90.57%	**9.88**
50	**50.35%**	7.78%	**16.76%**	**94.60%**	9.69

Table 4.5: User history sequence length study on N.S LSTM

We see a direct correlation between the max user history length and HR, the more we look back in the user history, the better our performance. We also see a major diversity impact over history length, especially on the overlap metric; reaching AALS performance levels. The model can capture a more personalized representation of the history and offer a unique set of recommendations.

4.3.5   Recommended items features study

With performance or diversity metrics, we get a generalist performance evaluation, which is good enough for most of the use cases. Studying our model recommended items features provides us with a use case-specific analysis on what we recommend. For H&M we have different item attributes we look at. The first analysis is how popular are the items our models recommend. We picked our baseline model and the best performing Neural Sequencer and evaluated their popularity on the top fourteen recommended items.
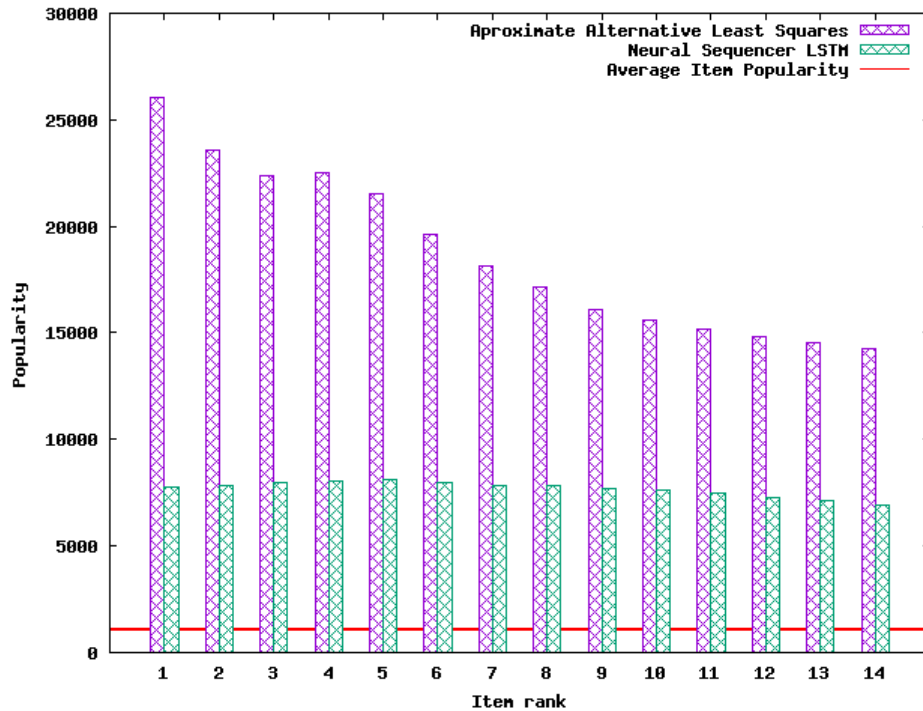
Figure 4.5: Average item popularity in top 14 recommendations.

Both models recommend significantly higher popular items than the average item popularity. Our baseline model is highly skewed towards high popularity items and decreases popularity as the item recommendation rank increases. The neural Sequencer model popularity stays consistent over the fourteen recommendations and stays lower than the baseline across all ranks.

Achieving good performance metrics should be tied to recommending more popular items, after all, the higher popularity, the higher chance of being sold. With this analysis we see that it is not a direct correlation and in fact the new neural-based models achieve better performance while recommending less popular items than the baseline. New models learn a better representation of the user and given their history can provide popular items suited for their profile.

