Resume

The work presented in *"Empirical Ablation Study of the Session-Based Graph Neural Network Model"* is a seventh semester project, aiming to increase the efficiency of the state of the art model presented in *"Session-based Recommendation with Graph Neural Networks"* (SR-GNN) by Wu et. al in order to improve scalability. This was achieved by conducting an ablation study on the SR-GNN model, where the model was modified by removing components to get a better idea of the behaviour and negligible parts of the model.

To create a foundation of how session-based recommendation systems work and what purpose they serve, already existing approaches were investigated. This is followed by a walk-through of the components in the SR-GNN in order to get a better understating of the model, before conducting the ablation study. Following the analysis of the components, a complexity analysis is performed to understand the complexity of the SR-GNN model, which is done to provide a more theoretical assessment of the run-time complexity of the model in its entirety, as well as the components.

After the groundwork was done, the work hypothesised how a model with the components removed would be more scalable compared to the original SR-GNN model. To test this, two datasets Yoochoose and Diginetica were used. In addition to this, several baseline models were implemented and tested against the SR-GNN model in order to see how it performs against other ways to implement session recommender systems.

The conduction of the experiments was done in four phases, where the first phase was meant to verify if the framework works as intended and was achieved by comparing results from the original work. Followed by the verification, an ablation study on the Gated Recurrent Units network was initiated, which extended the SR-GNN model into eight new models where the reset-gate, update-gate and hidden-state was either assigned weights or not. After finding the optimal GRU network, the GRU activations of the model were considered in a similar fashion to the second phase. Especially at this stage of the process the divergence of the run-times of the varying models were apparent, as the other evaluation metrics were similar. The final experiment was focused on scalability, where the improved scalability of the modified model was tested between the original SR-GNN model and an improved version.

The experiments were then discussed and concluded, where the discussion revolved around potential problems and errors of the implementation and construction of the experiments. An important part of the discussion include how the implementation of some of the original models were not published and how this forced some assumptions regarding the implementation to be made. Specifically there were not a clear formulation of the average pooling, however it was possible to re-create similar results following experimentation. Additionally tendencies and certain flaws of the datasets were addressed.. The work concludes that the connection matrix and the GRU part of the SR-GNN model, does not have a clear benefit to the model's overall performance, and concludes how the new model is an improvement to the SR-GNN model.

Empirical Ablation Study of the Session-Based Graph Neural Network Model

Roni Horo*, Mads H. Kusk*, Jeppe J. Holt*, Bjarke S. Baltzersen*,
Milad Samim* and Jamie B. Pedersen*
Department of Computer Science, Aalborg University, Denmark
*{rhoro18, mkusk18, jholt18, bbaltz18, msamim18, jjbp18}@student.aau.dk

December 21, 2021

Abstract

Since the introduction of Recommendation Systems, researchers have worked on making Recommendation Systems more efficient and more reliable. Recommendation systems have over the past decade become increasingly relevant with the rise of e-commerce and the increasing amount of data collected, scalable solutions are highly desirable. In recent years, researchers have focused on user interaction in Session-Based Recommendation Systems. *"Session-based Recommendation with Graph Neural Networks"* by Wu et al. [1] explores ways to efficiently obtain accurate user vectors by utilizing session sequences as graph-structured data. Further, the authors explore ways to optimize their initial model with an ablation study, where components are studied in the attention mechanism. This work extends this study to the whole network in *"Improving the Scalability of Session-Based Recommendation Systems"*, where the effects of various components throughout the GRU network are studied through a series of experiments with the goal of increasing the original model's scalability. The paper shows empirical evidence that the proposed method outperforms the original SR-GNN model.

1 Introduction

With the increasing amount of products and the rise of web services, it is becoming increasingly difficult to find appropriate recommendations. In order to combat this issue recommendation systems are becoming progressively more important. When given a sufficient amount of data, a recommendation system can find correlations between users and items to recommend items of potential interest to users. One branch of recommendation systems is session-based recommendation systems, which provides a method for predicting the next item clicked in a sequence of clicks from a user session. Therefore the recommendations are based on the items clicked in the current session and data from previous users' sessions. Further, session-based recommendation systems are useful for providing relevant recommendations in anonymous user sessions by considering only actions performed in the current session.

The model used as the underlying base model in this paper is the SR-GNN model by Wu et al.[1] in *"Session-based Recommendation with Graph Neural Networks"*. This model is chosen as it is shown to outperform several other state of the art models in this field. Another source of inspiration is the paper *"LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation"* by He et al.[2], where a study is conducted to find the key beneficial components in a model, and emphasize them to create a lighter but performant model, while discarding the negligible components. This type of study conducted is an ablation study, and is a process where certain parts of the model are removed to gain a better understanding of the models' behaviour. This process makes it possible to discover negligible components in a model, which then can be removed. One concern they had with Graph Convolution Networks(GCN) was the scalability, but by removing the unnecessary components from the model, the scalability and performance improved in [2].

In the work by Wu et al.[1] an ablation study was already conducted in order to create the model, but this paper aims to take this study one step further. This will be done to expand on and gain a better understanding of the individual components comprising the model. As well as removing components, without causing a detrimental effect on the evaluation metrics to maintain the usability of the model, while improving the scalability. The same datasets *Yoochoose* and *Diginetica*, used in the original paper, are also

utilized in this paper, as well as the evaluation metrics, precision at K(P@K) and Mean Reciprocal Rank(MRR) to maintain some consistency in an attempt at replicating their initial ablation study. This provides a way to gain a better understanding of which parts of the model were redundant for the evaluation metrics or negatively impacted the scalability of the model. Finally, the ablation study revealed that several components were without impact on the evaluation metrics and thereby the usability of the model. Instead, a model with similar usability and better scalability is presented. The code can be found on Github.[1]

Before the experiments are conducted, an exploration of related works utilizing session-based recommendation systems is presented, followed up by an elaboration of the SR-GNN model. This will lay the foundation on which the ablation study was conducted.

2 Related Work

In the following section, an overview of related works regarding session-based recommendation systems will be explored. In the initial part, a few conventional recommendation methods will be briefly elaborated, followed by some relevant deep-learning-based methods. Finally, a few graph-based neural network(NN) methods will be explored.

2.1 Conventional Recommendation Methods

Sarwar et al. (2001) present item-KNN in *"Item-based collaborative filtering recommendation algorithms"*[3] and this neighbourhood based algorithm shows how item-item predictions can be used to outperform user-user models. In the same session, item similarities are calculated using co-occurrence[2]. This model only considers the previously clicked item, when making a prediction.

Following are two sequential models based on Markov chains, the first one by Shani et al. (2002) who presents *"An MDP-Based Recommender System"*[4]. This work utilizes a different approach to session-based recommendation, namely Markov Decision Processes(MDPs), which is a process for optimizing sequential decisions step by step and views the problem as a sequential optimization problem. The other one by Rendle et al. (2010) *"Factorizing Personalized Markov Chains for Next-Basket Recommendation"*[5] (FPMC) concludes the model achieves better prediction for each sequence by using factorization of personalized probability transition matrices of the users. Markov chain based models assume independence i.e. they combine past components independently, which limits the performance. This is opposed to the inclusion of all sequences of user selections which have better performance, but results in worse scalability.

2.2 Deep-learning based methods

Hidasi et al. (2016-2018) presented two papers. The original, *"Session-based Recommendations with Recurrent Neural Networks"*[6], is the first time a recurrent neural network (RNN) was used in recommendation systems. The second paper, *"Recurrent Neural Networks with Top-k Gains for Session-based Recommendations"*[7] (GRU4Rec), is a follow-up paper aiming to improve the original. This method utilizes Gated Recurrent Units (GRU) which is also used by SR-GNN.

Two techniques to improve session-based recommendation systems are introduced by Tan et al. (2016) in the paper *"Improved Recurrent Neural Networks for Session-based Recommendations"*[8]. The first is data augmentation and the second is a method for taking temporal shifts in the input data distribution into account, since items only appear in sessions after they are released and user preferences may also change over time. This means old data could potentially decrease the performance of the recommendation system. These techniques can be used by other recurrent models to improve their performance as well.

