NTNU
Norwegian University of
Science and Technology

Attention Mechanisms in Hierarchical Session-Based Recommendation

Kristoffer Lervik

Master of Science in Computer Science
Submission date: June 2018
Supervisor: Massimiliano Ruocco, IDI

Norwegian University of Science and Technology
Department of Computer Science

Kristoffer Lervik

Attention Mechanisms in Hierarchical Session-Based Recommendation

TDT4900 Master's Thesis, Spring 2018

Supervisor: Massimiliano Ruocco

Artificial Intelligence Group
Department of Computer Science
Faculty of Information Technology and Electrical Engineering

Abstract

The use of attention mechanisms in different applications of recurrent neural networks has yielded significantly higher accuracies, but their use in session-based recommender systems is largely unexplored. In addition to increasing accuracy, attention mechanisms also allow for easy visualization of which input items impact the prediction of new outputs, which can lead to a better understanding of how users act, their habits and which past items are important for predicting future ones. In this project we explore different ways of including attention mechanisms in session-based recommender systems with hierarchical recurrent neural networks to improve accuracy. Four experimental models have been developed that apply attention mechanisms in different ways. These models have been tested on two separate datasets. We show that the attention mechanisms used in problems such as text translation and document classification yield a small increase in accuracy in the recommender system problem, with a slightly higher increase when combined with the use of a simple temporal attention mechanism. While we have been unable to achieve the same accuracy improvements as attention mechanisms have achieved in other problem domains, our improvements together with attention weight visualizations suggest that a temporal attention mechanism can work in recommender systems and that a more sophisticated temporal attention mechanism could increase accuracy even more.

Preface

This report was written as part of my TDT4900 Master's Thesis at the Norwegian University of Science and Technology during the Spring semester of 2018. It was supervised by Massimiliano Ruocco. The project consists of this report and a GitHub repository[1] containing the code used for experiments.

Kristoffer Lervik
Trondheim, June 17, 2018

[1] https://github.com/krislerv/tdt4900-master-thesis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's online society, people can be overwhelmed by choice when browsing websites such as YouTube, Spotify or Netflix. To combat this issue, these services employ the use of recommender systems: systems that give users recommendations on what to do next. Recommender systems are well-researched, but little attention has been paid to incorporating attention mechanisms into them. In this chapter we explain the background and motivation for this project as well as formulate a goal and some research questions that we aim to answer.

1.1 Background and Motivation

Early recommender systems such as collaborative filtering gave recommendations by comparing the histories of users. When recommending for one user, the system would find other users with similar histories and recommend things that those users liked.

In recent years deep neural networks have become increasingly popular and they have been applied to the recommender system problem as well. This has some pros and cons. The pro of using deep neural networks in general is that they can achieve higher accuracy by learning latent features that are hard to be manually engineered. Another pro is that with recurrent neural networks we can model sequences of actions, meaning that we can make predictions based on previous actions in a sequence. This is not possible with techniques such as collaborative filtering. The main con of using deep neural networks is interpretability. Deep neural networks in general are a very black-box type of system, meaning that even though the input and output are known, the inner workings of the systems aren't very easy to interpret.

One problem area that has been explored by several researchers in the last

few years is session-based recommender systems. They model sequences of user actions, split into sessions. The task is to predict the next action, given the earlier actions of the session.

A recent breakthrough in many areas of application of recurrent neural networks is attention mechanisms. They allow you to focus on (attend to) different parts of the input when predicting the output. This is done with a set of learnable weights, which can be used to create human interpretable visualizations of the network.

One other aspect of recommender systems with recurrent neural networks that has been mostly ignored is the temporal aspect of the problem. There is a variable amount of time between actions and between sessions. This information is likely to influence how important past actions are for predicting new ones, so using this information could result in better predictions.

1.2 Goals and Research Questions

As mentioned in the previous section, there is still a lot of research to be done in session-based recommender systems with recurrent neural networks. In particular, the use of attention mechanisms have been successful in other applications of recurrent neural networks with architectures very similar to that of the session-based recommender system problem. For this project, we will research the possibility of applying it here as well.

Goal Explore different ways that attention mechanisms can be applied to session-based recommender systems for the task of increasing accuracy.

This can potentially be a really big task, so to narrow it down we will focus on a hierarchical RNN based architecture for session-based recommendation as described in Section 3.2. We will primarily focus on trying to apply attention mechanisms in similar ways to what has been done in other applications of recurrent neural networks. The hierarchical architecture would then be the best option as its architecture is the most similar to those.

Research question 1 Can attention mechanisms that have been developed for other applications of recurrent neural networks be applied to hierarchical session-based recommender systems?

The other unexplored area for this problem is the temporal aspect. Trying to model time and time difference is a difficult task and an entirely different area of research. However, it may be possible to improve recommendation accuracy by utilizing the temporal aspect to create a simple attention mechanism.

Research question 2 How can the temporal aspect of recommender systems be included in an attention mechanism to increase accuracy?

1.3 Research Method

As our research will build on previous research, the first step in our methodology is to read previously published papers on relevant topics. Then, based on the research, we will create hypotheses and develop new solutions with respect to our project goal. Finally, we will run experiments to gather results.

1.4 Thesis Structure

Chapter 2 gives an overview of the background theory that is necessary to understand the project. This information is given in a high level manner.

Chapter 3 digs a bit deeper into some of the State of the Art for attention mechanisms and session-based recommender systems. It contains details of the theory that will be applied to the experimental models.

Chapter 4 presents the baseline and experimental models as well as the theory and ideas behind them.

Chapter 5 presents the experimental plan and setup. This chapter also present the datasets used to train the models. It concludes with the results of the experiments as well as a discussion of the results.

Chapter 6 concludes the project with an evaluation of the project as a whole as well as a some possible directions for future research.

Chapter 2

Background Theory

This project builds on a lot of theory and many concepts. This chapter will introduce those in a high level manner so that the rest of the report can be understood.

2.1 Artificial Neural Networks

An artificial neural network is a concept that tries to model the biological neural network in the human brain. The human brain consists of many neurons that are connected to each other through synapses. The neurons receive electrical signals from other neurons and can in turn fire their own electrical signals to other neurons again. These signals is what allows us humans to think.

An artificial neural network works much in the same way as its biological counterpart. An ANN is built out of a set of layers, each layer containing nodes (neurons) with weighted connections to the next layer of nodes. A simple ANN will consist of three layers: The input layer, the hidden layer and the output layer. This architecture is depicted in Figure 2.1. Each node in the hidden layer receives one or more inputs from nodes in the input layer. These inputs are summed using the weights of the connections between the input nodes and the hidden nodes. This sum is passed through an activation function (typically sigmoid, tanh or ReLU) to determine the output of the node. If the value is high enough, the hidden node fires. If not, it doesn't. These firings (or non-firings) of hidden nodes are the inputs to the output nodes. The output nodes work similarly to the hidden nodes, using a weighted sum of its inputs to determine the final output. Advances in this field in recent years have made neural networks become increasingly popular. However, they have some limitations such as being unable to model sequences in a good way.

From here on, artificial neural networks will be referred to simply as neural networks.

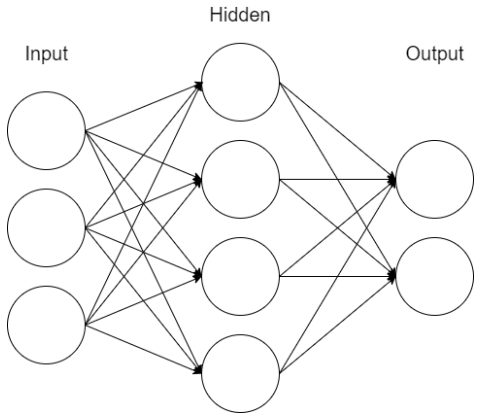


Figure 2.1: A simple artificial neural network. The circles are nodes and the arrows are weighted connections between the nodes.

2.2 Recurrent Neural Networks

While general neural networks are great at a lot of tasks, they are unable to accurately model sequences. One such example is text translation. If you fed a sequence of words into a feed-forward neural network, it would probably perform badly at translating that text to a different language. The reason for this is that the feed-forward network has no memory of the previous word it translated, so at best it would manage a crude word-by-word translation. This becomes especially problematic when the source and target languages have different sentence structures. What we need is a neural network that is able to remember what came before and this is where recurrent neural networks come in.

Recurrent neural networks are different from feed-forward neural networks in that they have a hidden state (this is different from hidden layers). For each new input-data, the hidden state is updated to reflect what it has seen before. That way, when it gets a new input it can look at both the input and the representation of what has come before to make a prediction. This can be seen in Figure 2.2.

[1]Figure taken from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

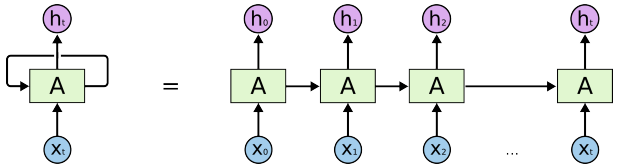


Figure 2.2: Left: A recurrent neural network. Right: An unrolled recurrent neural network. Here, x_t is the input at each time step, h_t is the hidden state/output and A is the RNN cell, either a simple activation function such as sigmoid/tanh or a more complex cell such as LSTM/GRU.[1]

A problem with this hidden state architecture is that it won't be able to accurately represent things that happened long in the past. Many problems, such as text translation, have these long-term dependencies between inputs far apart. Say you are trying to translate a sentence where two words that are dependent on each other are twenty words apart. This means that twenty words have modified the hidden state when you need the dependency between them. A solution to this problem is LSTMs, Long Short Term Memory. These LSTM cells are used in place of the normal nodes in the RNN. They work by having a cell state that transfers information between cells in a sequence. The LSTM cell has several gates that modify the cell state and they are able to learn which parts to keep and which parts to modify. GRU (Gated Recurrent Unit) is another type of cell that has been used to combat the long-term dependency problem. It was introduced by Cho et al. [2014] and has been shown to be able to outperform LSTMs in certain tasks [Hidasi et al., 2016]. It is a simplified version of the LSTM where the cell state and hidden state have been combined into one.

The next hidden state, h_t, is determined by a weighted sum of the previous hidden state and the new state, n_t. [2]

$$h_t = (1 - z_t)n_t + z_t h_{t-1} \tag{2.1}$$

The weights are decided by the update gate, z_t.

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \tag{2.2}$$

The new state is a combination of the new input and the previous hidden state.

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t(W_{hn}h_{t-1} + b_{hn})) \tag{2.3}$$

How much of the previous hidden state to keep is decided by the reset gate, r_t.

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \tag{2.4}$$

In these equations, σ is the sigma function. All instances of W and b are weight matrices and bias vectors respectively.

2.3 Session-Based Recommender Systems

As mentioned in Section 1.1, recommender systems are systems that give recommendations on what to do next based on a user's history. The simplest of these is collaborative filtering where it is assumed that people with similar histories have similar interests and give recommendations based on that. The problem with this approach is that they don't model sequences very well.

A solution to this problem is the use of recurrent neural networks. In the same way that words are ordered in sequences, events such as video-viewings are ordered in sequences. Thus, using the same techniques as when predicting the next word in a text, you can use recurrent neural networks to predict future user actions. As input, the system gets the user's latest action. It then outputs a likelihood for each possible action to be the next action in the sequence, based on the previous actions in the sequence. This way, you can use information about the user's recent history to better recommend things that might interest them in the future.

A newer concept in recommender systems is session-based recommendations. These systems split actions into sessions, meaning that if two consecutive actions are days apart, the first one won't affect the second one. This has the positive effect of more accurately modeling user behavior, but it also means that the sequences used to predict new actions become shorter which can result in a drop in accuracy. Some models that mitigate this problem are presented in Section 3.2.

2.4 Attention Mechanisms

In many problems in deep learning, the input can be split into smaller pieces. Examples of this include text, speech and images. Text is comprised of sentences which are comprised of words. Speech contains words and intonations. Images can be divided into sub-images. When generating outputs for these kinds of

[2]There are some small variations on the GRU layer in different literature. The formulas used here are taken from the official PyTorch 0.4.0 documentation as this is the framework that was used in this project. https://pytorch.org/docs/0.4.0/nn.html#torch.nn.GRU

problems it is often not optimal to look at the input as a a single indivisible entity, but rather as something that is comprised of many parts.

When we humans look at a picture in order to get an impression of it, we don't just look at the entire thing and base our impression on that. Instead, we focus on certain parts, one at a time. This is analogous to what attention mechanisms try to do. Instead of looking at the entire input all the time, attention mechanisms try to focus on (attend to) certain parts of the input at a time. In more technical terms: When generating output, find out which parts of the input are relevant for the next part of the output. Here, "parts of the input" can for example refer to elements in a sequence or features in an image.

Attention mechanisms are fairly new, but have already been used in many different applications such as text translation [Bahdanau et al., 2016], image caption generation [Vinyals et al., 2015], speech recognition [Chorowski et al., 2015] and text summarization [Rush et al., 2015].

To explain the details of attention mechanisms and how they improve on previous models, it is easiest to use an example. For this, we will use text translation. Before attention mechanisms, text translation systems would pass the input sentence through an RNN to produce a context vector. This context vector is then used as the initial hidden state in another RNN to generate the output sentence (see Figure 2.3). The problem with this model is that you must be able to encode the entire input sentence into a single vector. This becomes problematic, especially for long inputs [Bahdanau et al., 2016].

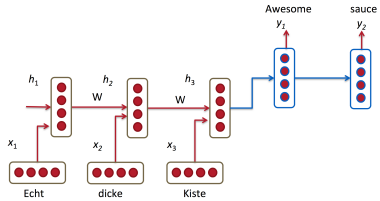


Figure 2.3: Text translation before attention mechanisms. The red part of the network encodes the input sentence into a single context vector. The network must then be able to generate the entire output sentence from this single vector.[3]

When using attention mechanisms, instead of using a single context vector for the entire output sentence, we create a context vector for each word in the output sentence. These context vectors are weighted sums of all the input states, taking into account the last hidden state generated by the output RNN to determine the weights. A more in-depth, mathematical explanation of this example is given in Section 3.1.

The power of attention mechanisms is that you no longer need to be able to encode an entire sentence into a single vector. This not only helps with the problem of long inputs, but having context vectors for each output word means that the model can more easily use combinations of input words when forming output words. This is especially helpful in text translation as a single word in one language might be translated into several words in another language. A visualization of the results of attention mechanisms in text translation can be seen in figure 2.4.

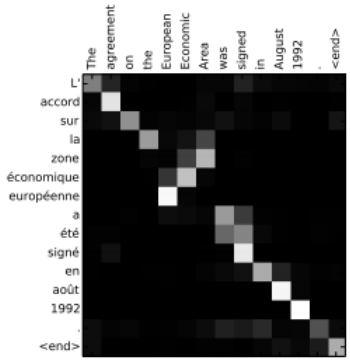


Figure 2.4: English to French translation. How the attention mechanism chose to focus on each word in the input sentence when predicting the target sentence (taken from [Bahdanau et al., 2016]).

[3]Figure taken from http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

Chapter 3

State of the Art

The experimental models presented in this project build on a lot of research. To help with the understanding of those models, this chapter gives a more in-depth look at some of the most important research papers that helped create them.

3.1 Text Translation Using Encoder-Decoder Networks with Attention Mechanisms

The idea of encoder-decoder networks is to encode an input sequence into a feature representation which can then be decoded into a target sequence [Cho et al., 2014]. In the case of translation, you pass a sequence of words in the source language into an RNN to encode it into a context vector. This context vector is then used as the initial hidden state for the decoder network, which will try to predict the translation of the input sentence in the target language. However, it is hard to encode all the relevant information needed in a fixed-length vector.

Bahdanau et al. [2016] tries to improve encoder-decoder translation by learning which parts of the source sentence are relevant for predicting words in the target sentence. The idea is that instead of creating a single context vector to predict the entire target sequence, you create a context vector for each word in the target sequence. The hidden state s_i for the i-th element in the target sequence is given by:

$$s_i = f(s_{i-1}, y_{i-1}, c_i). \tag{3.1}$$

where y_{i-1} is the last target item and c_i is the context vector for the i-th decoder item.