Secondly, we analyze the average price on recommended items, same models, and top fourteen recommended items.
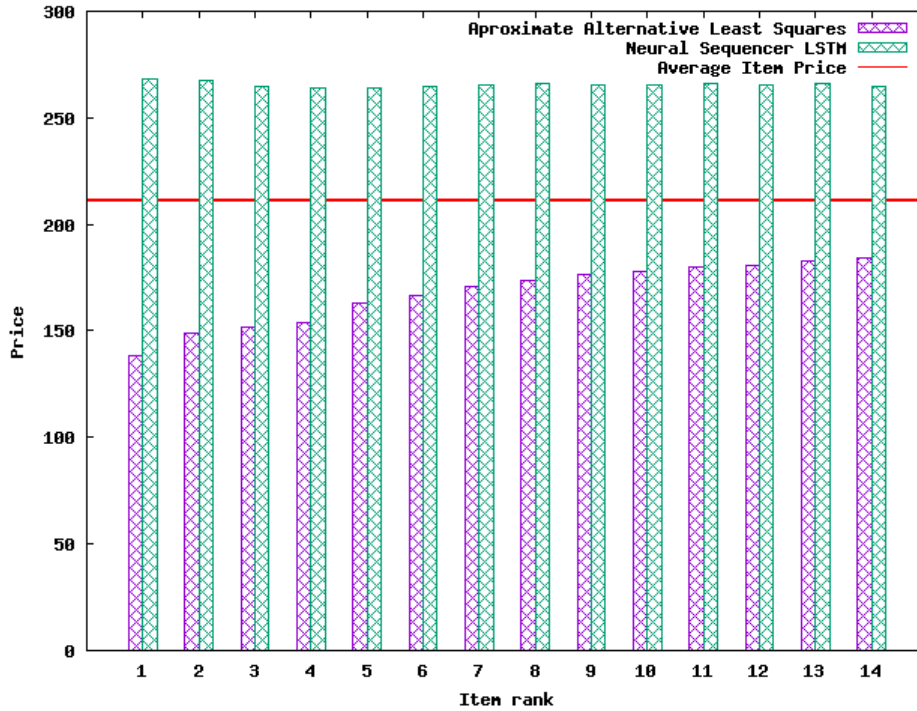
Figure 4.6: Average item price in top 14 recommendations.

Our model's performance have opposite behaviors when comparing with the average item price. The baseline model recommends lower price items, while the Neural Sequencer recommends higher price items. Neural sequencer produces almost twice the price recommendations while having a better hit ratio, potentially providing higher revenue. We can connect the findings with the past analysis on item popularity. Here at H&M, the best-selling products are basic clothing items such as white t-shirts or white packs of socks, aggressively priced items at shallow margins. For our last analysis, we decided to evaluate the item age of the top fourteen recommended items.
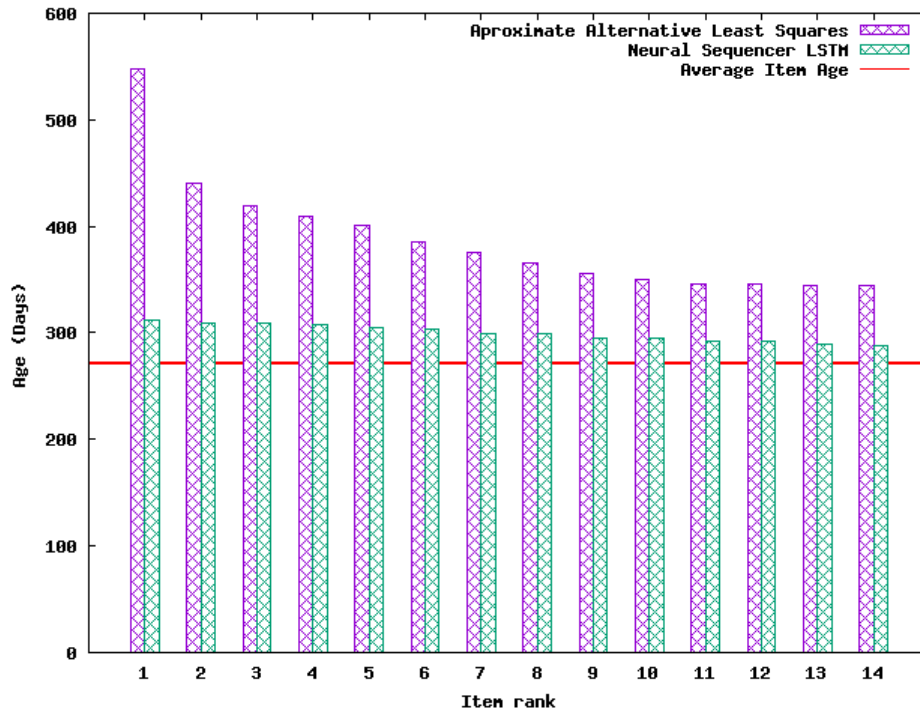
Figure 4.7: Average item ages in top 14 recommendations.