In *"When Recurrent Neural Networks meet the Neighbourhood for Session-Based Recommendation"*[9] Jannach et al. (2017) presents an empirical evaluation showing a heuristics-based nearest neighbour (kNN). This scheme is shown to outperform GRU4Rec in most cases while achieving the best results when combining the recurrent method of GRU4Rec with the kNN method. The work also proves RNNs can utilize the sequential signals in data not detected by the kNN method.

The next paper is *"Neural Attentive Session-based Recommendation"* (NARM) by Li et al. (2017). NARM extends GRU4Rec by proposing a hybrid encoder, which captures the user's sequential behaviour and their main purpose in a session. To accomplish this, an attention mechanism is utilized in the encoder.

"STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation" (2018) by Liu et al.[10] does not utilize RNN, but instead, a short-term attention priority model using multilayer perceptron (MLP) networks[3]. In addition to this, an attention mechanism on the embedding of the last click is utilized, thereby enabling the model to capture both a users general interest and current interest.

[1] https://github.com/Biksbois/BiksUp

[2] How frequent two or more items appear together in the data.

[3] MLP is a feedforward NN, wherein the layers are fully connected

2.3 Neural networks on graphs

Following is the third section presenting the related works which utilize NN on graphs. *"Gated Graph Sequence Neural Networks"* (2015) by Li et al.[11] utilizes the work *"The Graph Neural Network Model"* (2009) by Scarselli et al. as a baseline[12]. This work is modified by utilizing a GRU network and back-propagation through time (BPTT).

Some inspiration for this paper is based on the work by He et al. in *"LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation"* (2020)[2] where an ablation study is conducted. In this ablation study, it is concluded some components in the Graph Convolution Network (GCN) have no effect on the evaluation metrics of the model and only degrade the scalability. Based on these observations, the LightGCN model is proposed, only containing the essential components of a GCN for collaborative filtering.

Finally, in the work presented by Chen et al. (2020)[13] a Hybrid-order Gated Graph Neural Network (HGNN) used for session-based recommendation is proposed. This newer method called SR-HGNN outperforms SR-GNN. It has been observed that deep GNNs are negatively impacted by the over smoothing problem, occurring when different nodes' embedding become too similar. This occurs due to the aggregation and update function as well as the number of layers, potentially resulting in all nodes converging to the same value.[13][14][15] They attempt to solve this issue by learning higher-order representations, but since adding additional GNN layers potentially also cause over-smoothing, hybrid-order propagation is used instead of first-order propagation usually used in GNNs. By including a second-order propagation, the model can recognize more complex dependencies and avoid more trivial patterns. Hybrid-order propagation will obtain e.g. first and second orders, thereafter an attention mechanism determines each order's weight and establishes the hybrid-order item representations, resulting in the final item representations after several update steps. Hereafter the session representations are generated, containing the final item representations and used to make the recommendations. Notably, the optimal number of orders varies from dataset to dataset.[13] In their experiments section results are demonstrated from the comparison between SR-HGNN, SR-GNN and several other baseline models. The evaluation metrics used are P@10, P@20, MRR@10 and MRR@20. They show that SR-HGNN outperforms SR-GNN along with the other baseline models.[13]

3 SR-GNN

This section will present a walk-through of the SR-GNN model presented by Wu et al.[1]. This will be done in order to gain an understanding of all components used throughout the model and will form the prerequisite for the ablation study conducted in the latter part of this paper. The first part addressed is the input in the form of a session graph.

3.1 Session Graphs

A session graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ is a directed graph where each node $v_{s,i} \in \mathcal{V}_s$ represents an item and edges represents transitions from item to item, i.e. an edge is given by the previous item and the next clicked item as $(v_{s,i}, v_{s,i+1}) \in \mathcal{E}$. Each session graph has an associated connection matrix $A_s = [A_s^{in} : A_s^{out}] \in \mathbb{R}^{n \times 2n}$, where A_{in} and A_{out} are matrices representing the indegree and outdegree of a session graph. To further illustrate consider the following small example.

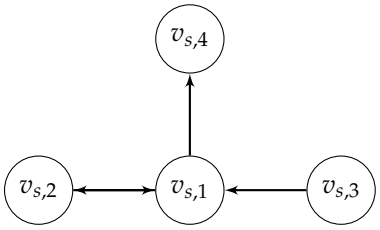

Figure 1: A Session graph with four nodes

The above graph illustrates a session graph of an anonymous session s with four distinct items. Associated with this graph is the following connection matrix.

	Indegree				**Outdegree**			
	$v_{s,1}$	$v_{s,2}$	$v_{s,3}$	$v_{s,4}$	$v_{s,1}$	$v_{s,2}$	$v_{s,3}$	$v_{s,4}$
$v_{s,1}$	0	1	1	0	0	1	0	1
$v_{s,2}$	1	0	0	0	1	0	0	0
$v_{s,3}$	0	0	0	0	1	0	0	0
$v_{s,4}$	1	0	0	0	0	0	0	0

Table 1: Indegree and Outdegree from the Graph presented in Figure **1**

The connection matrix A_s represents how nodes communicate with each other. Hence a row in A_s represents an item vector $A_{s,i} \in \mathbb{R}^{1 \times 2d}$ given by indegree and outdegree matrices. This item vector will be used as input to the GRU network after it has been initialized with weights and biases. Equation (1) expresses this according to Wu et al.[1], where the item vectors $A_{s,i}$ are initialized with weights and biases.

$$x_t = A_{s,i} : [\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}]^\top \mathbf{H} + \mathbf{b} \tag{1}$$

This next step is feeding the input into the GRU network.

3.2 Gated Recurrent Unit

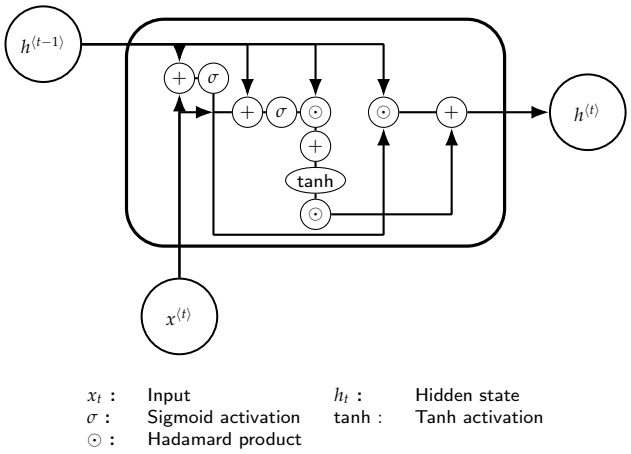

Figure 2: Illustrating the GRU order of operations

The above figure illustrates the workflow of a GRU network, where given an input and the previous iterations hidden state, one can follow each of the following steps visually in the network.

1. The first step is calculating the update gate z_t for time-step t.

$$z_t = \sigma\left(\mathcal{W}_z x_t + \mathcal{U}_z h_{t-1}\right) \tag{2}$$

   Where x_t is the input and h_{t-1} is the previous iteration's hidden state containing the memory for the network. The update gate is responsible for finding the relevant information from past iterations, to be passed on. This is achieved by normalizing the sum of the new input and the previous content with the sigmoid activation functions, which normalizes the result between 0 and 1.

2. The second step is calculating the reset gate r_t for time-step t.

$$r_t = \sigma\left(\mathcal{W}_r x_t + \mathcal{U}_r h_{t-1}\right) \tag{3}$$

   The reset gate is responsible for forgetting irrelevant information. This is done similarly to the update gate, with the use of a sigmoid activation function.

   Effectively sigmoid normalization approximates relevant information to 1, which maintains the information when it is multiplied and likewise, irrelevant information will approximate to 0 which the result also will when multiplied. The next two steps will elucidate further.

3. The third step is calculating the new hidden state h'_t which will hold the current memory for the network by using the result from the reset gate to eliminate information from the previous iteration.

$$h'_t = tanh\left(\mathcal{W}_o x_t + r_t \odot \left(\mathcal{U}_o h_{t-1}\right)\right) \tag{4}$$

   By performing a Hadamard product[4] between the reset gate and the previous hidden state, irrelevant information gets eliminated. The input x_t is added to this new pruned memory content and the result is normalized with the non-linear tanh activation function, which normalizes the result between -1 and 1.