Each context vector is based on all hidden states in the encoder layer and the last hidden state in the decoder layer. The context vector for the i-th element in the target sequence is given by:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{3.2}$$

Here, h_j is the output of the encoder RNN for the j-th element in the input sequence and T_x is the length of the input sequence. α_{ij} is the attention weight for the i-th decoder element and the j-th encoder element. The attention weight tells us how important the j-th encoder element is for predicting the i-th decoder element. α_{ij} is given by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{3.3}$$

where e_{ij} is given by:

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.4}$$

e_{ij} is the attention energy of the j-th encoder element and i-th decoder element and is a general model that scores how well the j-th encoder element matches the i-th decoder element. Note that the formula for a_{ij} is simply the softmax function, meaning that all the attention weights for each target item will sum up to 1.

This idea was further explored by Luong et al. [2015], who introduced several scoring functions for computing the attention energies, as well as a model for local attention where instead of receiving attention weights from all encoder nodes, you would only consider some of them.

3.2 Session-Based Recommendations with RNNs

One of the first attempts at using RNNs for recommender systems was Hidasi et al. [2016]. The idea is that the actions (video-viewings, songs listened to etc.) performed by a user in a single session are related to each other. They propose a model where a user's actions are split into sessions. The sessions are lists of IDs, each ID belonging to a certain action. Each ID in these sessions is then embedded and passed through one or more GRU (Gated Recurrent Unit) layers and finally a feed-forward layer which produces an n-dimensional output (n being the number of available actions). This is shown in Figure 3.1 The model achieves good results, but it suffers from not having an appropriate initial hidden state (known as the cold start problem). The hidden state is initialized with zeros

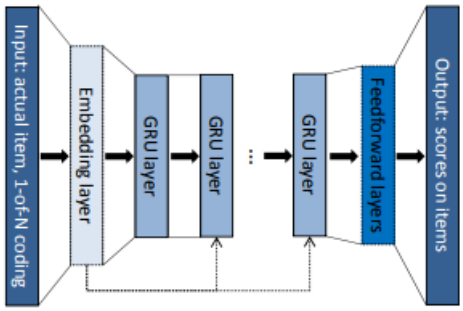


Figure 3.1: Architecture for session based recommendation (taken from [Hidasi et al., 2016]).

and is reset whenever a session is finished, meaning that predictions early in the session will be mostly random.

Ruocco et al. [2017] propose a hierarchical architecture where the first level makes predictions identical to Hidasi et al. [2016] (Figure 3.1) and the second level generates the initial hidden state. The second level does this using session representations of the user's most recent sessions. A session representations is meant to represent what the user did in a given session. The paper proposes two methods to create session representations: average pooling and last hidden state. Average pooling creates an average sum of the embeddings of each event in the session. Last hidden state simply uses the last hidden state from when the session went through the first level RNN.

When predicting a new session for a user, the most recent session representations for that user are retrieved. The session representations are considered as a sequence (from oldest to newest) and are passed through a GRU layer in the second level RNN. Since each session representation is meant to represent the user's activity in a given session, the output of the inter-session GRU is meant to represent everything the user has done recently. The output of the GRU is then used as the initial hidden state when predicting the user's next session. The hierarchical architecture achieves overall higher accuracy and is a great help to the cold start problem, meaning that accuracy early in the session is much higher. The architecture of the hierarchical model can be seen in Figure 3.2.

The model of Ruocco et al. [2017] is very similar to the model of Cho et al. [2014] for encoder-decoder text translation described in Section 3.1. If we imagine the architecture as an encoder-decoder network, the most recent session representations become the inputs to the encoder, the encoder then generates an initial hidden state for the decoder, which then generates recommendations for the user's

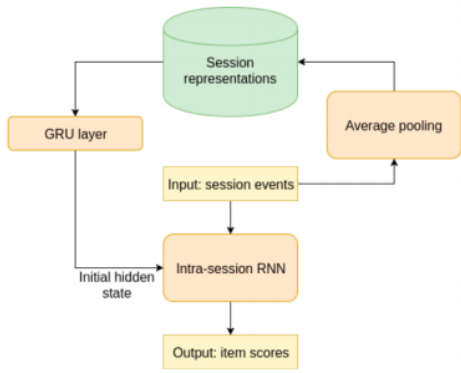


Figure 3.2: The hierarchical architecture from Ruocco et al. [2017] with average pooling used to create session representations. The module labeled "GRU layer" is the second level RNN that creates the initial hidden states (taken from Ruocco et al. [2017]).

next session.

Another version of the hierarchical session-based recommender system is [Quadrana et al., 2017]. It is in many ways similar to Ruocco et al. [2017], but it uses the concept of session representations in a different way. Instead of creating and storing representations of individual sessions, this paper has a single representation for each user. The user representation starts out as a zero vector and when the system is done predicting a session for a user, that user's representation is updated using the session's last hidden state. This can be seen in Figure 3.3. The update is done using the following:

$$c_m = GRU_{usr}(s_m, c_{m-1}) \tag{3.5}$$

where c_m is the new user representation, s_m is the session representation (last hidden state) of the last session, c_{m-1} is the last user representation and GRU_{usr} is a GRU layer used to model the user activity across sessions. The hidden state of the next session is then initialized as:

$$s_{m+1,0} = tanh(W_{init}c_m + b_{init}) \tag{3.6}$$

where W_{init} and b_{init} are learnable weights.

These hierarchical session based-recommender systems are able to achieve higher accuracy by using the user's past data. The drawback of this approach is

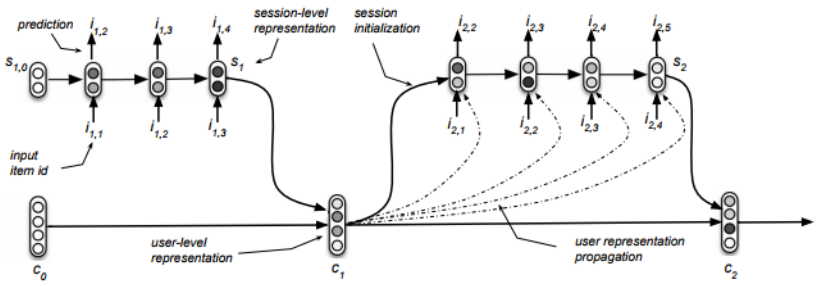


Figure 3.3: The two-layer architecture from Quadrana et al. [2017]. Each session prediction (top layer) updates the user-level representation. This is then used as the initial hidden state for the next prediction (taken from Quadrana et al. [2017]).

that they assume the user's past data is available, which is not necessarily the case [Tuan and Phuong, 2017]. Li et al. [2017] proposed a different architecture to tackle the cold start problem. They propose a different way of creating an initial hidden state where they have two encoders, a global encoder and a local encoder. These encoders together create an attention mechanism where the system is able to detect the main purpose of the session. When learning on a new session, the global encoder is simply a GRU that the whole session is run through to produce a final hidden state. This is to produce a session representation of the session, i.e. it finds out what the main focus of the session was. Say the user was browsing an online store and in a given session clicked on ten sweaters and a pair of shoes. It is then likely that the focus of the session was on the sweaters (the shoes might have been a miss-click). In mathematical terms, the session representation is given by:

$$c_t^g = h_t \tag{3.7}$$

where t is the length of the session and h_t is the hidden state of the global encoder at time t. This is similar to the Last Hidden State method of creating session representations in Ruocco et al. [2017]. The local encoder gets all the hidden states of the session and creates a context vector in a very similar way to Formula 3.2:

$$c_t^l = \sum_{j=1}^{t} a_{tj} h_j \tag{3.8}$$

where t is the length of the session, h_j is the j-th hidden state of the local encoder and a_{tj} is the j-th attention weight, given by:

$$\alpha_{tj} = v^\top \sigma(A_1 h_t + A_2 h_j) \tag{3.9}$$

where v, A_1 and A_2 are learnable weights and σ is the sigmoid function. Finally, c_t^g and c_t^l are concatenated to create the final context vector c_t.

Instead of using another RNN in the decoder, Li et al. [2017] use a bi-linear similarity function to calculate the similarity between the embedding of each candidate item and the context vector c_t. The similarity function is given by:

$$S_i = emb_i^\top B c_t \tag{3.10}$$

where emb_i is the embedding of item i and B is a matrix of learnable weights. Finally, the similarity scores of each item is put through a softmax function which produces the final ranking of items.

In the works of Liu et al. [2016], they explore a different approach to sequential recommendation. They propose a model for RNN recommendation that is context-aware. They mention two types of contexts: input contexts and transition contexts. Input contexts are all about what situation the user was in when they made an action. For example, the user was at home and it was raining so the user looked at umbrellas in an online store. The transition contexts consider the time between actions. If the time between two actions is very long, they are unlikely to be connected. To explain how these contexts are incorporated into the recommender system, we must look at how hidden states are computed:

$$h_k^u = f(r_k^u M_{c_{I,k}^u} + h_{k-1}^u W_{c_{T,k}^u}) \tag{3.11}$$

Here, h_k^u is the hidden state of user u at time k, r_k^u is the input of user u at time k, $c_{I.k}^u$ is the input context for user u at time k and $c_{T,k}^u$ is the transition context for user u at time k. The important thing to notice is that the weight matrices (M and W) depend on the context. Instead of using a single M matrix and a single W matrix, they use adaptive context-specific matrices. This way they can better capture the context of the user's actions. The paper doesn't go into much details on how the input contexts are specified, but it does mention how the transition context is decided. The time interval between two events is continuous so it would be impossible to learn a transition matrix for every value. Therefore, they split the time differences into buckets. The state of the art in session-based recommender systems have usually just ignored the temporal aspects of the problem (i.e. the use of timestamps). This might not be optimal as the non-uniform time differences between actions can have a big say in how important past actions are for predicting future ones. Liu et al. [2016] is therefore quite unique with its use of this largely unexplored area.

While recurrent neural networks are great, one must not forget other approaches as well. Jannach and Ludewig [2017] show that a session-based k-nearest-neighbors model achieves the same or better results as Hidasi et al. [2016]. It takes as input a user session and finds the k most similar past sessions. These are found using a heuristic method to sample nearby neighbors, allowing it to scale for large datasets. It calculates a score between each of these sessions and the input session and uses these to give a score to new potential actions. Finally, they propose a hybrid model of the k-nearest-neighbors approach and the model of Hidasi et al. [2016]. This is shown to outperform either model by themselves.

3.3 Hierarchical Attention Mechanisms

In certain applications, researchers have experimented with more than one level of attention mechanisms. Yang et al. [2016] proposed a hierarchical structure for document classification. A document consists of sentences that consist of words. The idea is the following: First, you embed each word. Then you pass the words of a sentence through a word encoder. This word encoder then produces a sentence representation of that sentence. After you have done this with every sentence in the document, you go to the sentence encoder. The sentence encoder does the same thing as the word encoder, except with sentence representations instead of word embeddings. The sentence encoder produces a document representation. This document representation can then be used for document classification.

Because words don't contribute equally to the representation of a sentence (a word like "the" doesn't really say anything about the contents of the sentence) they use an attention mechanism to emphasize words that are important to the meaning of the sentence. This is done similarly to Bahdanau et al. [2016], but without any form of the last decoder hidden state. To clarify, this means that the function in Formula 3.4 would depend only on h_j.

In the same way that all words don't contribute equally to a sentence, not all sentences contribute equally to the contents of a document. Therefore, they employ a second level of attention after the sentence encoder to emphasize the important sentences in the document. The formulas for the sentence level attention are identical to the word level attention (using the sentence encoder hidden states). The architecture of the hierarchical document classifier can be seen in Figure 3.4.

Qin et al. [2017] uses a dual-stage attention mechanism for an RNN implementation of the Nonlinear autoregressive exogenous (NARX) model. To put it simply, it predicts the next value of a time series based on the previous values in the series and previous values in multiple driving series. That is, the input consists of n driving series $X = (x^1, x^2, ..., x^n)$ where each driving series consists of T time steps, $x^k = (x_1^k, x_2^k, ..., x_T^k)$.

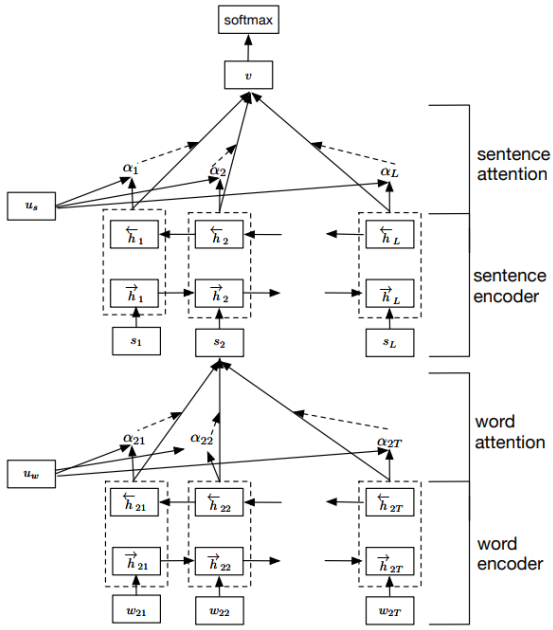


Figure 3.4: The hierarchical architecture for document classification from Yang et al. [2016]. It uses attention mechanisms to find which words are representative of a sentence and which sentences are representative of a document. (taken from Yang et al. [2016]).

It uses an encoder-decoder network where the inputs to the encoder at time t is the t-th time step of each driving series. These values are first passed through an attention mechanism that learns which driving series are relevant. After this, the inputs are passed to the encoder. The encoder hidden states at each time step is passed through another attention mechanism to select the relevant ones. This all culminates in a context vector which is used to predict the next value in the target time series.

Chapter 4

Architecture/Model

This chapter presents the models used in the project. The models consist of one baseline model and four experimental models, each attempting to incorporate attention mechanisms into the hierarchical session-based recommendation architecture.

4.1 Baseline Model

In order to evaluate the accuracy of the experimental models in a useful manner, we need a baseline to compare against. This project is largely based on the work of Ruocco et al. [2017] as described in Section 3.2. Therefore, the first part of this project was to implement the model as described in the paper.

Each user has up to n session representations, $R^u = (r_1^u, r_2^u, ..., r_n^u)$. From here on, we will omit the user-notation to simplify the notation. These session representations are passed though an RNN using GRU cells, ultimately producing a final hidden state h_n. Since each session representation is meant to represent the user's activity in a session, the output of the inter-session GRU is meant to represent everything the user has done recently. h_n is then used as the initial hidden state of another RNN which takes as input a new session that it will try to predict. Given a set of inputs $X = (x_1, x_2, ..., x_t)$, it produces output $O = (o_1, o_2, ..., o_t)$ where each output is the predicted next action given all previous actions in the session. The process of predicting a new session can be seen in Figure 4.1.

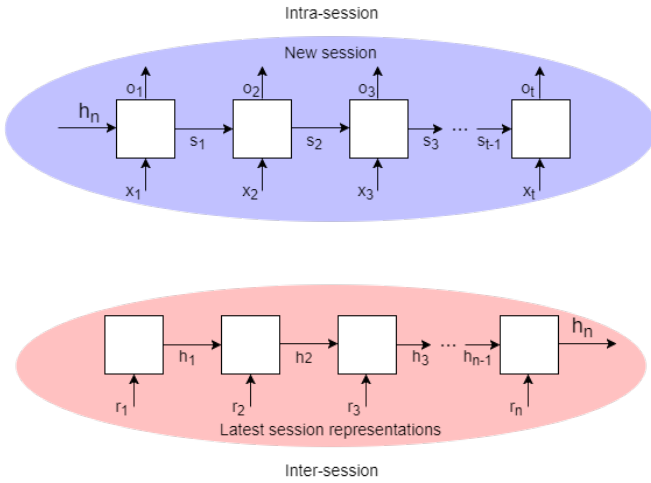


Figure 4.1: The baseline architecture. The pink area is the inter-session RNN which generates an initial hidden state for the intra-session RNN based on the user's most recent session representations. The blue area then uses the initial hidden state to predict the next action in the new session.

4.2 Experimental Models

This section will present the experimental models that have been developed throughout the project. The purpose of this project is to try to incorporate attention mechanisms in the hierarchical session-based recommender architecture, so all these models have that in common.

4.2.1 Intra-Session Attention

The first attempt at incorporating attention mechanisms into the baseline architecture was to implement the attention mechanism described by Bahdanau et al. [2016] in the intra-session part of the model. That is, the part that predicts the new session. To reiterate, the paper by Bahdanau et al. [2016] improved the text translation problem using an encoder-decoder network with an attention mechanism. As mentioned in Section 3.2, the architecture of the baseline model of this project is very similar to Cho et al. [2014]. Bahdanau et al. [2016] has the same architecture as Cho et al. [2014], but with the addition of the attention mechanism. Because of this, it should be possible to implement a version of the attention mechanism of Bahdanau et al. [2016] in the baseline architecture for

hierarchical session-based recommendation.