Both models present a skew towards older items as the higher the
rank they give an item, but in totally different scales. The baseline
model presents its top tanked item with twice the age as the average
item age, while the top ranked Neural Sequencer item only has an 10%
increase in age. The more basic and popular items with aggressive
prices are usually sold across all seasons, making them older than the
on-season products.

All three analyses have confirmed our initial findings with the di-
versity metrics evaluation. The baseline model has a reduced set of
recommended items and provides a less personalized set of recom-
mendations than the Neural Sequencer, proved by its skew towards
low-priced and popular items. We now have a business explanation
on how those differences in diversity metrics are translated in offered
recommendations.

Chapter 5

Conclusion

5.1   Discussion

The initial objective for this thesis was to explore new models that could capture and exploit the time relations of H&M use case to provide better recommendations than the current collaborative filtering methods. Results have shown that neural-based methods can increase both performance metrics and diversity metrics compared to our AALS methods.

We have proved the neural-based models benefit from ordering user-item interactions with our ordering experiment, with improvements in performance metrics and diversity metrics as the model will have a more accurate representation of the user. Experiments with the max user history length fed to the model supported those findings and proved the importance of a good user profile representation through its history.

We also found explainability results for our model when stress testing neural-based models with their embeddings sizes. We learned how user and item embeddings play different roles in the performance of the recommendations and can be subject to multiple optimizations. User embeddings play a bigger role in diversity metrics and do not help with performance metrics. Item embeddings are the main driver for performance metrics, and the diversity metric is more tied to items, Coverage.

Moreover, we failed to follow the path of NLP and Deep Learning improvements. We obtained our best performing models with recurrent networks with LSTM and GRU cells. We were unsuccessful

in following the improvement's path with the usage of transformers methods as Alibaba did [23], [50].

The thesis has served to initialize exploration of neural-based models at H&M, providing an initial overview of how NLP methodologies can be translated to recommendations while evaluating important factors to user engagement as diversity metrics that are extremely important from the business perspective.

There are multiple open paths to continue this work which will be treated in the future work section.

5.2 Limitations

The first limitation we found was to follow improvements in NLP models for our neural-based recommendation models. Transformers are known for requiring more fine-tuning than the methods which proved more success in our experiments. We are not sure if our results could not follow the NLP path due to our implementation or the change of use case. While both cases have a vocabulary (words or items) NLP has much harder interrelation between the words than clothes do at H&M. It is the addition of the user profile that makes our case drift to a new scenario where item relations take meaning inside a user context.

We mentioned how we choose approximate neighbor methods for ALS to reduce computational time. Neural-based models achieved training within one day with enough GPUs but presented a new paradigm for industrializing the models. Moving to GPU-based environments needs work, and finding people with experience on them is not as easy as with the CPU environment needed for AALS. We were surprised that GPU cloud computing costs for our models were lower than the CPU ones. We needed fewer (yet more powerful) GPU machines than CPU machines.

5.3 Future work

We opened multiple work paths with this thesis, we now list what we see as the most relevant ways to continue the work at H&M both from the academic and industrial perspectives.

- **Recommendations longevity**: how our model's performance degrades over time and how it plays a role in such a fast phase industry. H&M is a fast-fashion company, meaning that the items in stock are often replaced between seasons and do not conserve their value as they would do in other companies that aim for timeless trends or quality over looks. It is commonly known that reputable mountaineering brand's items would last and stay relevant for years, but that is the opposite of what customers at H&M look for. There is a lot of research on recommendation use cases where items have long longevity, for example, the movie recommendations at Netflix. There is a clear research line on how the longevity of recommended items plays a role in recommendation systems.