4. The fourth and final step is producing the new hidden state h_t containing the information passed on in the network for later iterations.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{5}$$

   Here the result from the update gate is used to determine what information to keep from the current memory content h'_t and the previous memory content h_{t-1}.

3.3 Session Embedding

Running a series of session graphs through the steps in the GRU network will produce vectors for all nodes. The next step is to obtain the session embeddings from these vectors. A session embedding is a latent representation of a session in a d-dimensional space. In the work presented by Wu et al. three types of session embeddings are addressed: local, global and a combination of the two called hybrid. A local session embedding $s_l \in \mathbb{R}^d$ can be defined as the last clicked item, hence given a session $s = [\mathbf{v}_{s,1}, \ldots, \mathbf{v}_{s,n}]$ the local session is $s_l = \mathbf{v}_{s,n}$. The global session embedding s_g is calculated by aggregating over all the nodes in the session graph $\mathcal{G}_s$.

$$s_g = \sum_{i=1}^{n} = \alpha_i \mathbf{v}_i \tag{6}$$

Where α_i is an attention mechanism given by:

$$\alpha_i = \mathbf{q}^\top \sigma(\mathbf{W}_1 \mathbf{v}_n + \mathbf{W}_2 \mathbf{v}_i + \mathbf{c}) \tag{7}$$

When the global session s_g is calculated, the hybrid session embedding s_h can be calculated as a linear transformation over the concatenation of the local and

[4] A Hadamard product is an element-wise product

global session embedding.

$$s_h = \mathbf{W}_3[s_l : s_g] \tag{8}$$

Where $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$ is used to compress the two session embedding vectors into the latent space $\mathbb{R}^d$, as $[s_l : s_g] \in \mathbb{R}^{2d \times 1}$.

3.4 Predicting Recommendations

For each item $\mathbf{v_i} \in \mathcal{V}$ the score $\hat{\mathbf{z}}_i$ can be computed by the following:

$$\hat{\mathbf{z}}_i = s_x^\top \mathbf{v_i} \tag{9}$$

Where s_x is the session embedding described in the previous section, meaning x could either be the local, global or hybrid embedding computed. So the s_x is what is computed by forward propagating a session through the network up until Equation (8), next this session embedding is used in a matrix multiplication, consisting of all the item embeddings, to compute the score for each item for that specific session. In Equation (9) rather than denoting the matrix multiplication, the equation shows the score computation for a single item embedding v_i, through a dot product. When the scores are computed, the softmax function will be applied to obtain the output vector for the model $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \text{softmax}(\hat{\mathbf{z}}) \tag{10}$$

Here $\hat{\mathbf{z}} \in \mathbb{R}^m$ denotes the recommendation scores over all items and $\hat{\mathbf{y}} \in \mathbb{R}^m$ denotes the probability of all items being the next clicked item. As a ground truth for the sessions, a one-hot encoding of the items will be computed as $\mathbf{y}$ and used to compute the loss with the cross-entropy loss functions:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{m} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}) \tag{11}$$

The cross-entropy loss function is a loss function, used for a multi-class classification, which suits the task under consideration. Further, it sums over all the scores, and for each computes the cross-entropy, representing the number of bits the model needs to encode the probability and weighs it according to the underlying probability distribution, the one hot-encoding of the label. Hence it computes an error, which weighs all the model's proposed scores for each item. This will be computed for each session graph, during the training per epoch. Further, the training implements the stochastic gradient descent algorithm for learning with the ADAM optimizer and the BPTT algorithm.

4 Complexity Analysis

When evaluating scalability, that is the performance as a function of the input size, the empirical approach involves simulating the model on large datasets and then measuring the time it takes for it to execute. This analysis can be directed both towards the training phase and the prediction phase.

The prediction phase as mentioned in Section **3** consists of propagating the session through the model to get the session embedding and then doing a dot-product with the session embedding and a set of candidate items to compute the scores. As such, this dot-product depends primarily on two aspects, being the dimension of the two vectors $s_x, \mathbf{v_i} \in \mathbb{R}^d$ and the number of candidate items to compute a score for. Where d is the dimension of the hidden embedding of the items. Reducing the dimensionality of d is prone to induce a trade-off in terms of the P@20 and the MRR@20 scores. For the number of candidate items, it might make sense to consider only items inside certain categorical proximity, but that is out of the scope of this paper. For the training phase, one could also do empirical time measurements. However, the focus will primarily be on the training phase, with the aim of performing a time complexity analysis. To enable the analysis, the training algorithm will be presented in pseudocode, as can be seen in Algorithm **1**.

4.1 SR-GNN Time Complexity

Now assessing the algorithm reflecting the SR-GNN learning procedure, the primary input is a list of the session embeddings, with a length of $n = len(S)$. Further, each session has a length reflecting the number of clicked items in the sequence, say the average length is $k = \frac{1}{n} \cdot \sum_{s \in S} len(S)$. Besides the number of sessions n and the average session length k, the time complexity also depends on the number of unique items in the datasets. This is presented as m = number of unique items $\in S$ and will be the number of item embeddings. In the pseudocode an outer repeat on line 1 can be seen, which continues until the model training is done, typically until a number of epochs are finished or the loss has remained the same across some specified number of epochs. More interestingly, from line 5 an outer loop over all the sessions can be seen, hence the code from lines 5 to 15 will run n times. Here a connection matrix is constructed for each session, which is implemented through a loop, and assuming an average session length, it will require $k \cdot k$ operations for construction. The connection matrix $\mathbf{A_s}$ constructed will be $\mathbb{R}^{k \times 2d}$, where k will vary depending on the specific session length, and $2d$ represents the concatenation of the in

and out-degrees.

Before continuing the following assumption is made regarding matrix multiplication; given two matrices $\mathbf{P} \in \mathbb{R}^{R \times C}$ and $\mathbf{Q} \in \mathbb{R}^{C \times D}$, the complexity of $\mathbf{PQ}$ is assumed to be $\mathcal{O}(R \cdot C \cdot D)$.

Lines 7 to 9 cover the equations from Equation (1) to Equation (5) from Section **3**. The primary operation here is matrix multiplication, which from Equation (1) is deemed to be the matrix multiplication of the connection matrix $\mathbf{A_s}$ with $\mathbf{H} \in \mathbb{R}^{2d \times d}$ from Equation (1), running in $\mathcal{O}(k \cdot 2d^2)$. The next equations (2) to (5) regards the computation in the GRU cell, where the primary operation similarly is matrix multiplication. Here an observation is that all the weights share the same shape, as $\{W_x, U_x \in \mathbb{R}^{d \times d} | \text{ for } x \in \{z, r, o\}\}$, hence they reflect 6 matrix-vector multiplications with each item embedding $\mathbf{v^t_{s,i}} \in \mathbf{R}^d$ and the complexity of the matrix-vector multiplication can be analyzed according to the assumption by modelling the item embedding as being a matrix with 1 column vector $\mathbf{R}^{d \times 1}$. This will yield the complexity of $\mathcal{O}(6 \cdot k \cdot d^2)$, as there are 6 such matrix-vector multiplications for each item in the session. These steps can be done for T iterations, but in [1] $T = 1$ was the default. Now summarizing the complexity of these steps it results in $\mathcal{O}(k \cdot 2d^2 + 6 \cdot k \cdot d^2)$, here it is seen that these steps depend primarily on the session length and the dimensionality of the item embedding, further this is done n times but that cost will first be integrated at the end.

From here Algorithm **1** in line 10 applies Equation (6) and Equation (7) to compute the global embedding and further, where the primary operation again is matrix-vector multiplications with similar dimensionalities. Equation (6) and Equation (7) refers to the computation of the global session embedding, which requires the computation of k alpha factors α_i (one for each item in the session), which are computed by Equation (7). Equation (7) primarily uses two matrix-vector multiplications product with $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ and the item embedding as explained in Equation (7). Furthermore it will compute a dot product with $\mathbf{q} \in \mathbb{R}^d$. The matrix-vector multiplications and the dot-product will require $\mathcal{O}(d^2 + d)$. As an alpha factor is needed for each item as shown in Algorithm **1**, this is required for all k items in the sessions, which results in $\mathcal{O}(k \cdot (d^2 + d))$.