The rationale behind this model is that to generate a long sequence of actions from one initial hidden state is difficult to do accurately. Instead, the intra-session RNN takes an extra input at each time step: a context vector. This context vector uses the previous hidden state of the intra-session RNN to create a weighted sum of the hidden states of the inter-session RNN. This way, the intra-session RNN can focus on different session representations at each time step and over time will learn which session representations are important for predicting the next action. The i-th context vector is given by:

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{4.1}$$

where h_j is the hidden state generated by the j-th session representation in the inter-session RNN. a_{ij} is an attention weight that says how important the j-th session representation is for predicting the i-th action in the new session. a_{ij} is given by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \tag{4.2}$$

where e_{ij} is the attention energy of the j-th session representation and i-th action in the new session. It is given by:

$$e_{ij} = a^{\top} tanh(Bh_j + Cs_{i-1}) \tag{4.3}$$

where a, B, and C are learnable weights and s_{i-1} is the previous hidden state of the intra-session RNN. This architecture can be seen in Figure 4.2.

Another way to calculate the attention energies was presented in Luong et al. [2015]. Instead of summing the previous decoder hidden state and the encoder hidden state, they are concatenated. This has the benefit of not losing any information when summing the two embeddings. This is the method we used for creating attention energies and is given by:

$$e_{ij} = a^{\top} tanh(B[h_j; s_{i-1}]) \tag{4.4}$$

where $[h_j; s_{i-1}]$ is the concatenation of the j-th encoder hidden state and the previous decoder hidden state.

In the paper of Bahdanau et al. [2016], they use a bidirectional RNN in the encoder (their equivalent of the inter-session RNN). In this case, each hidden state in the inter-RNN would be a concatenation of the forward and backward hidden state:

$$h_k = [\overleftarrow{h_k}; \overrightarrow{h_k}] \tag{4.5}$$

where $\overleftarrow{h_k}$ is the hidden state of the k-th session representation in the backwards RNN and $\overrightarrow{h_k}$ is the hidden state of the k-th session representation in the forward RNN. This can in some applications increase accuracy as an event late in a sequence might affect events earlier in the sequence [Tan et al., 2016]. In our case it did not lead to an increase in performance. Ruocco et al. [2017] also did not use a bidirectional encoder.

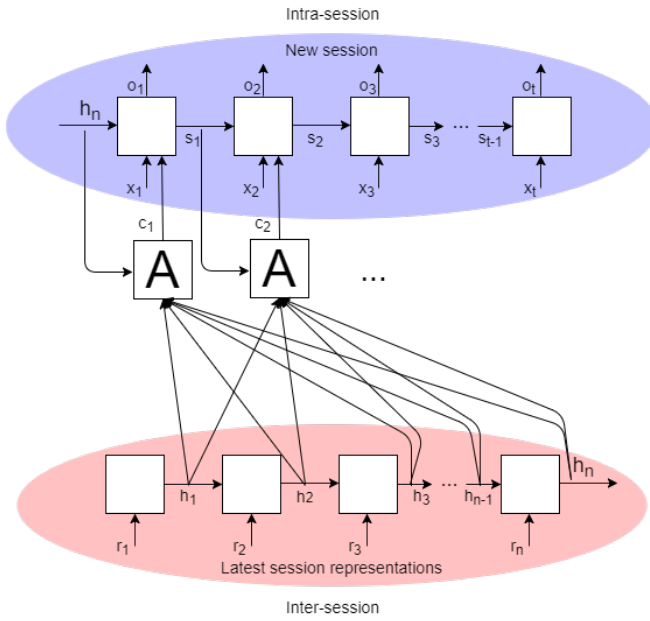


Figure 4.2: The intra-session attention architecture. In addition to creating an initial hidden state for the intra-session RNN, the hidden states of the inter-session RNN are now also used to create a context vector for each node in the intra-session RNN.

4.2.2 Inter-Session Attention

Something unique about the recommender system problem when it comes to applications of attention mechanisms is the temporal aspect of the problem. Many problems are either timeless (such as generating text) or the time between events is constant. In the recommendation problem, each event has a timestamp associated with it and the time between events varies.

The timestamp tells us what time of the day and which day of the week a user performed an action. This information likely influences the user's actions. A user is probably more likely to listen to death metal on a Friday night than on a Monday morning. In addition to this, the time between events and between session representations matters. If two different session representations are 3 hours and 160 hours old then they will likely have different importance when it comes to predicting a new session.

The attention mechanism in this section is simpler than the one in the previous section. Instead of looking at how each session representation affects each action in a new session, we use an attention mechanism to calculate how much each session representation should affect the initial hidden state of the intra-RNN. This can be seen in Figure 4.3.

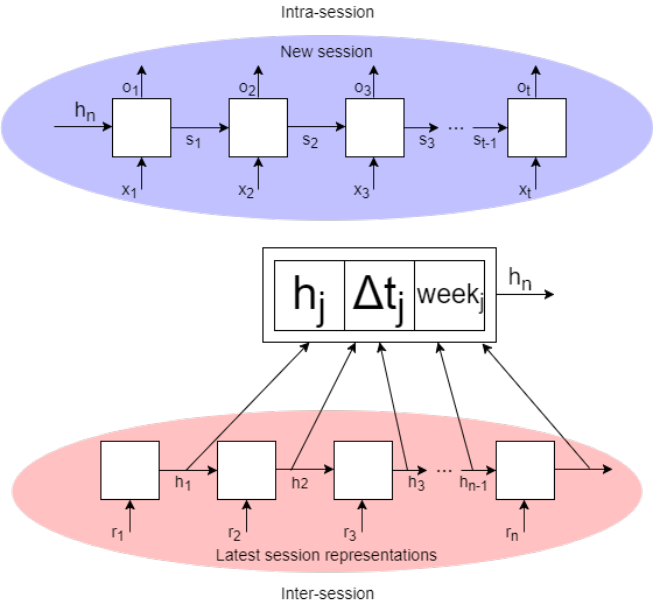


Figure 4.3: The inter-session attention architecture.

We create an initial hidden state by calculating a weighted sum of the hidden states of all the session representations.

$$h_n = \sum_{j=1}^{n} \alpha_j h_j \tag{4.6}$$

Here, a_j is an attention weight that says how important the j-th session representation is for predicting future sessions. In the previous model the attention weights were dependent on the last hidden state of the intra-session RNN, but we don't use those here. Instead of calculating how important each session representation is given the previous actions of a session, this will learn in a more general/global way which session representations are important and thus should be given a higher weight in the initial hidden state. The idea is then that the system should be able to identify which embeddings matter for predicting future actions and which ones do not. This idea is in many ways similar to [Yang et al., 2016] which learns which words are important in classifying a sentence and which sentences are important in classifying a document. a_j is given by:

$$\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{n} exp(e_k)} \tag{4.7}$$

where e_j is the attention energy of the j-th session representation. It is given by:

$$e_j = a^\top tanh(Bf(h_j, \Delta t_j, week_j)) \tag{4.8}$$

where a and B are learnable weights. The function f depends on three parameters.

h_j is simply the j-th hidden state of the inter-session RNN.

Δt_j is the time difference between the j-th session representation and the new session that we are trying to predict. This time difference is put into one of many buckets, each one hour long. There are a total of 169 buckets, one for each hour of a week (168 hours) and one for everything older than that. This is in many ways similar to the transition context of Liu et al. [2016] mentioned in Section 3.2. The time differences are then embedded so that they have the same dimension as h_j. The idea is that the system will learn how time differences affect how much a given session representation should influence the initial hidden state. It is natural to assume that more recent session representations should count more, but that might not be universally true. Humans are creatures of habit, so maybe sessions that happened 24 hours prior should have more influence than those that happened 12 hours prior.

$week_j$ is the time of the week when the j-th session representation started (for example: Wednesday, 22:00). Just like Δt_j, the time of the week is put into one of 168 buckets, one for each hour of the week. The value is then embedded. The idea here is that the context of the week affects how a user acts. A user's habits might be different on a Monday at 14:00 vs. a Saturday at 02:00. It is important that when training this attention mechanism that all users are in the same timezone. Therefore, a dataset containing only users from the same timezone will be used when testing the time of week attention mechanism.

The latter two parameters might sound similar, but there is a difference between them. Δt_j tells us about the age of the session representation relative to the new session we are trying to predict. $week_j$ tells us about the time of the week (regardless of the timestamp of the new session) the session took place.

The function f is then the concatenation of the hidden representations of the three parameters, or a combination of any two, or any one. For example, the attention energy formula for the inter-session attention mechanism using the hidden state and delta-time attention variables (but not the week-time attention variable) would look like the following:

$$e_j = a^\top tanh(B[h_j; \Delta t_j]) \tag{4.9}$$

where $[h_j; \Delta t_j]$ is the concatenation of the h_j and Δt_j variables.

4.2.3 Combined Inter-Session and Intra-Session Attention

Since we have two different attention mechanisms that work on different levels of the architecture, the natural next step is to combine them. This model does just that. It uses the inter-session attention mechanism to generate an initial hidden state and the intra-session attention mechanism to predict the next action in the new session. The goal is to see if joinly learning the two attention mechanisms will improve the accuracy more than either one on their own.

4.2.4 Hierarchical Attention

Yang et al. [2016] created a model for document classification with attention mechanisms in a hierarchical structure. They split a document into sentences which are split into words. The words in a sentence form a sequence and are passed through an RNN. The outputs are then used to create attention weights and finally a sentence representation is created by a weighted average of the RNN outputs. Once all the sentence representations have been created, they are passed through an RRN and used to create attention weights to compute a document representation as a weighted average of the sentence representations.

This model is in essence very similar to the model by Ruocco et al. [2017]. If we imagine words as events, sentences as sessions, documents as users, we see that the same model used by Yang et al. [2016] could also be used by a recommendation model very similar to Ruocco et al. [2017]. The document representation used for classification in Yang et al. [2016] is comparable to the initial hidden state used for prediction in Ruocco et al. [2017].

The session representations in the recommendation model are comparable to the sentence representations in the document classification model, but the way they are created is different. The session representations are created one

at a time as the model predicts new sessions. They are then stored away and retrieved the next time that user is having one of their sessions predicted. To make this more comparable to the document classification model (and to make attention mechanisms possible) the recommendation model has to be modified. Each time the model predicts a new session for a given user, the n most recent previous sessions (not session representations) for that user are retrieved and used to create session representations, whether that be average pooling, last hidden state or attention mechanism.

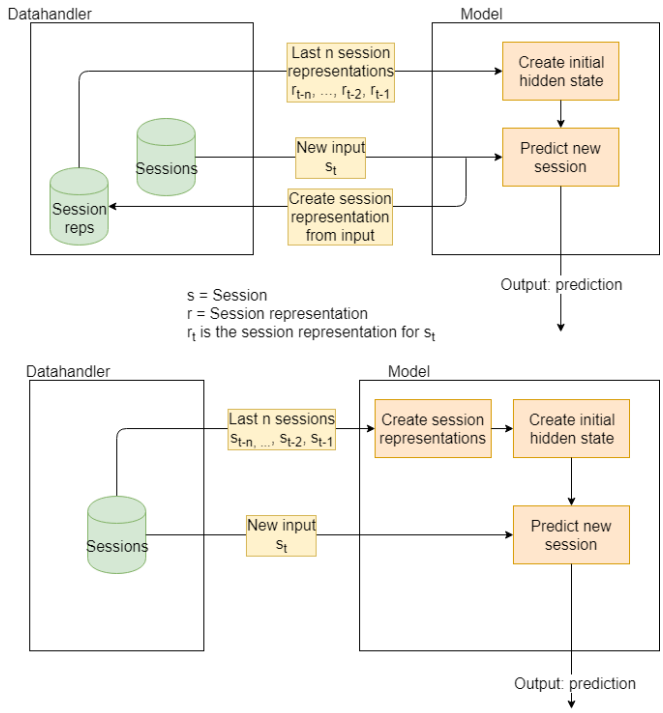


Figure 4.4: Illustrated differences between the hierarchical model and the baseline model. The top half is the baseline model and the bottom half is the hierarchical model. Each box in the model is a separate GRU (however, when not using attention mechanism the "create session representations" box can also be average pooling).

This difference might be subtle. Figure 4.4 illustrates the difference. In addition to allowing attention mechanisms this new model also benefits from

the fact that the events used to create session representations are re-embedded each time they are used. This means that the embedding matrix is more easily trained and training of the model doesn't suffer from the fact that old session representations have been created by an embedding matrix with weights that might be very different from what they are now.

To create session representations in this new model, we retrieve the n latest previous sessions for the user. These sessions are used to produce r_i, the $(n-i)$-th most recent session representation, in one of three methods: average pooling, last hidden state, attention mechanism. This can be seen in Figure 4.5.

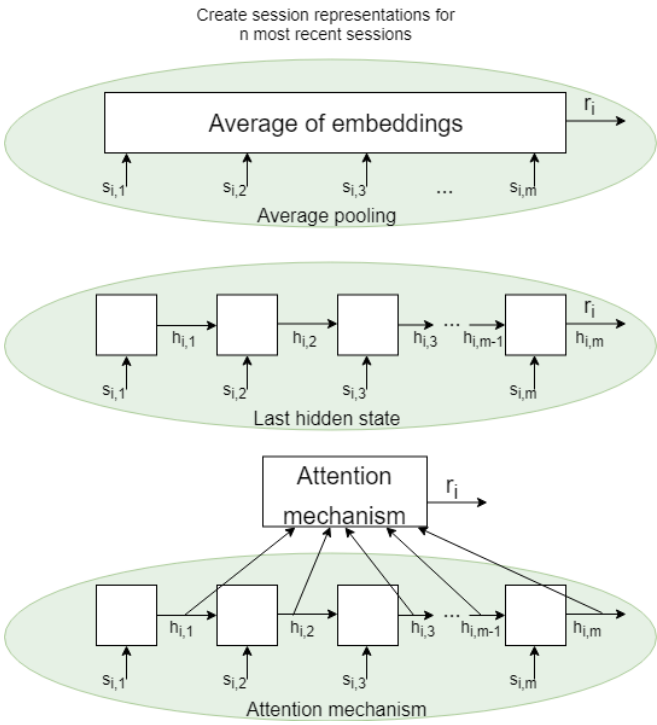


Figure 4.5: The three different ways of creating session representations in the hierarchical model. $s_{i,j}$ is the embedding of the j-th element of the i-th session. r_i is the session representation of the i-th session.

To formalize, using the average pooling method of creating session represen-

tations, the i-th session representation r_i is given by

$$r_i = \frac{1}{m} \sum_{j=1}^{m} s_{i,j} \tag{4.10}$$

where $s_{i,j}$ is the embedding of the j-th element in the i-th session and m is the length of the session.

The last hidden state method passes the session through a GRU and uses the output (the last hidden state) as the session representation.

$$r_i = GRU([s_{i,1}, s_{i,2}, ..., s_{i,m}]) \tag{4.11}$$

The attention method uses a weighted average of the outputs of the GRU, with the weights decided by an attention mechanism.

$$e_{i,j} = a^\top tanh(Bh_{i,j}) \tag{4.12}$$

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{m} exp(e_{i,k})} \tag{4.13}$$

$$r_i = \sum_{j=1}^{m} \alpha_{i,j} h_{i,j} \tag{4.14}$$

where a is a vector of learnable weights, B is a matrix of learnable weights and $h_{i,j}$ is the GRU output of the j-th timestep for the i-th session.

Now that we have the session representations, the next step is creating the user representation. This is done using the exact same methods as for the session representations. Using the average pooling method, the user representation u is given by

$$u = \frac{1}{n} \sum_{i=1}^{n} r_i \tag{4.15}$$

where r_i is the i-th session representation and n is the total number of session representations (up to a cap).

The last hidden state method passes the session representations, as a sequence from oldest to newest, through a GRU and uses the output (the last hidden state) as the user representation.

$$u = GRU([r_1, r_2, ..., r_n]) \tag{4.16}$$

The attention method uses a weighted average of the outputs of the GRU, with the weights decided by an attention mechanism.

$$e_i = c^\top tanh(Dh_i) \tag{4.17}$$

$$\alpha_i = \frac{exp(e_i)}{\sum_{k=1}^{n} exp(e_k)} \tag{4.18}$$

$$u = \sum_{i=1}^{n} \alpha_i h_i \tag{4.19}$$

where c is a vector of learnable weights, D is a matrix of learnable weights and h_i is the GRU output of the i-th session representation. u is then used as the initial hidden state when predicting the new session. This model is illustrated in Figure 4.6.