- **Transfer learning**: Transfer learning has played an important role for NLP in recent years [51]. The increase in available training data and the search for reducing computational time have driven transfer learning research to a point where a lot of times, big models as BERT or ROBERTA only required little fine-tuning to specific use cases to achieve good performance. This is something to explore at H&M as we have most of the world markets data available. The thesis project was only focused on a small market as Sweden. Can we use cross-market data to train generalist models and fine-tune them to specific markets? Can we reuse trained models within a season with little fine-tuning instead of starting from scratch every time we update the models? There could be a lot of work on finding the correct setup to use cross-market data and improve model reusability to increase both the model's performance and reduce computational cost.

- **Split representation learning and recommendation model**: Right now, the explored neural-based models optimize two problems simultaneously. They learn user and item representations while optimizing neural layer weights. Splitting these two optimizations problems could translate into better performance for both scenarios and provide a warm start for a final joint model. Such behavior has been seen with pre-trained word embeddings for machine translation models [52]. We have previous work at H&M [53] where we learn items representations from their images and context to find items to compose an outfit. Can such representa-

tions be used as a warm start for the recommendation models?

- **Join collaborative methods with deep learning learned representations**: Neural-based models have shown to improve our performance, but it is not clear whether the benefits come from the improved representations or the raw power of neural networks to learn distributions from data patterns. It would be interesting to explore if we can translate the learned representations to the latent space used for collaborative filtering and known neighbor methods.

- **Deep interactions between features**: Current models use the ID from user and items to learn their representation. Work at Microsoft [54] and Alibaba [23] have shown how interactions from user and items to features can result in improved performance. Replacing ID embeddings for item/user feature embeddings such as age, region, category, and color could benefit us with better performance and solve cold start problems where we could represent new items/users with only their attributes. Learning cross-feature relations such as user location and clothing colors can report benefits for the model.

The thesis has only been the beginning for H&M on exploring deep learning methods applied to recommendation models. We want to continue working along with the mentioned points and provide a new perspective on how these methods can be applied for the fashion industry.

5.4 Conclusion

This thesis shows how NLP Deep Learning methods, trained on user interactions sequences at H&M website, can be used to model user behavior and create personalized recommendations. We performed multiple experiments to prove how an ordered user history helps the model learn. Both item and user representations proved themselves to play an important role in our model's performance. New models won at performance but also saw different patterns on recommended items, recommending less popular items, and more expensive than our baseline model.

We believe it is the powerful representation learning and the ability to capture order within sequences that are responsible for the performance improvements. Multiple lines of work have been opened, the usage of cross market data could increase the learned representations as well as helping the model generalize. The removal of user identifiers is an interesting work as well, replacing them with user features instead could remove the cold-start problem and add more information to the model.

The fashion industry presents a new paradigm for recommender systems, high fast-phased trends and user-brand engagement. Companies in fashion do not only have the objective to recommend items users will buy, but also promote a trendy-fashion image of themselves to engage with the user and project the image of fashion leading company. Fashion companies often compete between themselves to create new trends and increase their customer group, metrics promoting diversity and better user engagement like the ones presented in this thesis will help compete in the market.

To sum up, our work has taken the research of companies like Alibaba, Amazon, and Google to the second-biggest worldwide fashion retailer and proved it useful. We have a new niche were our item's life frequency is much lower than theirs, and it is our duty to continue the work on researching recommender systems capable of learning high fast-phased trends.

Bibliography

[1] Netflix. "Netflix prize." (2007), [Online]. Available: `https://www.netflixprize.com/` (visited on 07/19/2021).

[2] "2018 reform of eu data protection rules," European Commission. (May 25, 2018), [Online]. Available: `https://eur-lex.europa.eu/eli/reg/2018/1725/oj` (visited on 09/01/2021).