Line 11 relates to Equation (8), which is used to compute the hybrid session embedding, which requires a matrix-vector multiplication with $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$ and the concatenated vector $[s_l, s_g] \in \mathbb{R}^{2d}$, running in $\mathcal{O}(2d^2)$.

Line 12 refers to the score computations from Equations (9) and (10), here the computed session embedding s_x is multiplied with the matrix consisting of the item embedding of all the items, which is in $\mathbb{R}^{m \times d}$. That is exactly similar to computing the dot-product with the session embedding and then each item embedding, and hence will run in $\mathcal{O}(m \cdot d)$.

The last part is the computation of the cross-entropy loss function on line 13 given in Equation (11), computing the loss over all the candidate items, which during training is m, hence the complexity thereof will be $\mathcal{O}(m)$.

Considering the lines under the loop from line 5, the following big-O complexities are arrived at for each line referred to in Algorithm **1**.

- Line 8: $\mathcal{O}(k \cdot 2d^2 + 6 \cdot k \cdot d^2) = \mathcal{O}(8 \cdot k \cdot d^2)$
- Line 9: $\mathcal{O}(k \cdot (d^2 + d)) = \mathcal{O}(k \cdot d^2 + k \cdot d)$
- Line 10: $\mathcal{O}(2d^2)$
- Line 11: $\mathcal{O}(m \cdot d)$
- Line 12: $\mathcal{O}(m)$
- Lines 8 through 12: $\mathcal{O}(9 \cdot k \cdot d^2 + k \cdot d + 2d^2 + m \cdot d + m)$

Now, the computations referred to above has to be run for each session in the training dataset, that is n times, in aggregate this results in the big-O complexity of $\mathcal{O}(n(9 \cdot k \cdot d^2 + k \cdot d + 2d^2 + m \cdot d + m))$. From here one could drop the constants, and then further reduce the expression to only the largest term. Doing such one would arrive at $\mathcal{O}(n \cdot m)$, that is the number of sessions times the number of items. However, the full expression reflects the overall computation steps in the training algorithm.

From the complexity analysis, it is learned that the model's training is polynomial, and depends directly on the number of sessions used in training, the length of these sessions, and the chosen dimensionality of the item embeddings.

4.2 Time Complexity of Components

The SR-GNN architecture can be split into a set of distinct components, where each contributes certain computations and as such impacts the overall time complexity. This section aims to separate the computational impact of each of the core components. The core components are the GNN propagation steps, that is Equation (1) to Equation (5), the attention mechanism and lastly the score and loss computations. The GNN propagation steps can be split into the connection matrix and the remaining steps.

The construction of the connection matrix requires $\mathcal{O}(k^2)$ operations and further requires $\mathcal{O}(k \cdot 2d^2)$ computations as per Equation (1). Additionally, each of these computations is required for each session, meaning it will be computed n times. However, each of the components will have to be computed for each sample, hence this factor will not be considered. Dropping the connection matrix will have a negligible impact computationally, but will require a re-thinking of the graph part of the model, as it is the direct graph related embodiment in the model.

The remaining part of the GNN propagation steps contributes $\mathcal{O}(T(6 \cdot k \cdot d^2))$ (as default $T = 1$), which similarly does not reflect particularly expensive computations, that is it does not deviate from the other components distinctively.

The attention mechanism is computed through Line 9 in Algorithm **1** and runs in $\mathcal{O}(k \cdot (d^2 + d))$, which again reflects a similar computational load. The attention mechanism is mentioned as offering the ability to consider the current interest of the specific session by considering the items clicked in the session [1].

The last distinctive component is the computation of the scores and the loss as performed in Line 11 and 12, which runs in $\mathcal{O}(m \cdot d + m)$. Here the score and loss computations depend on the total number of items as well as the dimensionality of the item embedding. Since each item requires consideration it is distinctive from the remaining components and reflects the largest computational component in the model, especially when considering it needs to be computed for every sample. This depends directly on the distribution of the datasets, if $m > k \cdot d$, meaning there are more unique items than the average number of items per session times the dimensionality it will be the largest term. However, if it is less, it will not be the largest term. The distribution of clicks, number of sessions and unique items for the used datasets are available in Table **2**, but will first be presented in Section **5**, and there it follows, that it is the largest term. Considering improvements to this part is not immediately apparent, as computing the categorical cross-entropy loss for each item embedding, necessarily requires the consideration of each item embedding.

Algorithm 1: The training procedure of SR-GNN

Input : List of sessions S
Parameter: Number of GNN porpagation steps T
Output : Model parameters $\Phi = \{H, b, W_z, U_z, W_r, U_r, W_p, U_o, W1, W2, W3, q, c\}$

1 **repeat**
2 Let V be the list of all the items
3 Convert each item $\mathbf{v} \in V$ to it's item embedding
4 Initialize model parameters Φ
5 **for** *sessions* $s \in S$ **do**
6 $\mathbf{A_s} \leftarrow$ is the connection matrix for session s do something
7 **for** *each time step* $t \in T$ **do**
8 Apply Equations (1) to (5) to get the next representation of $\mathbf{v^t_{s,i}}$
9 **end**
10 Apply Equations (6) and (7) to get the global session embedding $\mathbf{s_g}$
11 Apply Equation (8) to get the hybrid item embedding $\mathbf{s_h}$
12 Compute the predictions through Equations (9) and (10)
13 Compute the loss through Equation (11)
14 Update the model parameters Φ
15 **end**
16 **until** *Loss function in Equation* (11) *has converged or some*
17 *other stopping criterion is met*;
18 **return** Model parameters Φ

5 Experimental Setup

This section will present the datasets used in the experiments, the baseline algorithms used to compare the performance of the algorithms and the evaluation metrics. Further, the various models in the ablation study will be presented and elaborated. The experiments will be run in three iterations on a system with the following hardware specifications: GPU: **NVIDIA GeForce RTX 3070 Ti**, CPU **Intel i7-11700KF**, RAM: **32 GB**.

5.1 Datasets

Statistics	*Yoochoose 1/64*	*Yoochoose 1/4*	*Diginetica*
# of clicks	2,621,822	34,097,026	3,466,487
# of training sessions	369,859	5,917,745	612,858
# of test sessions	55,898	55,898	51,751
# of items	16,766	29,618	39,028
Average length	6.16	5.71	5.22

Table 2: Statistics of the Yoochoose (1/64, 1/4) and the Diginetica dataset

The experiments will be evaluated on two datasets Yoochoose[5] and Diginetica[6]. The Yoochoose dataset contains a stream of user clicks from an online retailer, spanning over six months and was first presented at the RecSys Challenge event. The other dataset, Diginetica, is made up of user sessions from e-commerce search engine logs, including clicks and purchases. Diginetica was first presented at the CIKM Cup 2016 Track 2: Personalized E-Commerce Search Challenge at the CIKM conference. In the experiments only the released transactions data will be used. When conducting the experiments on the Yoochoose and Diginetica datasets, sessions of length 1, and items clicked less than 5 times will be filtered out. The reasoning behind applying these filters on the datasets are foremost that it enables a comparison with the paper by Wu et al.[1] Another reason for filtering out sessions with a length less than one, is since the label for a given session is generated by holding out the last clicked item in the session, thus at least 2 clicks are required. Given the size of Yoochoose, only 1/64 of the datasets are used in the experiments as in [1]. The statistics of both datasets are available in Table **2**.

5.2 Baseline methods

To create a baseline for comparison of the method introduced in this work, the following baseline algorithms are chosen.

- **POP:** Recommends the top-N most frequent events in the training set.
- **S-POP:** Recommends the top-N most frequent events in the current session.
- **Item-KNN:** (Sarwar et al. 2001)[3] Recommends top-K similar items based on previously clicked items in the session. Cosine similarity is used to decide which items to pick.
- **GRU4REC:** (Hidashi et al. 2018)[7] Uses RNN and GRU to model the session and make recommendations.
- **NARM:** (Li et al. 2017)[16] A RNN based method, with an attention mechanism to capture the main purpose and sequential behaviour of the user.
- **STAMP:** (Liu et al. 2018)[10] A short-term attention/memory priority model, which captures users interest from a session context, considering both long-term memory as well as short-term memory.
- **SR-GNN:** (Wu et al. 2018)[1] An implementation of a gated graph NN for session-based recommendations.