The way the hierarchical model creates user representations is exactly the same way as the inter-session attention mechanism creates its initial hidden state. Because of this, we can concatenate the delta-t and week-t to the hidden state when creating attention energies in the same way as in Formula 4.8. In fact, the hierarchical attention model is a generalization of the inter-session attention model. However, the inter-session model was conceived and tested first and some of its parts were deemed unnecessary to carry over to the hierarchical model. To be able to present the full results of the inter-session model, the models were therefore presented separately.

4.3 Per-User Attention Weights

In Section 4.2, all the learnable weights of the attention mechanisms are global, meaning that the same linear layers are used to create attention weights for all the users. This might not be ideal, as different users can have different interests or habits which can result in the attention mechanism becoming too general and not being able to predict accurately for anyone.

This problem is in some ways similar to a problem faced in multilingual neural machine translation. The weights that are effective at translating from language A to language B might not be effective at translating from language C to language D. Some authors have experimented with multilingual NMT with attention mechanism and to tackle this problem they have set certain parts of the model to be language specific such as the encoder and decoder [Firat et al., 2016], or the attention layer and decoder [Dong et al., 2015].

We chose to experiment with per-user linear layers for computing attention weights. This is not a new model, but rather a modification of the ones presented

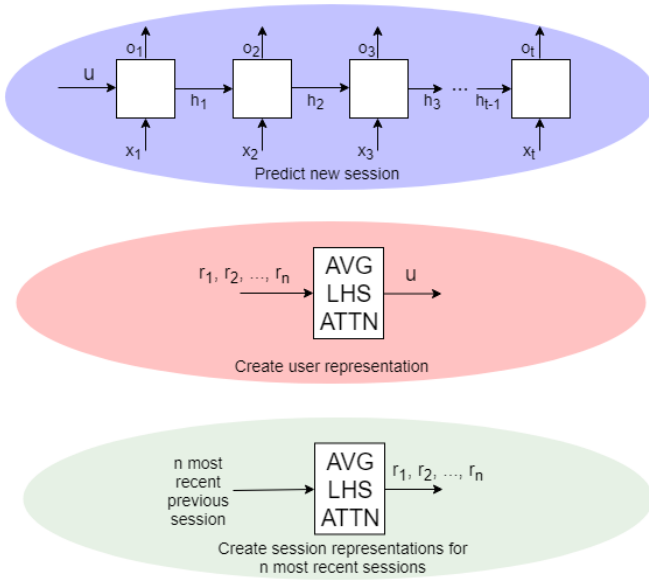


Figure 4.6: The layers of the hierarchical model. Green: The n most recent previous sessions for a user are used to create session representations. This is done with one of three methods: Average pooling, last hidden state, attention mechanism. Red: The session representations are used to create a user representation using one of the same three methods as in green. Blue: The user representation is used as initial hidden state to predict a new session.

in Section 4.2. For example, Formula 4.17 for the attention energies of user representations in the hierarchical attention model can be modified to

$$e_i = c_{usr}^{\top} tanh(D_{usr} h_i) \tag{4.20}$$

Here, c_{usr} and D_{usr} are still learnable weights, but they are specific to a single user, meaning that these weights will only be used to predict/learn for that specific user. Throughout the rest of the report, when we mention per-user attention weights, we mean per-user linear layers for computing attention weights.

Chapter 5

Experiments and Results

In this chapter we explain everything relating to the experiments. We present the experimental plan in relation to the research questions. We then explain everything related to the setup and configuration of the experiments in addition to the datasets used for training. Finally we present the results and discuss some of the more interesting findings.

5.1 Experimental Plan

This project has quite a few different experimental models. There is the intra-session model, inter-session model, combined intra-session and inter-session model, and the hierarchical model. The inter-session model can also be split up into any permutation of the three different types of inter-session attention. In addition, there is also the per-user attention weights.

The intra-session attention model will be tested to see if the attention mechanism of encoder-decoder translation can be applied to recommender systems and see similar results. This is in relation to our first research question.

In the inter-session attention, all permutations of the three attention mechanisms will be tested to see if attention mechanisms can improve the generation of initial hidden states. This is in relation to both research question, but with special interest in the second one about temporal attention.

Some configurations from the inter-session attention will be combined with the intra-session attention to see if the system performs better when jointly learning both attention mechanisms. This is in relation to both research questions.

For the hierarchical model a new set of baselines will be tested that will use the same methods (Average Pooling, Last Hidden State) used by the old baselines. We will test the hierarchical model with attention mechanisms to see

if this model outperforms the baseline of the hierarchical model and/or the other attention mechanisms. This is again in relation to both research question.

Also in relation to both questions are the tests of per-user attention weights. We want to see if the attention mechanisms will improve in accuracy if each user has their own linear layers for computing attention weights. These tests are somewhat slow as they require certain parts of the model to be done without mini-batching. Thus, only some models will be tested using user-specific attention mechanisms.

5.2 Datasets

Two datasets have been used for experimenting. The first dataset is from the music website Last.fm. It contains tuples of user id, timestamp, artist name, song name. Each tuple is a song the user has listened to. The second dataset is from the social media/news aggregate website Reddit. It contains tuples of username, subreddit, timestamp. Each tuple is one comment the user has made in the given subreddit. Some statistics for the datasets can be seen in Table 5.1.

Dataset	num items	num users	num sessions	avg session length
Last.fm	94284	977	630774	8.1
Reddit	27452	18271	1135488	3.0
Last.fm CET	42665	265	144926	7.7

Table 5.1: Main dataset statistics

5.2.1 Last.fm

The Last.fm dataset does not use the concept of sessions. Therefore, this had to be done manually. Ruocco et al. [2017] empirically set a time limit where, if two consecutive events were further apart than the time limit, they would be split into two separate sessions. The time limit for the Last.fm dataset was set to half an hour (1800 seconds). This might seem long for two songs to be considered part of the same session, but the reason for this is due to the next problem they faced.

They also encountered a problem with the amount of unique songs in the dataset. Using their proposed model with all the songs would not be possible as it would require too much memory. Instead, they chose to simplify the dataset by looking at individual artists instead of individual songs. The model predicts

the next artist the user is going to listen to and the memory usage is reduced to a manageable size. This way, the 1800 second time limit makes more sense.

5.2.2 Reddit

Similarly to the Last.fm dataset, the Reddit dataset also has no predefined sessions. Therefore, we have to split them manually like for the Last.fm dataset. However, the time limit shouldn't necessarily be the same. The Reddit dataset only records user comments, not all user activity. A user does a lot more than just commenting. They browse subreddits to find interesting topics and they read posts, all of which takes time. Therefore, it might be a long time between comments. Accounting for this, Ruocco et al. [2017] set the time limit to be one hour (3600 seconds). We used the same time limits for both datasets.

5.2.3 Last.fm CET

The week-time attention model requires the use of timestamps to determine what time of the day and which day of the week a session started (for example: Wednesday at 22:00). Both datasets are from international websites that have users from all over the world, but the timestamps are in UTC. This means that if a user in Norway and a user in Canada performed an action at the same time their timestamps would be identical, but their time of day would be different. This would hinder the model in learning. Because of this, a new dataset was created from the Last.fm dataset (which has country information on each user) to only include the users that belong to the CET timezone. This timezone was chosen as it was the one with the most users. As a result, the week-time model is only tested on this dataset.

5.2.4 Filtered Datasets

While testing, we hypothesized that the accuracies achieved with attention mechanisms could be improved by filtering the datasets in certain ways. We looked at two possible ideas. First, to limit the time from a user's first session to their last. Second, filtering users based on their average session length and session count. With these datasets we want to see if they outperform the full datasets and more importantly if they benefit more from attention mechanisms than the full datasets.

Limited Time-Interval Datasets

Each user's sessions are spread over a long timespan. For the Reddit dataset, the average timespan between a user's first session and their last is 265 days. For

the Last.fm dataset, the same timespan is 535 days. Users' interests and habits change over time so limiting the time frame might make the datasets more learnable. Datasets were created where, for each user, we would only keep the events that happened in the first x months of that user's history. We experimented with 1, 2 and 3 months. Statistics for the datasets can be seen in Table 5.2. The reason that the smaller datasets have fewer users is because of a requirement that each user has at least three sessions. So if a user had three sessions in the full dataset, but one of those sessions is outside of the time interval, then that user is dropped from the new dataset.

Dataset	num items	num users	num sessions	avg session length
lastfm-1-month	24816	893	32418	8.4
lastfm-2-months	33647	913	59814	8.3
lastfm-3-months	40105	924	90416	8.3
reddit-1-month	14789	11164	250320	3.7
reddit-2-month	18065	12745	430919	3.5
reddit-3-month	20102	13685	579545	3.4

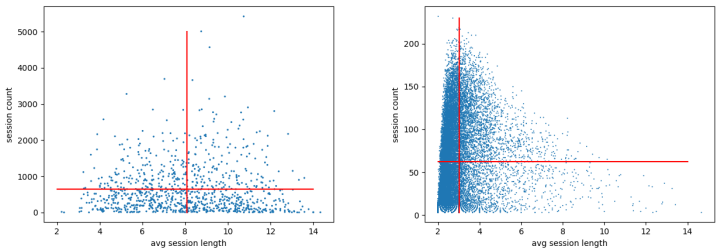
Table 5.2: Statistics for the Limited Time-Interval datasets

User Statistics-Filtered Datasets

We computed two statistics for each user: How many sessions they have in total and the average length of their sessions (shown in Figure 5.1). After this, we computed the average of these two statistics for all users in each dataset. We then created four user groupings: those with above average for both statistics, below average for both statistics, higher than average session length and lower than average session count, lower than average session length and higher than average session count. Our main hypothesis was that the dataset containing only users with higher than average in both statistics would perform better than the full datasets. The users with lower than average in both statistics could be considered "noise" preventing the models from learning. Statistics for the datasets can be seen in Table 5.3.

5.3 Experimental Setup

With the amount of data in the two (original) datasets, a GPU is required to avoid having to train for days at a time. The experiments were carried out by a machine provided by Telenor. The machine contains an Intel Xeon E5-2650



(a) Last.fm (b) Reddit

Figure 5.1: The session counts and average session lengths for each user in each dataset. The red lines denote the averages of each statistic.

Dataset	num items	num users	num sessions	avg session length
lastfm-removed-high-high	35147	311	86988	6.1
lastfm-removed-high-low	46969	180	218641	6.2
lastfm-removed-low-high	35133	309	69925	10.0
lastfm-removed-low-low	57433	177	255220	9.8
reddit-removed-high-high	13508	8139	190128	2.4
reddit-removed-high-low	18311	5306	559440	2.6
reddit-removed-low-high	6409	1881	55483	4.4
reddit-removed-low-low	15247	2945	330437	3.8

Table 5.3: Statistics for the Filtered datasets. "lastfm-removed-high-low" means the Last.fm dataset where all users with higher than average session length and/or lower than average session count have been removed. Note that when presenting results, we will omit the "removed" part of the names of these datasets.

v4 CPU, two NVIDIA Tesla P100-PCIE 16GB GPUs and 64GB of RAM. The processing power of the machine was split between 7+ students, each doing their own experiments.

The datasets were split into 80%-20% train-test datasets where the first 80% of each user's actions were put in the training set and the remaining 20% were put in the testing set. The training set is used to update weights with backpropagation while the testing set is used to evaluate how well the system generalizes and to record evaluation metrics.

The system uses the Adam optimizer and cross-entropy loss. Cross-entropy loss was chosen because both the output of the system and the true labels are probability distributions, so it makes sense to use cross-entropy. Dropout was used as regularization.

The hyperparameters are presented in Table 5.4. The hyperparameters used for testing the filtered datasets are the same as the full datasets.

Dataset	Reddit	Last.fm	Last.fm CET
Embedding size	50	100	100
Learning rate	0.001	0.001	0.001
Dropout rate	0.0	0.2	0.2
Max recent session representations	15	15	15
Mini-batch size	100	100	25
Number of GRU layers, intra-session level	1	1	1
Number of GRU layers, inter-session level	1	1	1
Type of session representations	LHS	AP	AP

Table 5.4: Hyperparameters for each dataset when using attention mechanisms. LHS stands for Last Hidden State and AP stands for Average Pooling.

To measure performance, two evaluation metrics were used, mean reciprocal rank and recall@k. The reciprocal rank of a ranking of items is the multiplicative inverse of the rank of the correct item (generally the *first* correct item, but in this domain there is only one). The mean reciprocal rank is the mean of the reciprocal rank of a collection of rankings.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{5.1}$$

where Q is a set of rankings and $rank_i$ is the ranking of the correct item. Recall measures the proportion of instances where the correct item was included in the ranking. This is combined with a minimum rank that the item must be at or above. For example, Recall@10 measures the proportion of items that were ranked in the top 10 of probabilities of being correct.

$$Recall@k = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \begin{cases} 1 & \text{if } rank_i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

These evaluations were performed on every item in the testing set, resulting in the scores presented in Section 5.4.

5.4 Experimental Results

This section will present the results for the different experimental models. Ruocco et al. [2017] uses the last hidden state method of creating session representations for the Reddit dataset and average pooling for the Last.fm dataset. Tests have shown that these are also the optimal methods to use when using attention mechanisms. Therefore, whenever results are presented for a specific dataset, the type of session representation is as such.

The number of permutations of models, model configurations and datasets is very high. Therefore, to avoid this chapter being nothing but tables, only the most relevant results will be presented. Also, in the interest of time and computing power available, only the most promising models/datasets have been tested with more than one RNG seed. Each metric is presented in the form of "*mean* ± *standard deviation*". If the standard deviation is not present, only one run was done of that test. If the standard deviation is present, the results are over five runs. The result of each run are based on the best epoch for the Recall@20 metric for that run.

5.4.1 Intra-Session Attention

The intra-session attention was an attempt at using the encoder-decoder text translation with attention architecture of Bahdanau et al. [2016] in a recommender system setting. The results can be seen in Table 5.5.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
Attention	0.3211 ± 0.001 (+0.1%)	0.3327 ± 0.001 (+0.1%)	0.3385 ± 0.001 (+0.1%)	0.4459 ± 0.0011 (-0.2%)	0.5324 ± 0.001 (-0.2%)	0.6159 ± 0.0012 (-0.2%)

(a) Reddit dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
Attention	0.0905 ± 0.0004 (-3.1%)	0.0981 ± 0.0004 (-3.0%)	0.1032 ± 0.0004 (-2.8%)	0.1444 ± 0.0007 (-3.2%)	0.2017 ± 0.0006 (-2.7%)	0.2762 ± 0.0007 (-2.2%)

(b) Last.fm dataset

Table 5.5: Intra-session attention experimental results.

As can be seen, the baseline outperforms the attention mechanism for both datasets. One reason for this might be that the assumption that text translation and recommender systems are similar is not actually true. There is much more structure in the grammar of a language than in a sequence of YouTube videos so maybe simply applying the same attention mechanism won't be as effective.