[3] D. Kluver and J. A. Konstan, "Evaluating recommender behavior for new users," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14, Foster City, Silicon Valley, California, USA: Association for Computing Machinery, 2014, pp. 121–128, ISBN: 9781450326681. DOI: `10.1145/2645710.2645742`. [Online]. Available: `https://doi.org/10.1145/2645710.2645742`.

[4] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, Nov. 2002. DOI: `10.1023/A:1021240730564`.

[5] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," *ACM SIGIR Forum*, vol. 51, pp. 227–234, Aug. 2017. DOI: `10.1145/3130348.3130372`.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009. DOI: `10.1109/MC.2009.263`.

[7] D. Agarwal and B.-C. Chen, "Regression-based latent factor models," Jan. 2009, pp. 19–28. DOI: `10.1145/1557019.1557029`.

[8] S. Funk. "Netflix prize." (2006), [Online]. Available: `https://sifter.org/~simon/journal/20061211.html` (visited on 07/19/2021).

[9]   J. Cao, H. Hu, T. Luo, *et al.*, "Distributed design and implementation of svd++ algorithm for e-commerce personalized recommender system," in *Embedded System Technology*, X. Zhang, Z. Wu, and X. Sha, Eds., Singapore: Springer Singapore, 2015, pp. 30–44, ISBN: 978-981-10-0421-6.

[10] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," Jun. 2008, pp. 337–348, ISBN: 978-3-540-68865-5. DOI: `10.1007/978-3-540-68880-8_32`.

[11] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005. DOI: `10.1109/TKDE.2005.99`.

[12] H. Wang, N. Wang, and D.-Y. Yeung, *Collaborative deep learning for recommender systems*, 2015. arXiv: `1409.2944 [cs.LG]`.

[13] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-n recommendation," *CoRR*, vol. abs/1811.04392, 2018. arXiv: `1811.04392`. [Online]. Available: `http://arxiv.org/abs/1811.04392`.

[14] A. Galassi, M. Lippi, and P. Torroni, "Attention, please! A critical review of neural attention models in natural language processing," *CoRR*, vol. abs/1902.02181, 2019. arXiv: `1902.02181`. [Online]. Available: `http://arxiv.org/abs/1902.02181`.

[15] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in Jan. 2011, pp. 73–105. DOI: `10.1007/978-0-387-85820-3_3`.

[16] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[17] A. Rajaraman and J. D. Ullman, "Data mining," in *Mining of Massive Datasets*. Cambridge University Press, 2011, pp. 1–17. DOI: `10.1017/CBO9781139058452.002`.

[18] K. Ma, "Content-based recommender system for movie website," Ph.D. dissertation, 2016. [Online]. Available: `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-188494`.

[19] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops, "Learning word embeddings from wikipedia for content-based recommender systems," in *Advances in Information Retrieval*, N. Ferro, F. Crestani, M.-F. Moens, *et al.*, Eds., Cham: Springer International Publishing, 2016, pp. 729–734, ISBN: 978-3-319-30671-1.

[20] C. Hansen, C. Hansen, L. Maystre, *et al.*, "Contextual and sequential user embeddings for large-scale music recommendation," Sep. 2020, pp. 53–62. DOI: `10.1145/3383313.3412248`.

[21] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16, Boston, Massachusetts, USA: Association for Computing Machinery, 2016, pp. 191–198, ISBN: 9781450340359. DOI: `10.1145/2959100.2959190`. [Online]. Available: `https://doi.org/10.1145/2959100.2959190`.

[22] A. Beutel, P. Covington, S. Jain, *et al.*, "Latent cross: Making use of context in recurrent recommender systems," in *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[23] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, ser. DLP-KDD '19, Anchorage, Alaska: Association for Computing Machinery, 2019, ISBN: 9781450367837. DOI: `10.1145/3326937.3341261`. [Online]. Available: `https://doi.org/10.1145/3326937.3341261`.

[24] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.