To run these algorithms the framework provided by Malte et al [17] will be utilized.

5.3 Evaluation metrics

Essential to the empirical experiments is a way of evaluating the outcome to track the usability of the model and this section will present the evaluation metrics utilized. Two different metrics will be used, the precision at K (P@K) and the Mean Reciprocal Rank (MRR). For the P@K measure, an important remark is that all the sessions are arranged in a leave one out arrangement[7], hence the precision can be at most 1/K, that is if the item is in the top k items. As such the P@K score is normalized through division by 1/K similar to Wu et al.[1]

1. **P@K:** Computes the percentage of the top K items, relevant to the query/session. As mentioned above, given the leave 1 out arrangement, the following description will reflect that circumstance. Specifically, the item embeddings are arranged in descending order when the softmax score are computed, where the top K items are considered and it is checked whether the actual next clicked item (the item left out) is in this list of Top K items. This is done for every test pair in the test set and the percentage of test pairs with

[5] www.kaggle.com/chadgostopp/recsys-challenge-2015
[6] www.competitions.codalab.org/competitions/11161

[7] Leave one out refers to that the last clicked item is removed and then used as the label to be predicted.

the next clicked item in the top K is computed as the P@K score for the model on the test set.

$$\textbf{P@K} = \frac{\text{K}}{|1|}\frac{1}{|Q|}\sum_{q\in Q}\frac{\text{\# of relevant items @k}}{\text{\# of recommended items @k}} \tag{12}$$

Where $|Q|$ is the total number of queries and $topK$ is the list of the top K items predicted for a session as described above. The Equation (12) computes the mean percentage of relevant items in top K across all queries in Q. Further, given users are not considered, any $q \in Q$ is not related to any specific users, but reflects sessions and in particular the sessions of the test set.

2. **Mean Reciprocal Rank:** The P@K is indifferent to the ranking of the items, which could in certain recommendation systems be of more critical importance. Here the MRR@K is more attractive, as it considers the ranking of the items. Specifically, it computes the mean of the reciprocal ranks. A rank beyond K is set to 0 as shown in Equation (13), as a piecewise function.

$$\textbf{MRR@K} = \frac{1}{|Q|}\sum_{i=1}^{|Q|} x \tag{13}$$
$$\text{if } rank_i < \text{K then } x = \frac{1}{rank_i} \text{ else } x = 0$$

Where Q is as described above, and $rank_i$ is the positional rank of the first relevant item in the query. Further x is the highest-ranking relevant item.

5.4 Ablation study

The ablation study will be performed in three phases of hypothesis testing.

1. The null hypothesis will test whether the testing framework is working as intended. This will be achieved by replicating the ablation study presented in the work by Wu et al.[1]. The expectation will be to see identical results.

2. The second phase is performing an ablation study of the various weights and bias throughout the GRU network. Here the best performing model from the null hypothesis experiment will be used.

3. The third phase is performing an ablation study of the different activation functions throughout the GRU network.

4. The last phase will perform an experiment to measure the scalability between the SR-GNN and the model deemed best in the third phase.

This testing framework will be elaborated further in the experiments.

5.5 Models

The outcome of the ablation study will be an analysis of how different components impacts the SR-GNN model. To analyse these components various models will be arranged, with a different set of components enabled. These models will now be elaborated. The models rather than being distinct models are variations of the baseline SR-GNN from the Wu et al. paper [1]. Thus a set of names each representing specific changes in the base model will be presented. The variations are in direct accordance with the aforementioned ablation study and can further be split into the three phases as well.

- SR-GNN: Is the exact base model from Wu et al. [1]
- SR-GNN-**L**: A model where the global session is disregarded and it only considers the local embedding, this being the item embedding of the last clicked item in the session.
- SR-GNN-**ATT**: Considers the global session embedding by the attention mechanism and disregards the idea of a distinct local embedding.
- SR-GNN-**AVG**: Considers the global session embedding through an average pooling mechanism, rather than learning an attention mechanism, hence intended to be a trade-off between prediction quality and efficiency. Disregards the idea of the local embedding.
- SR-GNN-**SUM**: Similar to the SR-GNN-**AVG**, but with sum pooling instead.
- SR-GNN-{**Weights**}: Here a set of models are evaluated, which considers the effect of the weights associated with the GRU circuit of the network. The names and the models will be presented in Section **6.2**.
- SR-GNN-{**Activation**}: Explores the impact of removing the activation functions in the GRU network. Similarly they will be presented later on.

The models listed above will compose the core part of the experiments.

6 Empirical Study

The goal of this paper is to explore how the SR-GNN model can be modified in order to improve scalability. This will be pursued through an empirical methodology, specifically by the use of an ablation

study. Each of the output models will be evaluations of certain components of the existing SR-GNN, which can be either removed, changed or adapted to enhance the scalability. These models will be trained and tested in the experimental setting elaborated in Section **5**, where the results will indicate the best model. This model will be compared with baseline algorithms, elaborated in Section **5.2**.

6.1 Testing the SR-GNN Framework

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
SR-GNN	70.60	30.59	5147.08	51.48	**18.03**	3578.08
Local	69.37	30.23	5730.84	48.13	16.27	5889.12
ATT	70.41	30.03	4831.20	**51.54**	17.79	**2750.22**
AVG	**70.80**	**30.86**	**4339.28**	51.35	17.87	3654.99
SUM	70.57	30.58	4401.47	51.27	17.78	3578.07

Table 3: Performance of the SR-GNN model compared with the models from this ablation study, the models are abbreviations from the models listed in Section **5.5**

This experiment is intended to test whether the testing framework is working as intended. The hypothesis revolves around the experiment performed by Wu et al. [1], where different session embeddings are tested. The models involved in this experiment are the following:

- SR-GNN
- SR-GNN-**AVG**
- SR-GNN-**ATT**
- SR-GNN-**L**
- SR-GNN-**SUM**

The expectation is a similar conclusion to what was drawn in this experiment conducted by Wu et al.[1] where the best model, in this case, was SR-GNN.

The results for the performance of the models are shown onTable **3**. This includes the evaluation metrics in terms of P@20 and MRR@20 but also the run-time of the training. The results indicate similar P@20 and MRR@20 scores across all models. Considering the run-time, three models seem to have an edge: SR-GNN-**ATT**, SR-GNN-**AVG** and SR-GNN. Since the SR-GNN model also utilizes attention and the models are quite similar, the following experiments will use SR-GNN as a baseline. This phase also shows that the results presented in this work, are more or less in line with the results presented by Wu et al. Therefore it is assumed the framework works as intended.

6.2 Experiments: Phase Two - GRU Weights

The previous experiment indicated the SR-GNN model to be best when considering performance and run-time. This experiment will utilize the SR-GNN model and study the effect of the GRU network weights in an ablation study. To do this, eight additional models will be created, each with a different combination of weights included and excluded. The models are as follows:

- SR-GNN-**NW**
- SR-GNN-**RW**
- SR-GNN-**HW**
- SR-GNN-**RHW**
- SR-GNN-**UW**
- SR-GNN-**RUW**
- SR-GNN-**UHW**
- SR-GNN

The abbreviations **R**, **U** and **H** refers to weights in the **R**eset-gate, **U**pdate-gate and the **H**idden-state (Hence SR-GNN-**RW** means only the reset gate uses weights and biases and SR-GNN-**RHW** means the reset gate and the hidden state uses weights and biases). Further the model SR-GNN-**NW** contains no weights or biases throughout the GRU network.