Another reason might be because session representations are one level of abstraction away from the actual actions of a user. What we mean by this is that

		Killswi… Collide Coldplay Collide Dredg A Perfe… Garbage A Perfe… Blackfield The Sma… Stabbin… Marilyn… Dashboa… God Liv… Type O … Amorphis Sneaker…		Bush Coldplay Audioslave Collide Dredg God Liv… Dredg Nine In… Saves T… Collide The Wal… A Perfe… Coldplay Dredg A Perfe… Nine In… Audioslave						Nirvana White W… Tool A Perfe… Him Marilyn… A Perfe… Katatonia Zwan Drain Sth Saves T… Weezer Evanesc… Stabbin… Tool Zwan		Zero 7 Evanesc… Coldplay Audioslave Tool Nine In… A Perfe… Nirvana Nine In… A Perfe… Audioslave Nine In… Pearl Jam Garbage Arise F… Deftones		The Sma… Theatre… Nine In… Audioslave Garbage Brave Bush The Sma… No Doubt Placebo A Perfe… Echo Image Vnv Natio… Tool Bush Radiohead Radiohead Vnv Natio… Bush Nine In… Marilyn… Michell… Afi Caw! Caw… Sin Blo… Amorphis Nirvana Namnambul… Nine In…	
	Coldplay Killswi…	Fiona A… God Liv…	Saves T… God Liv…	Dredg A Perfe…	A Perfe… Marilyn…	Tool Apoptyg…	Evanesc… Sneaker…	The Sma… Dashboa…	Me Firs… Vnv Nation	Placebo Veruca …	Weezer Arise F…	Stone T… Zwan	Tool 30 Seco…	Garbage Tiamat	Afi The Sma…
Him	0.0350	0.1865	0.1307	0.1731	0.1476	0.1210	0.0881	0.0404	0.0503	0.0157	0.0035	0.0008	0.0024	0.0026	0.0016
The Smashing Pump…	0.0520	0.1069	0.1118	0.1603	0.1376	0.1252	0.1056	0.0684	0.0800	0.0246	0.0089	0.0021	0.0057	0.0067	0.0034
Tool	0.0545	0.1212	0.1288	0.1529	0.1334	0.1091	0.0962	0.0729	0.0801	0.0271	0.0077	0.0019	0.0043	0.0062	0.0031
The Goo Goo Dolls	0.0633	0.1362	0.1323	0.1413	0.1227	0.0938	0.0966	0.0781	0.0798	0.0296	0.0081	0.0020	0.0047	0.0067	0.0042
Skold	0.0663	0.1337	0.1364	0.1436	0.1241	0.0948	0.0936	0.0721	0.0801	0.0290	0.0085	0.0020	0.0044	0.0066	0.0043
Him	0.0629	0.1424	0.1450	0.1500	0.1212	0.0920	0.0890	0.0650	0.0756	0.0282	0.0089	0.0023	0.0047	0.0068	0.0053
The Smashing Pump…	0.0671	0.1413	0.1416	0.1509	0.1180	0.1022	0.0870	0.0657	0.0746	0.0254	0.0076	0.0022	0.0053	0.0067	0.0037
Nine Inch Nails	0.0693	0.1415	0.1409	0.1493	0.1197	0.0942	0.0871	0.0663	0.0750	0.0280	0.0088	0.0022	0.0051	0.0072	0.0047
Him	0.0725	0.1442	0.1448	0.1439	0.1126	0.0912	0.0825	0.0656	0.0766	0.0304	0.0112	0.0027	0.0062	0.0087	0.0061
Him	0.0643	0.1460	0.1488	0.1419	0.1110	0.0880	0.0837	0.0665	0.0799	0.0331	0.0112	0.0027	0.0065	0.0092	0.0065
Him	0.0680	0.1460	0.1399	0.1394	0.1139	0.0973	0.0781	0.0705	0.0799	0.0317	0.0115	0.0025	0.0062	0.0090	0.0055
The Smashing Pump…	0.0714	0.1441	0.1405	0.1401	0.1147	0.0949	0.0786	0.0683	0.0798	0.0325	0.0116	0.0024	0.0060	0.0088	0.0055
Him	0.0658	0.1462	0.1445	0.1419	0.1143	0.0919	0.0801	0.0626	0.0793	0.0349	0.0125	0.0025	0.0063	0.0096	0.0068
Him	0.0669	0.1404	0.1404	0.1384	0.1180	0.0984	0.0783	0.0660	0.0818	0.0341	0.0126	0.0025	0.0062	0.0094	0.0058
Nine Inch Nails	0.0701	0.1470	0.1474	0.1444	0.1136	0.0891	0.0769	0.0647	0.0724	0.0317	0.0138	0.0031	0.0073	0.0101	0.0077
Tool	0.0653	0.1405	0.1458	0.1404	0.1144	0.0898	0.0805	0.0720	0.0789	0.0328	0.0137	0.0026	0.0064	0.0093	0.0069
Nine Inch Nails	0.0672	0.1441	0.1483	0.1441	0.1135	0.0891	0.0786	0.0682	0.0764	0.0316	0.0136	0.0027	0.0061	0.0090	0.0068
Marilyn Manson	0.0655	0.1471	0.1502	0.1457	0.1119	0.0834	0.0802	0.0708	0.0755	0.0312	0.0136	0.0025	0.0059	0.0086	0.0073
The Smashing Pump…	0.0674	0.1475	0.1494	0.1464	0.1119	0.0810	0.0804	0.0682	0.0738	0.0335	0.0133	0.0024	0.0061	0.0095	0.0083

Figure 5.2: Attention weights for the intra-session attention mechanism with the Last.fm dataset. The leftmost column shows the top prediction for each time step. The other columns contain one session representation each. The actions in each session are displayed at the top while the weights below say how much that session representation was weighted when predicting the action on the left.

we are trying to predict single actions by using representations of sequences of actions. In contrast, in the translation example the inputs are words and the outputs are words. Because of this the session representations might not lend themselves to being used for predicting single actions.

While training the network, attention weights were recorded. An example of the intra-session attention weights for the Last.fm dataset can be seen in Figure 5.2. In the vast majority of cases, the intra-session attention mechanism simply prioritizes the more recent sessions over older ones. However, there is one notable observation worth discussing.

Even though it is the most recent session, the left-most session representation is weighted significantly lower than several of the older ones. It can be seen in

several examples that short sessions, in particular those with only two actions, are weighted significantly lower than longer ones. This is not universally true, but it seems to hold more often than not.

It seems that shorter sessions are heavily penalized. One might think that this is because session representations are created in a way that makes the values of the representation have higher magnitude when there are more actions in the session. However, the Last.fm session representations are created using the average pooling method so the length of the session shouldn't matter. More intra-session attention weights can be seen in Appendix 1 and 2.

5.4.2 Inter-Session Attention

The inter-session attention mechanism had three different types of attention that could be used separately or combined to create the initial hidden state for a new session. The hidden attention is based on the hidden state of the inter-session RNN. The delta-t attention is based on the time difference between session representations and the new session. The week time attention tries to learn if sessions at certain points of the week are more important than others. Because the week time attention requires a different dataset, the results for that attention mechanism is shown in a table separate from the others. The non-week-time attention results are shown in Table 5.6 and the results for the week-time attention are shown in Table 5.7.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
Hidden	0.319 ± 0.0024 (-0.6%)	0.3309 ± 0.0022 (-0.5%)	0.3368 ± 0.0022 (-0.4%)	0.4471 ± 0.0018 (+0.1%)	0.5365 ± 0.0013 (+0.5%)	0.6209 ± 0.0014 (+0.6%)
Delta-t	0.3188 ± 0.0016 (-0.6%)	0.3306 ± 0.0015 (-0.5%)	0.3365 ± 0.0014 (-0.5%)	0.4455 ± 0.0011 (-0.2%)	0.5337 ± 0.0005 (0.0%)	0.6182 ± 0.0006 (+0.2%)
Hidden+Delta-t	0.3194 ± 0.0031 (-0.4%)	0.3313 ± 0.0029 (-0.3%)	0.3373 ± 0.0028 (-0.3%)	0.4481 ± 0.0021 (+0.3%)	0.5374 ± 0.001 (+0.7%)	0.6227 ± 0.0011 (+0.9%)

(a) Reddit dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
Hidden	0.0942 ± 0.0002 (+0.9%)	0.1018 ± 0.0003 (+0.7%)	0.107 ± 0.0003 (+0.8%)	0.1502 ± 0.0003 (+0.7%)	0.2083 ± 0.0004 (+0.5%)	0.2836 ± 0.0005 (+0.5%)
Delta-t	0.0938 ± 0.0003 (+0.4%)	0.1015 ± 0.0003 (+0.4%)	0.1067 ± 0.0003 (+0.5%)	0.1497 ± 0.0004 (+0.3%)	0.2079 ± 0.0006 (+0.3%)	0.2832 ± 0.0006 (+0.3%)
Hidden+Delta-t	0.094 ± 0.0007 (+0.6%)	0.1018 ± 0.0007 (+0.7%)	0.107 ± 0.0006 (+0.8%)	0.1502 ± 0.0007 (+0.7%)	0.2089 ± 0.0003 (+0.8%)	0.2849 ± 0.0005 (+0.9%)

(b) Last.fm dataset

Table 5.6: Inter-session attention experimental results.

The results for the non-week-time attention mechanisms show that all attention mechanisms outperform the baseline and that the hidden + delta-t model has the highest accuracy. In addition to these results, we have also visualized

some attention weights from various models. One of these is the hidden attention model for the Reddit dataset. These visualizations can be seen in Figure 5.3. Additional visualizations of the inter-session hidden attention mechanism with Last.fm can be seen in Appendix 3.

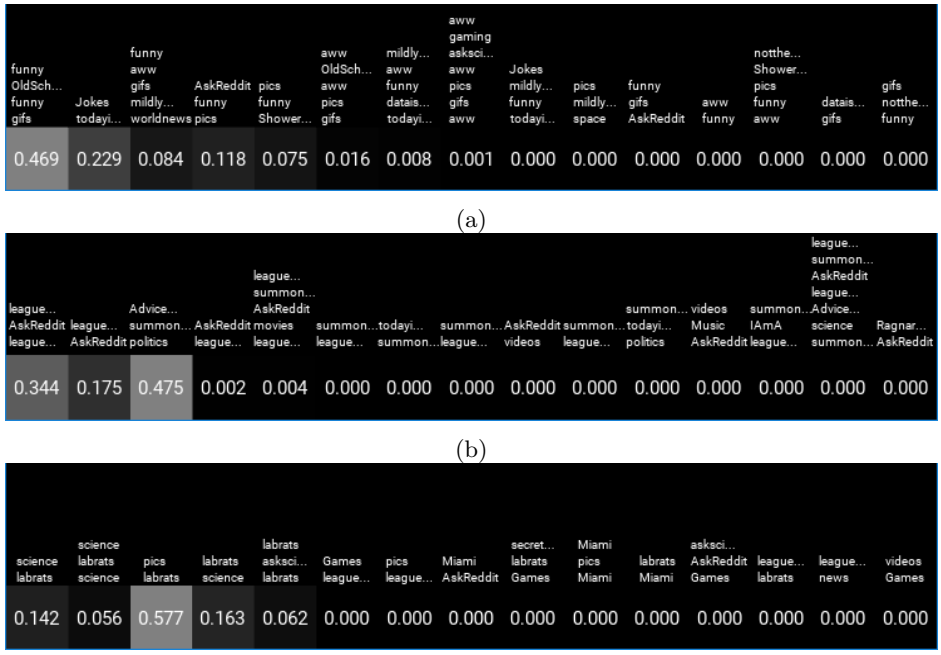


Figure 5.3: Attention weights for the hidden inter-session attention mechanism using the Reddit dataset. The top row shows the session representations, one for each column. The bottom row shows the attention weight for each session representation.

One of the first things we can see in Figure 5.3a is that there is a gradual decline in weights from left to right (most recent session to older ones). Similarly to the intra-session model, this model seems to also prioritize more recent sessions. However, this is not the only pattern we observe.

In Figure 5.3b we can see that even though newer sessions are prioritized, older sessions can gain a higher weight with the right content. This can be seen even more clearly in Figure 5.3c, though it is not immediately clear what causes a session representation to get a higher weight. These session representations are created using the last hidden state method, meaning that how the different

actions in a session combine into a session representation is more complicated than when using average pooling. For instance, two sessions (funny, AskReddit) and (AskReddit, funny) will have different session representations and thus might have wildly different attention weights.

In Section 4.2.2, we hypothesized that two events that are separated by a 24-multiple of hours (24 hours, 48 hours etc.) are more related than those that are not. In Figure 5.4, we can see some attention weights from a run of delta-t attention on the Reddit dataset.

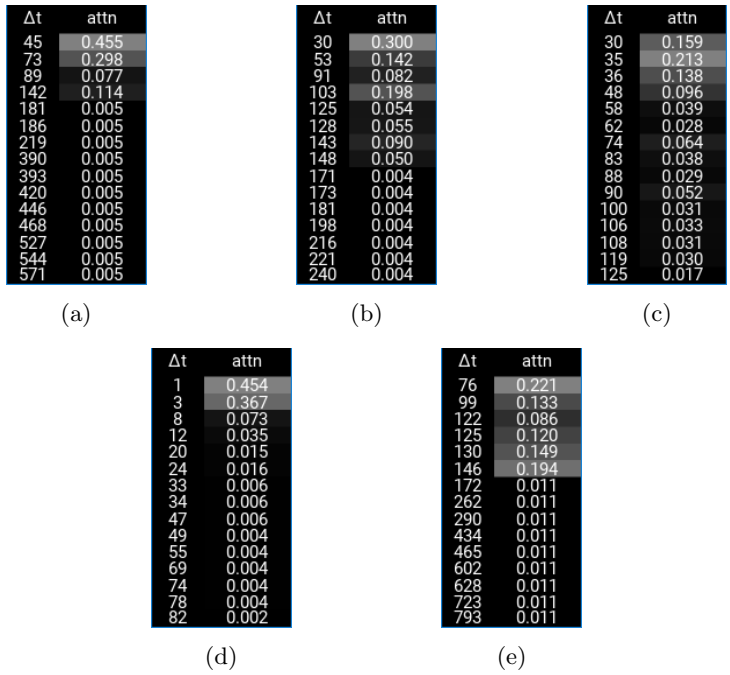


Figure 5.4: Attention weights for the delta-t inter-session attention mechanism using the Reddit dataset. The left column in each figure is the delta-t, the number of hours from that session to the new one we're trying to predict. The right column is the weight the attention mechanism assigned to that session.

Recall that the delta-time is sorted into buckets, each one hour long, for a total of 168 buckets and a final one for everything older than 168 hours. That is why all the attention weights for sessions more than 168 hours old are the same.

In Figure 5.4a, we can see that the event that is roughly 6 days ago (142 hours) is more relevant than the one that is a bit less than 4 days ago (89 hours).

This could suggest that there is some merit to the theory, but this is definitely not always the case. In Figure 5.4b we see that the session 103 hours ago is much higher weighted than the one 91 hours ago even though it is not near a 24-multiple of hours. Figure 5.4c shows the 24-hour property again. Here, 74 and 90 are higher weighted than their neighbors, but they are still very small in magnitude. They pale in comparison to the most recent sessions.

That is perhaps the most obvious observation one can make from these attention weights. While there may or may not be a correlation between a 24-multiple hour difference between sessions and the attention weights, there is definitely a correlation between how recent a session is and its attention weight. It can most easily be seen in Figure 5.4d that very recent sessions are extremely important when suggesting newer ones. Every session that is at least 20 hours old is practically ignored because the two most recent are so dominant. This could suggest that the empirically selected time that decides whether two events are part of the same session is suboptimal. It could also mean that we need a better model for time differences between sessions than the one hour bucket system.

Figure 5.4e does not follow any of these patterns, but has an interesting pattern nonetheless. In this one, the most recent session is over three days old so the sessions aren't dominated by a few recent ones. Although there are some variations in the attention weights, it seems to suggest that when there are no very recent sessions, all session representations should count roughly equally.

In conclusion, there might be a small difference between sessions that land on the 24 multiple number of hours away from a new session, but by far the most important sessions are those that are very recent.

The week-time attention results suffer from only being tested on one small dataset. As can be seen in Table 5.7, all the models have a higher Recall@20 than the baseline, but this difference is very small.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.1004	0.1076	0.1123	0.1568	0.2119	0.2793
Week-t	0.0994 (-1.0%)	0.1067 (-0.8%)	0.1115 (-0.7%)	0.1561 (-0.4%)	0.2114 (-0.2%)	0.2806 (+0.5%)
Hidden + Week-t	0.1005 (+0.1%)	0.1078 (+0.2%)	0.1125 (+0.2%)	0.157 (+0.1%)	0.2124 (+0.2%)	0.2804 (+0.4%)
Delta-t + Week-t	0.1001 (-0.3%)	0.1074 (-0.2%)	0.1121 (-0.2%)	0.1563 (-0.3%)	0.2112 (-0.3%)	0.2798 (+0.2%)
Hidden+Delta-t+Week-t	0.1012 (+0.8%)	0.1085 (+0.8%)	0.1132 (+0.8%)	0.1567 (-0.1%)	0.2118 (-0.0%)	0.281 (+0.6%)

Table 5.7: Inter-session attention experimental results with the Last.fm CET dataset.