[25] J. Yacim and D. Boshoff, "Impact of artificial neural networks training algorithms on accurate prediction of property values," *Journal of Real Estate Research*, vol. 40, pp. 375–418, Nov. 2018. DOI: `10.1080/10835547.2018.12091505`.

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014. arXiv: `1404.7828`. [Online]. Available: `http://arxiv.org/abs/1404.7828`.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 807–814, ISBN: 9781605589077.

[28] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[29] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[30] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. Beijing: O'Reilly, 2017, ISBN: 978-1-4919-1425-0. [Online]. Available: `https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/`.

[31] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, *A novel connectionist system for unconstrained handwriting recognition*, 2008.

[32] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014. arXiv: `1402.1128`. [Online]. Available: `http://arxiv.org/abs/1402.1128`.

[33] D. Lopez, *Recurrent neural network (rnn), long-short term memory (lstm) & gated recurrent unit (gru)*, Sep. 2015. [Online]. Available: `http://dprogrammer.org/rnn-lstm-gru`.

[34] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. eprint: `https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf`. [Online]. Available: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[35] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. arXiv: `1406.1078`. [Online]. Available: `http://arxiv.org/abs/1406.1078`.

[36] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, 850–855 vol.2. DOI: `10.1049/cp:19991218`.

[37] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 92–102, Apr. 2018, ISSN: 2471-285X. DOI: `10.1109/tetci.2017.2762739`. [Online]. Available: `http://dx.doi.org/10.1109/TETCI.2017.2762739`.

[38] R. Nallapati, B. Xiang, and B. Zhou, "Sequence-to-sequence rnns for text summarization," *CoRR*, vol. abs/1602.06023, 2016. arXiv: `1602.06023`. [Online]. Available: `http://arxiv.org/abs/1602.06023`.

[39] F. Jonsson, "Evaluation of the transformer model for abstractive text summarization," M.S. thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2019, p. 38.

[40] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. arXiv: `1706.03762`. [Online]. Available: `http://arxiv.org/abs/1706.03762`.

[41] D. E. Rumelhart and J. L. Mcclelland, "Parallel distributed processing: Explorations in the microstructure of cognition. volume 1. foundations," Jan. 1986. [Online]. Available: `https://www.osti.gov/biblio/5838709`.

[42] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[43] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990, ISSN: 0364-0213. DOI: `https://doi.org/10.1016/0364-0213(90)90002-E`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/036402139090002E`.

[44] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient estimation of word representations in vector space," Jan. 2013, pp. 1–12.

[45] S. Vargas, "Novelty and diversity enhancement and evaluation in recommender systems and information retrieval," in *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '14, Gold Coast, Queensland, Australia: Association for Computing Machinery, 2014, p. 1281, ISBN: 9781450322577. DOI: `10.1145/2600428.2610382`. [Online]. Available: `https://doi.org/10.1145/2600428.2610382`.

[46] N. Jones and P. Pu, "User technology adoption issues in recommender systems," Jan. 2007.

[47] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98, Dallas, Texas, USA: Association for Computing Machinery, 1998, pp. 604–613, ISBN: 0897919629. DOI: `10.1145/276698.276876`. [Online]. Available: `https://doi.org/10.1145/276698.276876`.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: `http://arxiv.org/abs/1412.6980`.

[49] L. N. Smith and N. Topin, "Super-convergence: Very fast training of residual networks using large learning rates," *CoRR*, vol. abs/1708.07120, 2017. arXiv: `1708.07120`. [Online]. Available: `http://arxiv.org/abs/1708.07120`.

[50] F. Sun, J. Liu, J. Wu, *et al.*, *Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer*, 2019. arXiv: `1904.06690 [cs.IR]`.

[51] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019, pp. 15–18.

[52] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, Jan. 2014, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`.

[53] G. Deligiorgis, "Context-aware graph convolutional network with multi-clusters mini-batch for link prediction," M.S. thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2020, p. 78.

[54] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "Xdeepfm," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018. DOI: `10.1145/3219819.3220023`. [Online]. Available: `http://dx.doi.org/10.1145/3219819.3220023`.