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
SR-GNN	70.49	30.59	4071.07	**51.26**	17.77	3081.98
RW	69.48	30.39	4810.90	50.63	17.05	3138.95
HW	70.05	30.33	3893.91	50.21	17.15	2990.15
RUW	69.58	30.01	4874.78	50.33	17.02	3576.43
UHW	**70.64**	**30.79**	**3662.37**	51.18	17.69	3072.47
UW	69.48	29.93	5043.96	50.49	17.00	3145.94
RHW	70.54	30.76	3814.92	51.05	**17.79**	3283.31
NW	69.65	30.26	4583.56	49.69	16.31	**2812.44**

Table 4: Performance of the SR-GNN model compared with the models from this ablation study, the models are abbreviations from the models listed in Section **5.5**

The results are shown on Table **4**. One observation from the results is regarding the insignificant impact the weights have on the evaluation metrics, but observing the run-time of the models, higher discrepancies are present. Observing each weight's impact shows models with the hidden-state weights are advantageous compared with models without, these are the SR-GNN-**HW**-, SR-GNN-**UHW**- and the SR-GNN-**RHW** model, where the SR-GNN-**UHW** model is the best performing on the Yoochoose 1/64 dataset.

In order to create a clearer representation of what impact each of the **R**, **U** and **H** components had on the model, an average score will be calculated. This score is an average of all the models where the given component is utilized. As an example **H** includes all the models containing the weights and bias for the hidden-state, these being SR-GNN-**HW**, SR-GNN-**UHW** and SR-GNN-**RHW**.

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
R	69.86	30.38	4500.20	50.57	17.28	3332.89
U	69.90	30.24	4527.03	50.66	17.24	3264.94
H	**70.41**	**30.63**	**3790.4**	**50.81**	**17.53**	**3115.31**
N	69.65	30.26	4583.56	49.69	16.31	**2812.44**

Table 5: The averaged results from experiment 2

From Table **5** it becomes much clearer which weights were present in the best performing models since the **H** models performed best on each of the measured metrics. Thus, the model chosen to be continued with was SR-GNN-**UHW**, because it performed the best out of the three models including the hidden state weights.

The following experiment will utilize the SR-GNN-**UHW** as a baseline.

6.3 Experiments: Phase Three - GRU Activations

The previous experiments showed the impact GRU weights had on the SR-GNN model and indicated the model SR-GNN-**UHW** was the best performing model. This experiment will test the impact of the GRU activation functions similarly to the previously conducted experiment. The models for this experiment are listed below:

- SR-GNN-**NA**
- SR-GNN-**HA**
- SR-GNN-**UA**
- SR-GNN-**UHA**
- SR-GNN-**RA**
- SR-GNN-**RHA**
- SR-GNN-**RUA**
- SR-GNN-**UHW**

The abbreviation **A** is for activation, **NA** is for no activations and the others as previously described.

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
SR-GNN	70.49	30.59	4071.07	51.26	17.77	3081.98
UHW	70.64	30.79	3662.37	51.18	17.69	3072.47
HA	70.62	31.40	3212.77	50.88	17.52	**2821.71**
UHA	70.64	31.29	4057.00	50.93	17.50	2974.47
RHA	70.63	31.24	3754.08	**51.18**	**17.82**	2977.85
UA	70.55	31.34	**3067.94**	50.65	17.20	2833.80
RA	70.69	31.23	3763.27	51.07	17.78	2976.22
RUA	70.63	30.84	3914.77	51.10	17.74	3054.15
NA	**70.73**	**31.44**	3220.24	50.90	17.35	2828.53

Table 6: Performance of the SR-GNN model compared with the models from this ablation study, the models are abbreviations from the models listed in Section **5.5**

Table **6** shows the results of this experiment. Unlike the previous experiments, it is more difficult to determine the best model. When evaluating performance in terms of evaluation metrics, there are negligible differences in the various models. Looking at the run-time of the models, a few models seems to have an edge, including the SR-GNN-**HA**, SR-GNN-**UA** and the SR-GNN-**NA** models..

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
R	70.62	31.10	3810.70	**51.12**	**17.78**	3002.74
U	70.61	31.16	3675.52	50.89	17.48	2983.72
H	**70.63**	**31.31**	**3671.55**	51.00	17.61	**2961.62**
N	70.73	31.44	3220	50.90	17.35	2828

Table 7: The averaged results from experiment 3

Table **7** shows the average results for each component. **R** seems to perform the best on the Yoochoose dataset, while the **H** remains the fastest on both, and performs best on the Diginetica dataset. While these do not show an as clear tendency as the Table **5**, it is still possible to derive certain conclusions from the table such that the models including the weights in the hidden state **H** still performs the best, and removing the activation function does not significantly harm the evaluation metrics of the model, but does seem to decrease the run-time significantly.

Considering the result from this experiment and the previous experiment, there is an indication that the impact of weights and activation functions are not major and that a key component in the GRU network is the hidden state. Following this observation an additional experiment will be conducted. This experiment will utilize only the hidden state, ie. the SR-GNN model will be reduced to a simple Light-SR-GNN. This transforms the model to be one consisting of an initial embedding layer for the item embeddings followed by an attention layer. This will be compared to the candidate models that have been explored throughout these experiments, the results are shown on Table **8**.

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
SR-GNN	70.49	30.59	4071.07	**51.26**	**17.77**	3081.98
UHW	**70.64**	30.79	3662.37	51.18	17.69	3072.47
HA	70.62	**31.40**	**3212.77**	50.88	17.52	**2821.71**
Light-SR-GNN	69.71	30.31	4983.65	49.86	16.91	4197.75

Table 8: Performance of the SR-GNN model compared with the candidate models from the ablation study.

Here three models from each part of the ablation study are compared with the Light-SR-GNN model. All models have one thing in common, the hidden state. Reducing the SR-GNN model to a Light-SR-GNN decreases the evaluation metrics slightly in terms of MRR@20 and P@20 slightly, but observing the run-time of the model a much higher discrepancy is present. The Light-SR-GNN model has a much higher run-time compared with the other models. This indicates some components in the GRU-network are necessities in terms of the model converging earlier, although the Light-SR-GNN model is not far off in

terms of evaluation metrics, but is slower. Thus, of the 22 models presented throughout these experiments the SR-GNN-**HA** model will be the candidate model and the following experiment will utilize this in an experiment where the scalability of the model will be tested.

6.4 Experiments: Phase Four - Scalability

The ablation study conducted has concluded the best model to be SR-GNN-**HA**. Theoretically, this model runs in $\mathcal{O}(n(m \cdot d))$, this is the same as the SR-GNN model, the only difference is fewer matrix multiplications and fewer activation functions, as elaborated in Section **4**. To see how the models scale to larger datasets and whether the complexity analysis holds, an experiment will be conducted. This experiment will aim to gradually increase the sample size and record the training time.

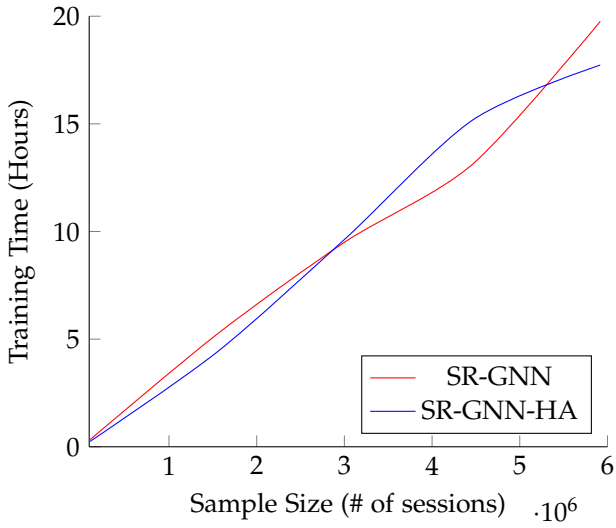

Figure 3: Scalability of the SR-GNN and the SR-GNN-**HA** models

The result is shown on Figure **3**. The result indicates both models run in linear time, which aligns with the complexity analysis. Although both models seem similar, a few observations can be seen when comparing both models at specific events. Both models deviate at two events, the first time is around 4 million sessions where the SR-GNN seems to have an edge, the second time is around 6 million events where the SR-GNN-**HA** model stabilizes and requires 2 hours less training compared to the SR-GNN model. This indicates that, if the experiment ran for greater sample sizes, there would have been a higher difference in the two models training time.