This is the problem of only having one small dataset, it is hard to know if a model really is outperforming the baseline or not. Luckily, we have attention

Sun 13:00	0.071
Sat 16:00	0.074
Fri 10:00	0.074
Thu 18:00	0.075
Thu 16:00	0.072
Thu 10:00	0.064
Thu 8:00	0.063
Wed 20:00	0.072
Tue 18:00	0.067
Tue 12:00	0.056
Mon 16:00	0.063
Sun 17:00	0.070
Fri 17:00	0.068
Thu 19:00	0.053
Thu 14:00	0.059

(a)

Tue 17:00	0.070
Tue 15:00	0.068
Tue 13:00	0.063
Tue 12:00	0.069
Tue 10:00	0.069
Tue 8:00	0.066
Mon 23:00	0.061
Mon 21:00	0.072
Mon 20:00	0.070
Mon 15:00	0.060
Mon 13:00	0.066
Sat 15:00	0.069
Sat 13:00	0.063
Sat 12:00	0.073
Fri 21:00	0.061

(b)

Figure 5.5: Attention weights for the week-time inter-session attention mechanism using the Last.fm dataset. The left column shows the time of the week rounded to the nearest hour for each session. The right column is the weight the attention mechanism assigned to that session.

weight visualizations here as well and in Figure 5.5 they show a very clear picture. All the attention weights are pretty much uniform, suggesting that the attention mechanism is unable to learn a real connection between the time of the week that a session started and the importance of that session. This pattern holds true for all sets of week-time attention weights we have looked at.

5.4.3 Combined Inter-Session and Intra-Session Attention

The combined attention mechanism was an experiment to see if the intra-session and inter-session attention mechanisms could be improved by combining them. We performed one test per dataset. For the Reddit dataset we combined the intra attention with the inter hidden attention. For the Last.fm dataset we combined the intra attention with the inter hidden + delta-t attention. Lastly we tested with the Last.fm CET dataset by combining the intra attention with all three inter attentions. The results for the combined attention mechanism can be seen in Table 5.8.

It seems that for both the Reddit and Last.fm datasets, the results from the combined attention are somewhere between the intra attention and the corresponding inter attention. This suggests that the combination of the intra and inter attention mechanisms is no greater than the sum of its parts. The inter attention, which performed well, pulls it up and the intra attention, which performed poorly, pulls it down. In the end, the combined attention lies somewhere in between.

Not only does the ability to jointly learn attention weights not seem to help

increase accuracy, but more importantly, the combined attention performs worse than their respective baselines.

	MRR@5	MRR@10	MRR@20	R@5	R@10	R@20
Baseline	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
Intra	0.3211 ± 0.001 (+0.1%)	0.3327 ± 0.001 (+0.1%)	0.3385 ± 0.001 (+0.1%)	0.4459 ± 0.0011 (-0.2%)	0.5324 ± 0.001 (-0.2%)	0.6159 ± 0.0012 (-0.2%)
Hidden	0.319 ± 0.0024 (-0.6%)	0.3309 ± 0.0022 (-0.5%)	0.3368 ± 0.0022 (-0.4%)	0.4471 ± 0.0018 (+0.1%)	0.5365 ± 0.0013 (+0.5%)	0.6209 ± 0.0014 (+0.6%)
Intra+Hidden	0.3194 (-0.4%)	0.3312 (-0.4%)	0.337 (-0.4%)	0.4454 (-0.3%)	0.533 (-0.1%)	0.6168 (-0.1%)

(a) Reddit dataset

	MRR@5	MRR@10	MRR@20	R@5	R@10	R@20
Baseline	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
Intra	0.0905 ± 0.0004 (-3.1%)	0.0981 ± 0.0004 (-3.0%)	0.1032 ± 0.0004 (-2.8%)	0.1444 ± 0.0007 (-3.2%)	0.2017 ± 0.0006 (-2.7%)	0.2762 ± 0.0007 (-2.2%)
Hidden+Delta-t	0.094 ± 0.0007 (+0.6%)	0.1018 ± 0.0007 (+0.7%)	0.107 ± 0.0006 (+0.8%)	0.1502 ± 0.0007 (+0.7%)	0.2089 ± 0.0003 (+0.8%)	0.2849 ± 0.0005 (+0.9%)
Intra+Hidden+Delta-t	0.0902 (-3.4%)	0.0978 (-3.3%)	0.1029 (-3.1%)	0.1446 (-3.1%)	0.202 (-2.5%)	0.2767 (-2.0%)

(b) Last.fm dataset

	MRR@5	MRR@10	MRR@20	R@5	R@10	R@20
Baseline	0.1004	0.1076	0.1123	0.1568	0.2119	0.2793
Hidden+Delta-t+Week-t	0.1012 (+0.8%)	0.1085 (+0.8%)	0.1132 (+0.8%)	0.1567 (-0.1%)	0.2118 (-0.0%)	0.281 (+0.6%)
Intra+Hidden+Delta-t+Week-t	0.0952 (-5.2%)	0.1024 (-4.8%)	0.107 (-4.7%)	0.1491 (-4.9%)	0.203 (-4.2%)	0.2712 (-2.9%)

(c) Last.fm CET dataset

Table 5.8: Combined attention experimental results. The top entry in each table is the baseline and the bottom entry is the combined attention. The intra attention and inter attention results are shown for reference (the intra attention results for the Last.fm CET dataset is missing as the dataset was created specifically for testing the Week-t inter attention mechanism).

Note that this model was made and tested before the hierarchical model was implemented. While an intra + hierarchical combined attention is possible, it was decided to be a non-priority as the aforementioned tests showed no improvement when combining attention mechanisms.

5.4.4 Hierarchical Attention

As mentioned in the experimental plan, we want to test two things in the hierarchical model. First, if the hierarchical model without attention outperforms the baseline without attention. Second, if the hierarchical model with attention outperforms the hierarchical model without attention.

The results are presented in Table 5.9. The hierarchical model without attention outperforms the baseline model quite significantly. This is expected as

	MRR@5	MRR@10	MRR@20	R@5	R@10	R@20
Baseline	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
Hierarchical (no attn)	0.3339 ± 0.0013 (+4.1%)	0.3452 ± 0.0012 (+3.9%)	0.3508 ± 0.0012 (+3.7%)	0.4609 ± 0.0013 (+3.2%)	0.5453 ± 0.001 (+2.2%)	0.6255 ± 0.0013 (+1.3%)
Hierarchical (attention)	0.3291 ± 0.002 (+2.6%)	0.3405 ± 0.0019 (+2.4%)	0.3462 ± 0.0019 (+2.3%)	0.4563 ± 0.0006 (+2.2%)	0.5413 ± 0.001 (+1.4%)	0.6227 ± 0.0011 (+0.9%)
Hierarchical (delta-t attention)	0.3246 ± 0.0027 (+1.2%)	0.3363 ± 0.0026 (+1.2%)	0.3421 ± 0.0026 (+1.1%)	0.4567 ± 0.0017 (+2.3%)	0.5443 ± 0.0008 (+2.0%)	0.6262 ± 0.0007 (+1.5%)

(a) Reddit dataset

	MRR@5	MRR@10	MRR@20	R@5	R@10	R@20
Baseline	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
Hierarchical (no attn)	0.0946 ± 0.0002 (+1.3%)	0.1024 ± 0.0002 (+1.3%)	0.1076 ± 0.0002 (+1.3%)	0.1514 ± 0.0004 (+1.5%)	0.2109 ± 0.0003 (+1.8%)	0.2876 ± 0.0005 (+1.9%)
Hierarchical (attention)	0.0925 ± 0.0004 (-1.0%)	0.1002 ± 0.0004 (-0.9%)	0.1054 ± 0.0004 (-0.8%)	0.1473 ± 0.0005 (-1.3%)	0.2056 ± 0.0005 (-0.8%)	0.2811 ± 0.0005 (-0.4%)

(b) Last.fm dataset

Table 5.9: Hierarchical attention experimental results.

the hierarchical model is not storing and retrieving session representations, but is instead creating them each time they are needed. This means that the session representations are no longer based on old weights in the embedding matrix or the intra-session GRU. The new session representations are more representative of the events they consist of.

In the results for the hierarchical model with attention we see some interesting results. The Reddit dataset performs quite well compared to the baseline, about on par with the hidden + delta-t inter attention. However, when comparing to the hierarchical model without attention it actually performs worse. The Last.fm dataset performs even worse, with an accuracy lower than that of the baseline.

The reason for the bad accuracy with Last.fm is likely due to the way session representations are created when using the hierarchical attention mechanism. Recall that there are two ways to create session representations: last hidden state and average pooling. In all our experiments so far, Reddit has used the last hidden state method while Last.fm has used the average pooling method as this has shown to give the best results. However, in order to have an attention mechanism when creating session representations, the last hidden state method has to be used. This means that the Last.fm dataset has to use the worst performing of the two methods and its accuracy suffers as a result.

As mentioned in Section 4.2.4, the hierarchical model can also use the temporal attention mechanisms of the inter-session model when creating user representations. Because the Last.fm dataset performed poorly with this model we decided to focus on the Reddit dataset. We also chose to skip the week-t attention mechanism as it did not improve the inter-session attention model. In Table 5.9, the model labeled Hierarchical (delta-t attention) is equivalent to the inter-session hidden + delta-t attention, but with the addition of the extra layer of attention mechanism when creating session representations introduced by the

hierarchical model. This model finally beats the hierarchical model without attention. This is also the highest (percentage-wise) increase in accuracy that we have seen from augmenting an attention mechanism with a temporal component.

The hierarchical model brings a new level of attention and with it some new attention weight visualizations. When creating session representations the model will try to learn which events are more important than others and weight them accordingly. Some attention weights can be seen in Figure 5.6.

Event	Attn
Black Flag	0.029
The Makers	0.028
Poison Idea	0.031
68 Comeback	0.077
The Devil Dogs	0.056
Caw! Caw!	0.014
M.O.T.O.	0.047
Bones Brigade	0.036
7 Seconds	0.024
Crime	0.096
Jay Reatard	0.030
Bbq	0.074
Registrators	0.082
The Zombies	0.022
Simon & Garfunkel	0.021
The Maggots	0.073
The Little Killers	0.107
Clorox Girls	0.045
Bantam Rooster	0.056
Blacktop	0.050

(a)

Event	Attn
Chromatics	0.065
Glass Candy	0.072
Radiohead	0.053
Modeselektor	0.067
Radiohead	0.038
Modeselektor	0.061
Radiohead	0.040
Modeselektor	0.058
Radiohead	0.037
Modeselektor	0.046
Radiohead	0.042
Modeselektor	0.047
Radiohead	0.035
Modeselektor	0.048
Radiohead	0.040
Modeselektor	0.057
Radiohead	0.034
Modeselektor	0.045
Radiohead	0.042
Modeselektor	0.074

(b)

Event	Attn	Sub Count
techsupport	0.139	262,000
JUSTNOMIL	0.132	266,000
techsupport	0.111	262,000
politics	0.106	3,852,000
politics	0.082	3,852,000
politics	0.058	3,852,000
WTF	0.052	5,144,000
WTF	0.050	5,144,000
pics	0.043	18,733,000
AskReddit	0.039	19,340,000
WTF	0.035	5,144,000
politics	0.034	3,852,000
politics	0.032	3,852,000
funny	0.030	19,672,000
AskReddit	0.026	19,340,000
videos	0.026	17,839,000

(c)

Figure 5.6: Attention weights for the hierarchical attention mechanism on the session representation level using the Last.fm dataset. The left column shows all the events of a single session. The right column is the weight the attention mechanism assigned to that event. In Figure (c) the third column is the amount of subscribers for that subreddit as of 06.06.18.

It seems that, in contrast to most of the other attention mechanism visualizations we have looked at, these attention weights aren't simply ordered by age. We can see that in Figure 5.6a the attention weights are all over the place. This makes sense as the time between each single event is very short. Figure 5.6b shows a session that almost exclusively alternates between Radiohead and Modeselektor. In this case we see that the Modelselektor events are always weighted higher than the Radiohead event right before. This seems to show that there is some prioritization of artists going on.

It seems that some items have consistently higher weights than others. These items, however are often somewhat unpopular so the sample size is quite small. On the flipside, some of the most popular subreddits such as AskReddit and funny seem to be consistently lowly weighted. Figure 5.6c shows an interesting

example of this. In it we have sorted the events by the attention weights and next to each event is the subscriber count of the given subreddit. What we can see is that smaller subreddits tend to be weighted higher than more popular ones.

This isn't a perfect fit of course, but remember that the attention weights are based on GRU outputs so the attention weights are based on not only the current item (the item to the left of the weight), but also all other items earlier in the session. This makes it a bit harder to interpret what's going on. A few examples of attention weights given to the AskReddit subreddit is shown in Appendix 5.

5.4.5 Filtered Datasets

As mentioned in the beginning of this chapter, there are two things we want to know when using these datasets. First, will they outperform the full datasets. Second, will they benefit more from attention mechanisms than the full datasets. This section features 14 different datasets so to look at each of them with every attention mechanism would be madness. Instead we will present the baselines for all of the datasets, then we will show the results for the best Last.fm dataset and the best Reddit dataset with some attention mechanisms as well.

Limited Time-Interval Datasets

We wanted to see if limiting the time interval in which we look at user history would help predictions. The baseline results can be seen in Table 5.10.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Full dataset	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
1 Month	0.347 (+8.2%)	0.3588 (+7.9%)	0.3645 (+7.7%)	0.4756 (+6.5%)	0.5638 (+5.7%)	0.645 (+4.5%)
2 Month	0.3468 (+8.1%)	0.3588 (+7.9%)	0.3646 (+7.8%)	0.4791 (+7.3%)	0.5685 (+6.5%)	0.6508 (+5.4%)
3 Month	0.3416 (+6.5%)	0.3534 (+6.3%)	0.3591 (+6.1%)	0.4736 (+6.0%)	0.5619 (+5.3%)	0.6437 (+4.3%)

(a) Reddit dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Full dataset	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
1 Month	0.101 (+8.1%)	0.1081 (+6.9%)	0.1127 (+6.1%)	0.1553 (+4.1%)	0.2092 (+1.0%)	0.2766 (-2.0%)
2 Month	0.1033 (+10.6%)	0.111 (+9.8%)	0.1159 (+9.1%)	0.1638 (+9.8%)	0.2217 (+7.0%)	0.2934 (+3.9%)
3 Month	0.1065 (+14.0%)	0.1146 (+13.4%)	0.1199 (+12.9%)	0.1682 (+12.7%)	0.2294 (+10.7%)	0.3058 (+8.3%)

(b) Last.fm dataset

Table 5.10: Baseline results for the limited time-interval datasets.

It is very clear that the time-filtered datasets outperform the full datasets. This makes sense as a user's interests change over time, so trying to learn the

user's entire history makes things more difficult. In fact, it hurts the accuracy of the model. The only time-filtered dataset that does not outperform the baseline is the Last.fm 1 month dataset. This can likely be attributed to the fact that the dataset is so small that it cannot achieve the same accuracy as the full dataset.

While these are interesting results, what we are most interested in is whether applying attention mechanisms to these datasets will increase accuracy more than with the full datasets. The temporal attention mechanism in particular is very interesting in this case as all the events happen over a shorter time period. Table 5.11 shows the results for the Reddit 2 month and Last.fm 3 month datasets for some attention mechanisms.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
2 month baseline	0.3468	0.3588	0.3646	0.4791	0.5685	0.6508
Intra attention	0.3457 (-0.3%)	0.3571 (-0.5%)	0.3628 (-0.5%)	0.474 (-1.1%)	0.5596 (-1.6%)	0.6405 (-1.6%)
Hidden+Delta-t attention	0.3457 (-0.3%)	0.3579 (-0.3%)	0.3637 (-0.2%)	0.4836 (+0.9%)	0.5742 (+1.0%)	0.6569 (+0.9%)
Hierarchical attention	0.3475 (+0.2%)	0.3589 (0.0%)	0.3644 (-0.1%)	0.4767 (-0.5%)	0.562 (-1.1%)	0.6417 (-1.4%)

(a) Reddit 2 month dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
3 month baseline	0.1065	0.1146	0.1199	0.1682	0.2294	0.3058
Intra attention	0.1028 (-3.5%)	0.1109 (-3.2%)	0.1161 (-3.2%)	0.1649 (-2.0%)	0.2263 (-1.4%)	0.302 (-1.2%)
Hidden+Delta-t attention	0.1071 (+0.6%)	0.1151 (+0.4%)	0.1204 (+0.4%)	0.1704 (+1.3%)	0.231 (+0.7%)	0.3076 (+0.6%)
Hierarchical attention	0.1061 (-0.4%)	0.1142 (-0.3%)	0.1193 (-0.5%)	0.1667 (-0.9%)	0.2279 (-0.7%)	0.3024 (-1.1%)

(b) Last.fm 3 month dataset

Table 5.11: Attention results for the limited time-interval datasets.