6.5 Evaluating Experiments

Models	Yoochoose 1/64			Diginetica		
	P@20	MRR@20	T(s)	P@20	MRR@20	T(s)
POP	4.95	1.31	0.014	1.4	0.15	0.013
S-POP	26.73	20.00	0.029	26.15	16.68	0.013
Item-KNN	55.11	23.00	14.277	45.07	15.52	14.204
GRU4Rec	72.55	32.63	697.472	50.99	17.40	1338.128
NARM	74.85	32.26	1200.618	49.47	16.24	2487.746
STAMP	77.12	37.19	2213.976	42.33	12.71	4522.789
SR-GNN	70.60	30.59	5147.082	51.48	18.03	3578.083
SR-GNN-HA	70.62	31.40	3212.760	50.88	17.52	2821.680

Table 9: The performance of SR-GNN-**HA** with other baseline algorithms for two datasets

To demonstrate the performance of SR-GNN-**HA** it is compared to the other session-based recommendation algorithms presented in Section **5.2**. The comparison is shown in Table **9**, where it can be seen how the older algorithm POP and S-POP performs poorly in terms of the evaluation metrics. The metrics where they excel are in the run-time, where Item-KNN performed noticeably better in comparison but had a worse run-time. POP, S-POP and Item-KNN are all non-neural algorithms and are relatively simple in their implementation. The simplicity of their implementation can be seen in their run-times being lower than the other algorithms, though their performance also lacks. It should however be noted that Item-KNN outperforms one of the neural algorithms on the Diginetica dataset. The neural algorithms generally perform better than the non-neural, but with a much higher run-time. GRU4Rec, NARM and STAMP all have similar performance in terms of the evaluation metrics, but differ largely in run-time and which dataset they perform the best on, such as STAMP performing the best on Yoochoose 1/64 among all neural algorithms, while also performing the worst on Diginetica. In terms of run-time, GRU4Rec performed the best on both datasets, with NARM coming in second and STAMP in third. The run-time roughly doubles for Diginetica for the algorithms compared to Yoochoose 1/64. For SR-GNN-**HA** it performs a bit worse compared to other neural algorithms on Yoochoose 1/64 but is better than both NARM and STAMP on Diginetica, and is as good as GRU4Rec. In terms, of performance, it is worst on Yoochoose 1/64 but interestingly does not show the same tendency of doubling its run-time on Diginetica.

7 Discussion

In the following section, the empirical study will be discussed. The discussion will aim to address the validity of the results and the methodology used throughout the study. Furthermore, observations in the datasets and the work by the original authors will be addressed.

7.1 Results from the Experiments

The first experiment conducted was a replication of the ablation study from Wu et al. The purpose of the study was to set the baseline for the following ablation study since by recreating the original ablation study confidence in the testing framework could be heightened. One problem however was regarding some implementation, which was not fully elaborated by the original authors and as a result thereof some assumptions were necessary. More specifically, some of the models might not have been constructed exactly how the original models were, hence it is impossible to assure the experiments are completely identical. One such model is the SR-GNN-**AVG** where the implementation was not given in detail in the paper by Wu et al. and multiple ways of constructing the average pooling were therefore attempted [1]. In the implementation, each session's length is normalized to ensure all sessions in a batch has the same length, which is done to optimize the computations through whole matrix multiplications. Such length normalization implies that most items will have lots of empty clicks towards the end, where a placeholder item embedding is placed. In the average pooling, it was attempted to pool across the entire normalized session length, which yielded poor results dissimilar to the results as mentioned by Wu et al. To accommodate for that, it was attempted to only average across the clicked items and ignore the repeated placeholder item, which yielded similar results to the results presented by Wu et al. However, there is no guarantee the implementation matches fully in this case, but the results indicate so. In the fourth experiment, the run-time of the SR-GNN and SR-GNN-**HA** are computed on different sizes of the Yoochoose dataset to observe how the models handle larger datasets. In this experiment, it can be observed how the complexity is linear for both models, which aligns with the complexity analysis conducted in Section **4**, where the complexity is deemed polynomial with the power being 1, hence a linear run-time. However, the scalability of both models are not conclusive from this experiment alone. Considering Figure **3** from Section **6.4** different conclusions may have been drawn if the experiment ran for multiple iterations and with an increased sample size as each training time plotted is based on one iteration, which would deviate if several iterations were run, and an average was plotted.

Since this work has a focus on the scalability of the models, three metrics for evaluating the performance were chosen, including MRR@20, P@20, and the total training time. The experiments were conducted in a way where after each phase of the experiments, the best model was chosen and used in the subsequent experiments. When looking at the results, several models could be argued to be the best one on some metric, and no model stood out between the different metrics or even the datasets. Therefore it could have made sense to pick several models, but given time restrictions this was not a possibility. Another way could have been to pick one model for each dataset, but we wanted to find one generally good model. Therefore we picked the SR-GNN model in the initial experiment, despite it not being the fastest model. It was chosen since it showed promising results in our experiments and was the best in the work produced by Wu et al. In the second experiment the SR-GNN-**UHW** model was picked because of the promising results, especially on the Yoochoose dataset, and lastly we chose the SR-GNN-**HA** model was choose because it was generally a good model. One thing to note about choosing the best model from the experiments is the number of iterations the model was trained on. Each model ran for three iterations and since many of the models have similar performance, using more iterations could potentially result in different results. Looking at e.g. the standard deviation on the models from Section **6.2** the standard deviation for the models P@20 score ranged from: $0.023 - 0.618$. This shows, that even though there is not a high standard deviation, given the similar performance of the models, several of them could potentially switch places in terms of their ranked performance. Unfortunately due to limited time, it was not possible to run the experiments for more iterations even though it could have been beneficial.

7.2 Dataset Analytics

This part of the discussion will address the datasets used, the lack of additional datasets and possible flaws with the datasets. Section **5.1** elaborates the two datasets used in the study, including Diginetica and Yoochoose. Because of the size of the Yoochoose dataset, the original authors utilized two subsets of the datasets, 1/64 and 1/4 of the Yoochoose dataset. Given time restrictions the 1/4 Yoochoose dataset was dropped, but further analysis shows how the impact it may have had could be negligible. Firstly a scalability experiment was conducted in Section **6.4**, where we see how the selected models scale to various subsets of the Yoochoose dataset. Secondly, the outcome of an analytical observation from the datasets indicates the distribution of items in both 1/64 and 1/4 Yoochoose are identical, these observations will be elaborated further later. The exclusion of the 1/4 Yoochoose allows for a faster execution time of all the models but restricts the intermediate results since each experiment does not consider a large scale dataset now. This limits the possible outcome of the experiments since different conclusions from each experiment may have been drawn if a larger dataset was considered.

Observing the results presented by Wu et al. [1], identical performance can be seen on Yoochoose 1/64

and Yoochoose 1/4 for all models and the result show significant lower performance on the Diginetica dataset which are equivalent to the tendencies observed in our experiments. Because of these observations we choose to analyse the distributions of clicks in the datasets.

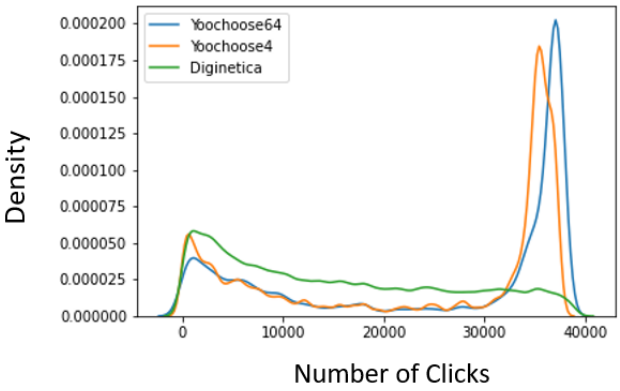

Figure 4: Item-Click distribution over the Yoochoose 1/64, Yoochoose 1/4 and Diginetica datasets

Figure **4** shows the item click distributions over the three datasets. The figure reflects a kernel density estimation plot of the clicks in the datasets. It applies a Gaussian kernel with a specific bin-width and estimates the probability of clicks in any of the bins leading to the plot. Considering the plots of the Yoochoose 1/64 and Yoochoose 1/4 datasets, which are similar, two peaks exist. The first peak reflects how given any click there is a high probability it is a low frequency item. However, more importantly through the second peak, we see the majority of the clicks stems from so-called super items with over $> 30,000$ clicks. A few observations can be drawn from the figure. Firstly the Yoochoose 1/64 and Yoochoose 1/4 datasets are more or less the same in terms of distribution, secondly, these two datasets have a few items with the majority of the clicks. This means a vast majority of items in these datasets have very few clicks. In contrast, when observing the Diginetica dataset, a more even distribution is prevalent. This does not address the poor performance, in fact, a more uniform distribution should be ideal for the experiments as it lessens the likelihood that the model will fit to the bias of the distribution. However the source does not give clues for what might have yielded such a distribution, it only mentions that the dataset is constructed by Diginetica and its partners. To explore this, the sessions were analyzed as a network, where each session was modelled as nodes and edges between nodes were made if an intersection of items existed. From this, it was learned how all items primarily were connected in a large component for both datasets. As such, there was not a set of significant size components in the Diginetica dataset, which could have been the case if the dataset was constructed from multiple distinct sources. Summarily, a distinction between the Yoochoose dataset and the Diginetica dataset exists as reflected on Figure **4**.

When running the baseline models we got unexpected results. The results for the baselines differ from the results shown by Wu et al.[1] and are in some cases better or worse. An interesting observation is how the evaluation metrics differ for POP, S-POP and Item-KNN which does not contain any random elements, therefore the expectation would be to obtain equivalent results. One likely explanation for this could be differences in the datasets, thus leading to different results. This was confirmed when looking at the statistics for the datasets in Table **2** and the ones provided in [1]. From more investigation it can be seen how the provided datasets from [1] does not match with their statistics, so it can be assumed that the datasets have been updated or changed at some point since publication.

8 Conclusion

The focus and exploration of the SR-GNN model presented by Wu et al. [1] have led to a couple of discoveries. First of all the ablation study of the hybrid embedding construction and the core experiments by Wu et al.[1] have been reproduced in Section **6**, which establishes a foundation of confidence in the results. However, one limitation is that the experiments are not run on the larger Yoochoose 1/4, which is argued in Section **7.2** to follow the same underlying patterns as in Yoochoose 1/64, but could still give additional insight into how the models differ. The two Yoochoose datasets more specifically both follow a power-law distribution in the item clicks, where certain items with many clicks dominate the overall clicks across all the sessions.

Some of the baseline algorithms compared within Section **6.5** achieve competitive results compared to the SR-GNN model. However, this is not to the detriment of the study, as it focuses narrowly on the proposed components in the SR-GNN model.

The experiments revealed that given the SR-GNN model it is possible to achieve similar results in terms of MRR@20 and P@20 when discarding the idea of the connection matrix and the GRU layer, which together are the core part of the graph layer. This conflicts with the primary idea of the model, which is the modelling of sessions as a graph, where the nodes are items and edges are clicks between items. This transforms the model to be one consisting of an initial embedding layer for the item embeddings followed by an attention layer. This is reflected through the

presentation of the Light-SR-GNN model presented in Section **6.3**, as the model achieves MRR@20 and P@20 scores close to the SR-GNN model. That is without considering optimizations, which could have been added to the Light-SR-GNN, but due to constraints have not been explored, but are simply referred to in Section **9**. Hence the total training time aspect of the Light-SR-GNN is not as optimal even though it contains fewer weights, which could potentially be mended by additions. This fact can be connected to the time complexity analysis presented in Section **4** as the bottleneck is shown to be the requirement of computing a score for each item embedding for every session training sample, which is distinct from the components discarded in Light-SR-GNN. Summarily, it was learned that the graph modelling, which entailed the connection matrix and the GRU part of the SR-GNN model has not a clear, or even negligible benefit to the model's overall performance. Finally, it can be seen that the application of an ablation study to the SR-GNN model has shed a light on the less impactful components, which led to the SR-GNN-**HA** model elaborated in Table **6**, which improves upon the SR-GNN model.

9 Future work

Given the time constraints, not all interesting and relevant aspects have been considered.

As future work, it could be interesting to change the attention mechanism in [1] to a soft attention mechanism, which considers the interest of all clicked items in relation to all other items, as opposed to the current implementation which considers the attention only from the standpoint of the last clicked item. Changing this could alleviate the bias of emphasizing the last clicked item.

Further aspects to explore could be to extend the analysis to the Yoochoose $1/4$ dataset, and possibly consider more datasets, as there is a clear contrast in performance in both the MRR@20 and P@20 scores of the model depending on the dataset.

An additional future work could be the exploration of the impact of applying higher order connection matrices, as proposed in [13].

One aspect not covered in this paper is security. This topic is very relevant when making a recommendation system since a good system should not only have good evaluation metrics but also be robust against things like spam attacks[18]. Because of this, one relevant path to discover could be to study the influence of different spammer behaviours on the recommendation systems to improve on this aspect.

References

1. Wu, S., Tang, Y., Zhu, Y., Wang Liang and Xie, X. & Tan, T. *Session-based Recommendation with Graph Neural Networks* in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (eds Hentenryck, P. V. & Zhou, Z.-H.) **33** (AAAI Press, Honolulu, HI, USA, July 2019), 346–353. https://aaai.org/ojs/index.php/AAAI/article/view/3804.

2. He, X. *et al.* LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *CoRR* **abs/2002.02126.** arXiv: 2002.02126. https://arxiv.org/abs/2002.02126 (2020).

3. Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. *Item-based collaborative filtering recommendation algorithms* in *Proceedings of the 10th international conference on World Wide Web* (2001), 285–295.

4. Shani, G., Brafman, R. I. & Heckerman, D. An MDP-based Recommender System. *CoRR* **abs/1301.0600.** arXiv: 1301.0600. http://arxiv.org/abs/1301.0600 (2013).

5. Rendle, S., Freudenthaler, C. & Schmidt-Thieme, L. *Factorizing Personalized Markov Chains for Next-Basket Recommendation* in *Proceedings of the 19th International Conference on World Wide Web* (Association for Computing Machinery, Raleigh, North Carolina, USA, 2010), 811–820. ISBN: 9781605587998. https://doi.org/10.1145/1772690.1772773.

6. Hidasi, B., Karatzoglou, A., Baltrunas, L. & Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *CoRR* **abs/1511.06939** (2016).

7. Hidasi, B. & Karatzoglou, A. *Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations* in (Association for Computing Machinery, 2018). ISBN: 9781450360142. https://doi.org/10.1145/3269206.3271761.

8. Tan, Y. K., Xu, X. & Liu, Y. Improved Recurrent Neural Networks for Session-based Recommendations. *CoRR* **abs/1606.08117.** arXiv: 1606.08117. http://arxiv.org/abs/1606.08117 (2016).

9. Jannach, D. & Ludewig, M. *When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation* in *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Association for Computing Machinery, Como, Italy, 2017), 306–310. ISBN: 9781450346528. https://doi.org/10.1145/3109859.3109872.

10. Liu, Q., Zeng, Y., Mokhosi, R. & Zhang, H. *STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation* in (July 2018), 1831–1839.

11. Li, Y., Tarlow, D., Brockschmidt, M. & Zemel, R. Gated Graph Sequence Neural Networks (Nov. 2015).

12. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* **20,** 61–80 (2009).

13. Chen, Y.-H., Huang, L., Wang, C.-D. & Lai, J.-H. Hybrid-order Gated Graph Neural Network for Session-based Recommendation. *IEEE Transactions on Industrial Informatics,* 1–1 (2021).

14. Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* **abs/1609.02907.** arXiv: 1609.02907. http://arxiv.org/abs/1609.02907 (2016).

15. Zhou, J. *et al. Graph Neural Networks: A Review of Methods and Applications* 2021. arXiv: 1812.08434 [cs.LG].

16. Li, J. *et al. Neural Attentive Session-Based Recommendation* in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (Association for Computing Machinery, Singapore, Singapore, 2017), 1419–1428. ISBN: 9781450349185. https://doi.org/10.1145/3132847.3132926.

17. Malte Ludewig Noemi Mauro, S. L. & Jannach, D. Empirical analysis of session-based recommendation algorithms. *Springer.* https://link.springer.com/article/10.1007%5C%2Fs11257-020-09277-1 (2020).

18. Zhang, C. *et al.* Enhancing the robustness of recommender systems against spammers. *PLOS ONE* **13,** e0206458 (Nov. 2018).