While the limited time-interval baselines are clearly better than the old baselines in terms of accuracy, the attention mechanisms show much of the same story as with the full datasets. As before, the intra attention mechanism is the worst of the bunch. The hidden + delta-t attention mechanism performs percentage-wise about the same as for the full dataset. As for the hierarchical attention it seems that Last.fm 3 month performs about the same as the hierarchical attention with the full dataset, but Reddit 2 month performs much worse. Our hope was that the temporal attention mechanism in particular would see improvement when we only used sessions in a small time interval, but it seems that the limited time-interval datasets do not benefit more from the temporal attention mechanism than the full datasets.

User Statistics-Filtered Datasets

Our main theory with these datasets was that the datasets containing only users with above average session count and above average session length would perform better. This is because we filter out the users with only a few sessions that contribute nothing but noise. The only users left are those with long sessions and many of them. Alas, it didn't help. Table 5.12 shows the baseline results for the filtered datasets. A reason that we see a decrease in accuracy might be because there isn't enough data to train on. As can be seen with the Last.fm 1 month dataset above, smaller datasets tend to perform worse than the larger ones. If you have even less data, you might run into the exact opposite problem, as can be seen in the results for Reddit high-low. This result in particular is remarkable, but not reliable as the dataset used is incredibly small.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.3208 ± 0.0012	0.3324 ± 0.0011	0.3383 ± 0.001	0.4466 ± 0.001	0.5336 ± 0.0011	0.6172 ± 0.0019
High-High	0.3459 (+7.8%)	0.3544 (+6.6%)	0.3588 (+6.1%)	0.4414 (-1.2%)	0.5054 (-5.3%)	0.5697 (-7.7%)
High-Low	0.2832 (-11.7%)	0.2946 (-11.4%)	0.3006 (-11.1%)	0.4065 (-9.0%)	0.4918 (-7.8%)	0.5781 (-6.3%)
Low-High	0.485 (+51.2%)	0.4948 (+48.9%)	0.4993 (+47.6%)	0.6114 (+36.9%)	0.6841 (+28.2%)	0.7485 (+21.3%)
Low-Low	0.2868 (-10.6%)	0.3004 (-9.6%)	0.3069 (-9.3%)	0.4244 (-5.0%)	0.5261 (-1.4%)	0.6198 (+0.4%)

(a) Reddit dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Baseline	0.0934 ± 0.0001	0.1011 ± 0.0001	0.1062 ± 0.0001	0.1492 ± 0.0001	0.2072 ± 0.0003	0.2823 ± 0.0003
High-High	0.1193 (+27.7%)	0.125 (+23.6%)	0.1288 (+21.3%)	0.1659 (+11.2%)	0.2095 (+1.1%)	0.2645 (-6.3%)
High-Low	0.1061 (+13.6%)	0.1131 (+11.9%)	0.1177 (+10.8%)	0.1599 (+7.2%)	0.2134 (+3.0%)	0.2807 (-0.6%)
Low-High	0.0739 (-20.9%)	0.0806 (-20.3%)	0.0855 (-19.5%)	0.1206 (-19.2%)	0.1721 (-16.9%)	0.2427 (-14.0%)
Low-Low	0.0791 (-15.3%)	0.0865 (-14.4%)	0.0917 (-13.7%)	0.1311 (-12.1%)	0.1874 (-9.6%)	0.2628 (-6.9%)

(b) Last.fm dataset

Table 5.12: Baseline results for the statistics filtered datasets. "high-low" means that the dataset has removed all users with above average session length and/or below average session count. Note that the Reddit low-high dataset is very small so the increase in accuracy should be ignored.

For the attention mechanisms we present the results for the Reddit low-low and Last.fm high-low datasets using the same attention mechanisms as for the time-limited datasets. The results are shown in Table 5.13. Testing the filtered datasets with the intra attention mechanism, we see mostly what we saw with intra attention on the full datasets and the time-limited datasets; the baseline outperforming the attention mechanism. The one exception to this (not shown

in table) is the Reddit high-high dataset (Reddit where all users with higher than average session length and/or higher than average session count have been removed). In this dataset, each metric in the intra attention mechanism is equal to or slightly greater than their baseline counterparts. This, while interesting, is ultimately pointless as the baseline results for this dataset is much worse than the full dataset. The results for hidden + delta-t and hierarchical attention are similar as those for the time-limited datasets.

The fact that these datasets, both time-limited and statistics-filtered, didn't increase attention accuracy might be a sign that it is not the data itself that doesn't fit well with attention mechanism, but rather the problem domain. It seems that even if your dataset only consists of "ideal" users, the attention mechanism is unable to improve accuracy any more than the full datasets.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Low-low baseline	0.2868	0.3004	0.3069	0.4244	0.5261	0.6198
Intra attention	0.2867 (-0.0%)	0.2998 (-0.2%)	0.3063 (-0.2%)	0.4233 (-0.3%)	0.5208 (-1.0%)	0.6137 (-1.0%)
Hidden+Delta-t attention	0.2879 (+0.4%)	0.3021 (+0.6%)	0.3086 (+0.6%)	0.4305 (+1.4%)	0.5366 (+2.0%)	0.6292 (+1.5%)
Hierarchical attention	0.2822 (-1.6%)	0.2948 (-1.9%)	0.3012 (-1.9%)	0.4155 (-2.1%)	0.5101 (-3.0%)	0.6012 (-3.0%)

(a) Reddit low-low dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
High-low baseline	0.1061	0.1131	0.1177	0.1599	0.2134	0.2807
Intra attention	0.1004 (-5.4%)	0.1073 (-5.1%)	0.1119 (-4.9%)	0.1513 (-5.4%)	0.2033 (-4.7%)	0.2705 (-3.6%)
Hidden+Delta-t attention	0.1053 (-0.8%)	0.1124 (-0.6%)	0.1171 (-0.5%)	0.1591 (-0.5%)	0.2133 (-0.0%)	0.2812 (+0.2%)
Hierarchical attention	0.1019 (-4.0%)	0.109 (-3.6%)	0.1138 (-3.3%)	0.1538 (-3.8%)	0.2078 (-2.6%)	0.2774 (-1.2%)

(b) Last.fm high-low dataset

Table 5.13: Attention results for the statistics filtered datasets.

5.4.6 Per-User Attention Weights

When testing per-user attention weights we encountered several problems. Firstly the fact that it is incredibly slow to train. This is because in PyTorch, at least as far as we could figure out, there is no way to pass each item in a mini-batch through a different linear layer while still keeping the benefits of parallelization. Therefore, the parts of the model that calculated attention weights had to be done one by one.

The more serious problem was the amount of linear layers we needed in total

for the model, especially for the Reddit dataset. The attention mechanism uses two linear layers. The hierarchical model has two levels of attention. If using per-user attention weights, that's four linear layers per user. Multiply that by 18271 users (Reddit) and you have 73084 linear layers in a single model, which PyTorch does not appreciate. This resulted in the training slowing down over time.

Ways to work around these issues with per-user attention weights is discussed in Section 6.4, but for now we decided instead to test the per-user attention weights on some of the smaller datasets that we had created earlier. We tested the per-user attention weights on the Last.fm 3 month dataset as this was the best performing dataset as well as the Reddit low-low dataset as this dataset performed decently while having a manageable amount of users. The results are in Table 5.14. Even the Reddit low-low dataset couldn't handle the intra attention model with per-user attention weights so that result is missing.

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
Low-low baseline	0.2868	0.3004	0.3069	0.4244	0.5261	0.6198
Global hidden+delta-t	0.2879 (+0.4%)	0.3021 (+0.6%)	0.3086 (+0.6%)	0.4305 (+1.4%)	0.5366 (+2.0%)	0.6292 (+1.5%)
Per-user hidden+delta-t	0.2791 (-2.7%)	0.2926 (-2.6%)	0.299 (-2.6%)	0.4173 (-1.7%)	0.5179 (-1.6%)	0.6099 (-1.6%)
Global hierarchical	0.2822 (-1.6%)	0.2948 (-1.9%)	0.3012 (-1.9%)	0.4155 (-2.1%)	0.5101 (-3.0%)	0.6012 (-3.0%)
Per-user hierarchical	0.2753 (-4.0%)	0.2879 (-4.2%)	0.294 (-4.2%)	0.4063 (-4.3%)	0.5007 (-4.8%)	0.5885 (-5.1%)

(a) Reddit low-low dataset

	MRR@5	**MRR@10**	**MRR@20**	**R@5**	**R@10**	**R@20**
3 month baseline	0.1065	0.1146	0.1199	0.1682	0.2294	0.3058
Global intra	0.1028 (-3.5%)	0.1109 (-3.2%)	0.1161 (-3.2%)	0.1649 (-2.0%)	0.2263 (-1.4%)	0.302 (-1.2%)
Per-user intra	0.0993 (-6.8%)	0.1073 (-6.4%)	0.1124 (-6.3%)	0.1585 (-5.8%)	0.219 (-4.5%)	0.2941 (-3.8%)
Global hidden+delta-t	0.1071 (+0.6%)	0.1151 (+0.4%)	0.1204 (+0.4%)	0.1704 (+1.3%)	0.231 (+0.7%)	0.3076 (+0.6%)
Per-user hidden+delta-t	0.1048 (-1.6%)	0.1129 (-1.5%)	0.1182 (-1.4%)	0.1664 (-1.1%)	0.2282 (-0.5%)	0.3038 (-0.7%)
Global hierarchical	0.1061 (-0.4%)	0.1142 (-0.3%)	0.1193 (-0.5%)	0.1667 (-0.9%)	0.2279 (-0.7%)	0.3024 (-1.1%)
Per-user hierarchical	0.1059 (-0.6%)	0.1144 (-0.2%)	0.1198 (-0.1%)	0.1681 (-0.1%)	0.2324 (+1.3%)	0.3109 (+1.7%)

(b) Last.fm 3 month dataset

Table 5.14: Per-user attention weights experimental results. Global means global attention weights. Per-user means per-user attention weights.

All in all the results are very bad. The only instance where per-user attention

weights outperformed global attention weights was the hierarchical model with Last.fm 3 month. We can see that Last.fm 3 month outperformed Reddit low-low (percentage-wise) for both the hidden + delta-t attention as well as the hierarchical attention. The reason for this is likely the same reason as to why per-user attention weights performed so badly overall. There is simply not enough data per user to train the model. Looking at Table 5.2 and Table 5.3, we can find the average number of events per user in the two datasets discussed in this section. We find that users in the Reddit low-low dataset have an average of 426 events and users in the Last.fm dataset have an average of 812 events. Users in Last.fm 3 month have almost twice the events of the users in Reddit low-low. More data means more accuracy, but judging by the results we still need much more. This approach should in theory give higher accuracy, but in practice it is failing.

Chapter 6

Evaluation and Conclusion

In this chapter we summarize the results and the achievements of the project as a whole. We also present some possible paths to take this work further. Whenever accuracies are mentioned in this chapter we are referring to the Recall@20 metric.

6.1 Evaluation

The inter-session attention performed better than the baseline with the hidden + delta-t attention being the best with a +0.9% increase in accuracy in both datasets. A lot of the attention weights favored more recent sessions. This was seen in the intra-session model as well as the inter-session hidden and delta-t models. One visualization of the delta-t attention showed that very recent session representations (less than five hours old) seem to be very highly weighted. As mentioned in the previous chapter, this could mean that the time difference required for two events to be considered to be part of different sessions might be suboptimal. It could also suggest that the way we model the delta-t by putting the time difference in one hour buckets is too simple. A different approach could be to not have the time difference of each bucket grow linearly (one hour each), but instead grow quadratically (recent buckets are more granular and later buckets cover larger timespans). We also saw that if there were no very recent sessions then the session representations were weighted more equally, which would also fit into the quadratic delta-t bucket model.

The intra-session attention performed worse than baseline for both full datasets and equally bad for the smaller datasets and per-user attention weights. It is hard to see whether the intra-session attention prioritized certain events or not. This is because the attention mechanism is based on session representations. They are one level of abstraction away from the events themselves and so it is hard to see

which single items in the session representations are more important. We do see, however, that some of the session representations have high weights even when they're not the most recent so their contents must matter.

The inter-session week-t attention with the Last.fm CET dataset achieved a +0.6% increase over its baseline which, while better than baseline, fails to improve more than the hidden + delta-t attention did on the full dataset. It is also very clear from the attention weights that it learns little in its current implementation. This could suggest that there is no connection between time of week and user habits or it could suggest that the model is too simple. As stated in Section 1.2, temporal modeling is an entirely different area of research so expecting results with something as naive as this simple implementation can't be expected.

In the combined attention we saw that combining the intra and inter attention mechanisms did not increase accuracy more than they did on their own. One attention mechanism brings the accuracy up, the other brings it down. We can see a similar pattern with the inter-session hidden attention and delta-t attention. They individually increased accuracy by a bit, and together they increased it more.

The hierarchical baseline model achieved accuracy increases of +1.3% and +1.9% in the Reddit and Last.fm datasets respectively. As mentioned in the last chapter, this is likely due to the fact that each session representation is recreated every time it is needed. This means that the embeddings of events used to create the session representations are up to date with the latest weight changes in the embedding matrix and not based on old values. The hierarchical attention mechanism did not perform as well as we had hoped with +0.9% for Reddit and −0.4% for Last.fm. In particular the Last.fm hierarchical attention mechanism performed worse than the (original, non-hierarchical) baseline, but as discussed previously, this is likely due to having to use the last hidden state method of creating session representations. The hierarchical delta-t attention did improve the accuracy for Reddit with an increase of +1.5% over the baseline, beating the hierarchical model without attention. Judging by the attention weight visualizations it seems that some items tend to more highly weighted than others. The more popular subreddits are consistently weighted lowly. Therefore, generally, a subreddit will have more impact on a session representation if it has a smaller subscriber count.

The limited-time interval datasets showed some great results in their baselines. While both datasets have large accuracy increases with the time-limited datasets we see that Last.fm 3 month is the clear winner with a +8.3% increase over its baseline while Reddit 2 month has a +5.4% increase. The reason Last.fm 3 months sees a much larger accuracy increase than the Reddit 2 month dataset might be because a user's music taste changes faster than their interests in specific subreddits. Looking at it more globally, the music taste of the Last.fm

userbase will change over time as new songs are released, but while the Reddit userbase might discover new subreddits over time, they will still visit the older more established ones.

The statistics-filtered datasets did not perform as well as the limited-time interval datasets. We hypothesized that the datasets that removed all users with below average session length or below average session count would perform better. While this was the best performing dataset for Last.fm, the improvement was modest.

With both the time-filtered datasets and the statistics-filtered datasets we wanted to see if we could create an "ideal" dataset that was well suited for attention mechanisms. However, it seems that not having an ideal dataset is not the problem as these did not improve the accuracy of the attention mechanisms any more than with the full datasets.

With per-user attention weights we saw that the datasets do not have enough data on each user to be able to properly train its weights. We also saw that the model struggles on large datasets due to the amount of training needed.

6.2 Discussion

All in all the highest accuracy for Last.fm was achieved by Last.fm 3 month with hierarchical per-user attention. For Reddit, the highest was Reddit 2 month with hidden + delta-t attention. For the full datasets the highest accuracy was achieved by the inter-session hidden + delta-t attention for Last.fm and the hierarchical delta-t attention for Reddit.

While the highest accuracies were achieved by filtered datasets, the filtered datasets did not achieve their primary objective of being more suited for attention mechanisms. They achieved higher accuracies, but the relative increases in accuracy with attention mechanism did not increase versus the full datasets.

Therefore, it is not the datasets that are the problem. Bahdanau et al. [2016] has received a lot of praise for introducing attention mechanisms in text translation and others have followed in their footsteps by applying attention mechanisms to other problem areas as well. It is then natural that we should attempt the same methods with recommender systems. However, while the overall architecture of the hierarchical session-based recommender system introduced by Ruocco et al. [2017] is very similar to that of Bahdanau et al. [2016], the problem domains are not. Language has a very strict grammar and many rules. This means that after one word you can often very easily guess the next one. However, nobody will stop you from listening to Adele right after listening to Kanye West. The recommender system "grammar" is a lot more free flowing. Because of this it is harder to know which parts of the input are important for predicting which parts

of the output, which is the essence of the attention mechanism of Bahdanau et al. [2016].

There is also the issue of the abstraction between single events and session representations that we mentioned earlier. Bahdanau et al. [2016] translated words into words. Simple. We're trying to "translate" session representations into single events.

Despite these issues we have shown that the attention mechanisms based solely on hidden state do see a small increase in accuracy. We have also shown with the inter-session hidden + delta-t attention that a temporal attention mechanism can slightly increase accuracy. The attention weight visualizations also show clear patterns, which means we might just need a more sophisticated way of modeling the temporal aspect to see a more significant increase in accuracy.

Per-user attention weights has a laundry list of problems and clearly need more work, but should in theory be able to increase accuracy further. A method to fix these issues is described in Section 6.4.

6.3 Contributions

Our goal was to explore different ways that attention mechanisms can be applied to session based recommender systems for the task of increasing accuracy.

In particular, our first research question asked if attention mechanisms that have been developed for other applications of recurrent neural networks can be applied to hierarchical session-based recommender systems. We have shown that an attention mechanism based solely on the hidden stated of the encoder and decoder can be applied to recommender systems as well, but only with modest results.

We have also shown that we can achieve somewhat better results by augmenting these attention mechanisms with a temporal component. This relates to our second research question about whether the temporal aspect of recommender systems can be included in an attention mechanism to increase accuracy.

In addition, we have through visualizations of attention weights shown that the attention mechanisms do learn certain patterns, so it is definitely possible that other attention mechanisms can be applied to see greater results in the future.

6.4 Future Work

The model that consistently performed best was the inter-session temporal attention (hidden + delta-t). We used a very simple way of modeling the time between sessions by just putting the time differences into buckets of one hour

each. A more sophisticated model, such as scaling the time-difference between sessions quadratically instead of linearly or modeling the Delta-t someway different altogether would likely perform better.

While per-user attention weights might be promising in theory, they won't see any practical application in their current form. Instead of each user getting their own attention weight matrices, the users should be clustered. This could either be done based on their interests or by some other metric. This should also be done as a pre-training step. The benefit of clustering is two-fold. Firstly we avoid the issue of having a model with way too many linear layers in it. This is not only a problem because it causes issues with certain deep learning frameworks, but also because many users won't have enough data to properly train their weights anyways. The other benefit of this is that it allows us to parallelize the parts that were previously un-parallelizable by only including users from the same cluster in any given mini-batch. Clustering users would also allow us to experiment with per-cluster encoders and/or decoders as described in Section 4.3.

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 [cs.CL], May.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenkand, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1402.1454 [cs.CL], February.

Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. arXiv:1506.07503 [cs.CL], June.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. arXiv:1601.01073 [cs.CL], January.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks. arXiv:1511.06939 [cs.LG], March.

Jannach, D. and Ludewig, M. (2017). When recurrent neural networks meet the neighborhood for session-based recommendation. *RecSys '17 Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 306–310.

Li, J., Ren, P., Chen, Z., Ren, Z., and Ma, J. (2017). Neural attentive session-based recommendation. arXiv:1711.04725 [cs.IR], November.

Liu, Q., Wu, S., Wang, D., Li, Z., and Wang, L. (2016). Context-aware sequential recommendation. arXiv:1609.05787 [cs.IR], September.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv:1508.04025 [cs.CL], September.

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., and Cottrell, G. (2017). A dual-stage attention-based recurrent neural network for time series prediction. arXiv:1704.02971 [cs.LG], August.

Quadrana, M., Karatzoglou, A., Hidasi, B., and Cremonesi, P. (2017). Personalizing session-based recommendations with hierarchical recurrent neural networks. arXiv:1706.04148 [cs.LG], August.

Ruocco, M., Skrede, O. S. L., and Langseth, H. (2017). Inter-session modeling for session-based recommendation. arXiv:1706.07506 [cs.IR], June.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. arXiv:1509.00685 [cs.CL], September.

Tan, Y. K., Xu, X., and Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. arXiv:1606.08117 [cs.LG], September.

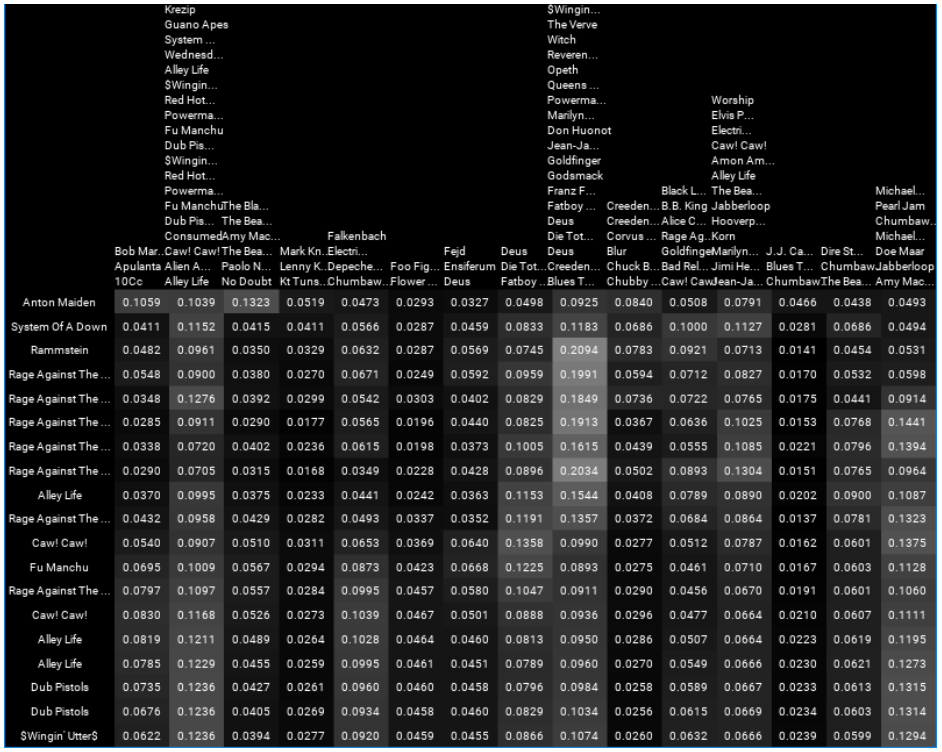
Tuan, T. X. and Phuong, T. M. (2017). 3d convolutional networks for session-based recommendation with content features. *RecSys '17 Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 138–146.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. arXiv:1411.4555 [cs.CV], April.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Appendices

1 Intra-Session Attention Weights for Last.fm



	Bob Mar.. Apulanta 10Cc	Caw! Caw! Alien A... Alley Life	The Bea... Paolo N... No Doubt	Mark Kn.. Lenny K.. Kt Tuns..	Falkenbach Electri... Depeche... Chumbaw..	Foo Fig... Flower ...	Fejd Ensiferum Deus	Deus Die Tot... Fatboy ..	Deus Die Tot... Creeden... Blues T...	Blur Chuck B... Chubby ...	Black L... B.B. King Alice C... Rage Ag.. Bad Rel...	Worship Elvis P... Electri... Caw! Caw! Amon Am... Alley Life The Bea... Jabberloop Hooverp... Korn Marilyn... Jimi He... Jean-Ja...	J. J. Ca... Blues T... Chumbaw..	Dire St... Chumbaw.. The Bea...	Michael... Pearl Jam Chumbaw.. Michael... Doe Maar Jabberloop Amy Mac...
Anton Maiden	0.1059	0.1039	0.1323	0.0519	0.0473	0.0293	0.0327	0.0498	0.0925	0.0840	0.0508	0.0791	0.0466	0.0438	0.0493
System Of A Down	0.0411	0.1152	0.0415	0.0411	0.0566	0.0287	0.0459	0.0833	0.1183	0.0686	0.1000	0.1127	0.0281	0.0686	0.0494
Rammstein	0.0482	0.0961	0.0350	0.0329	0.0632	0.0287	0.0569	0.0745	0.2094	0.0783	0.0921	0.0713	0.0141	0.0454	0.0531
Rage Against The ...	0.0548	0.0900	0.0380	0.0270	0.0671	0.0249	0.0592	0.0959	0.1991	0.0594	0.0712	0.0827	0.0170	0.0532	0.0598
Rage Against The ...	0.0348	0.1276	0.0392	0.0299	0.0542	0.0303	0.0402	0.0829	0.1849	0.0736	0.0722	0.0765	0.0175	0.0441	0.0914
Rage Against The ...	0.0285	0.0911	0.0290	0.0177	0.0565	0.0196	0.0440	0.0825	0.1913	0.0367	0.0636	0.1025	0.0153	0.0768	0.1441
Rage Against The ...	0.0338	0.0720	0.0402	0.0236	0.0615	0.0198	0.0373	0.1005	0.1615	0.0439	0.0555	0.1085	0.0221	0.0796	0.1394
Rage Against The ...	0.0290	0.0705	0.0315	0.0168	0.0349	0.0228	0.0428	0.0896	0.2034	0.0502	0.0893	0.1304	0.0151	0.0765	0.0964
Alley Life	0.0370	0.0995	0.0375	0.0233	0.0441	0.0242	0.0363	0.1153	0.1544	0.0408	0.0789	0.0890	0.0202	0.0900	0.1087
Rage Against The ...	0.0432	0.0958	0.0429	0.0282	0.0493	0.0337	0.0352	0.1191	0.1357	0.0372	0.0684	0.0864	0.0137	0.0781	0.1323
Caw! Caw!	0.0540	0.0907	0.0510	0.0311	0.0653	0.0369	0.0640	0.1358	0.0990	0.0277	0.0512	0.0787	0.0162	0.0601	0.1375
Fu Manchu	0.0695	0.1009	0.0567	0.0294	0.0873	0.0423	0.0668	0.1225	0.0893	0.0275	0.0461	0.0710	0.0167	0.0603	0.1128
Rage Against The ...	0.0797	0.1097	0.0557	0.0284	0.0995	0.0457	0.0580	0.1047	0.0911	0.0290	0.0456	0.0670	0.0191	0.0601	0.1060
Caw! Caw!	0.0830	0.1168	0.0526	0.0273	0.1039	0.0467	0.0501	0.0888	0.0936	0.0296	0.0477	0.0664	0.0210	0.0607	0.1111
Alley Life	0.0819	0.1211	0.0489	0.0264	0.1028	0.0464	0.0460	0.0813	0.0950	0.0286	0.0507	0.0664	0.0223	0.0619	0.1195
Alley Life	0.0785	0.1229	0.0455	0.0259	0.0995	0.0461	0.0451	0.0789	0.0960	0.0270	0.0549	0.0666	0.0230	0.0621	0.1273
Dub Pistols	0.0735	0.1236	0.0427	0.0261	0.0960	0.0460	0.0458	0.0796	0.0984	0.0258	0.0589	0.0667	0.0233	0.0613	0.1315
Dub Pistols	0.0676	0.1236	0.0405	0.0269	0.0934	0.0458	0.0460	0.0829	0.1034	0.0256	0.0615	0.0669	0.0234	0.0603	0.1314
$Wingin' UtterS	0.0622	0.1236	0.0394	0.0277	0.0920	0.0459	0.0455	0.0866	0.1074	0.0260	0.0632	0.0666	0.0239	0.0599	0.1294

Figure 1: Another example of the Last.fm intra-session attention mechanism prioritizing long sessions.

	Type O ... Stabbin... Him	Theatre... Weezer Afi Opeth Type O ... Amorphis Marilyn... Soundga... Katatonia Cave In Catheri...	Nine In... Brave Marilyn... A Perfe... Garbage Nirvana Sneaker... Amorphis Dashboa... Tiamat	Veruca ... Type O ... Nirvana Garbage Amorphis Tiamat The Goo... 30 Seco...	Fiona A... Filter Evanesc... Drain Sth Chapter... Dashboa...	Chapter... Nine In...	Porcupi... Nine In... Nirvana The Sma... Saves T... Marilyn... Nirvana The Sma... Audioslave A Perfe... Nirvana The Goo... The Sma... Radiohead A Perfe... Tool A Perfe...	Filter Blackfield Veruca ...	Porcupi... Fuel Garbage One Of Us Nine In... Weezer Bush Zwan Covenant 30 Seco... Garbage Amorphis Tool Machine... 30 Seco...	Evanesc... White W... God Liv... Radiohead God Liv... Nirvana God Liv... Evanesc... Vnv Nation Bush The Sma... Thursday Aghora Bush Radiohead Vintersorg God Liv... Savage ... Garbage	Katatonia Tool Blackfield The Pos... Smash M... Amorphis Tool God Liv...	Fiona A... Dashboa... Amorphis The Sma...	Tool Him Nine In... The Goo... A Perfe... Tool Marilyn...	Brave The Goo... Marilyn... Duran D... Caw! Caw! Garbage Gravity... Nirvana Amorphis Soundga... Katatonia Arise F... Ghost O... Evanesc... Porcupi... A Perfe... Matchbo... Brave Type O ...	Soundga... Soft Cell Weezer Afi God Liv... Stone T... Amorphis
Porcupine Tree	0.1251	0.1995	0.1343	0.1221	0.0778	0.0255	0.1066	0.0933	0.0640	0.0331	0.0089	0.0030	0.0036	0.0014	0.0010
The Goo Goo Dolls	0.0669	0.1675	0.0968	0.0881	0.0699	0.0450	0.1250	0.1099	0.1169	0.0727	0.0158	0.0055	0.0061	0.0051	0.0083
Zwan	0.0716	0.1495	0.0817	0.0804	0.0696	0.0365	0.1322	0.1239	0.1338	0.0804	0.0177	0.0045	0.0048	0.0043	0.0082
God Lives Underwater	0.1038	0.1416	0.0919	0.0816	0.0763	0.0437	0.1039	0.0974	0.1232	0.0844	0.0232	0.0066	0.0064	0.0056	0.0098
The Goo Goo Dolls	0.1089	0.1424	0.0846	0.0749	0.0724	0.0418	0.1079	0.0981	0.1323	0.0844	0.0238	0.0065	0.0065	0.0052	0.0096
Amorphis	0.1014	0.1384	0.0881	0.0793	0.0782	0.0490	0.1079	0.0991	0.1242	0.0774	0.0266	0.0065	0.0068	0.0056	0.0107
Tool	0.1002	0.1441	0.0877	0.0786	0.0797	0.0449	0.1078	0.1042	0.1226	0.0740	0.0258	0.0060	0.0065	0.0058	0.0114
Amorphis	0.1017	0.1456	0.0863	0.0788	0.0834	0.0479	0.1072	0.1051	0.1186	0.0706	0.0245	0.0063	0.0062	0.0054	0.0117
Nirvana	0.0974	0.1407	0.0847	0.0780	0.0831	0.0492	0.1129	0.1071	0.1226	0.0714	0.0245	0.0058	0.0058	0.0050	0.0111
Bush	0.0902	0.1279	0.0763	0.0780	0.0841	0.0483	0.1269	0.1166	0.1298	0.0714	0.0250	0.0050	0.0052	0.0043	0.0100
Bush	0.0872	0.1259	0.0766	0.0784	0.0878	0.0493	0.1274	0.1179	0.1293	0.0702	0.0237	0.0049	0.0051	0.0045	0.0109
The Goo Goo Dolls	0.0924	0.1235	0.0739	0.0740	0.0869	0.0498	0.1307	0.1130	0.1359	0.0702	0.0231	0.0049	0.0047	0.0045	0.0117
Porcupine Tree	0.1006	0.1230	0.0734	0.0714	0.0807	0.0488	0.1291	0.1115	0.1416	0.0700	0.0227	0.0049	0.0046	0.0045	0.0123
Tool	0.1099	0.1228	0.0746	0.0695	0.0720	0.0477	0.1253	0.1155	0.1441	0.0687	0.0224	0.0047	0.0048	0.0046	0.0126
The Goo Goo Dolls	0.1180	0.1237	0.0760	0.0665	0.0631	0.0463	0.1208	0.1248	0.1434	0.0677	0.0219	0.0047	0.0049	0.0047	0.0129
The Smashing Pump...	0.1236	0.1263	0.0764	0.0624	0.0575	0.0437	0.1154	0.1366	0.1406	0.0678	0.0212	0.0047	0.0052	0.0049	0.0131
Porcupine Tree	0.1264	0.1301	0.0755	0.0594	0.0553	0.0411	0.1100	0.1477	0.1359	0.0690	0.0203	0.0048	0.0054	0.0051	0.0133
God Lives Underwater	0.1258	0.1337	0.0743	0.0585	0.0551	0.0393	0.1066	0.1562	0.1304	0.0706	0.0193	0.0050	0.0055	0.0054	0.0135
Afi	0.1223	0.1362	0.0740	0.0591	0.0560	0.0384	0.1062	0.1610	0.1259	0.0717	0.0184	0.0051	0.0056	0.0055	0.0137

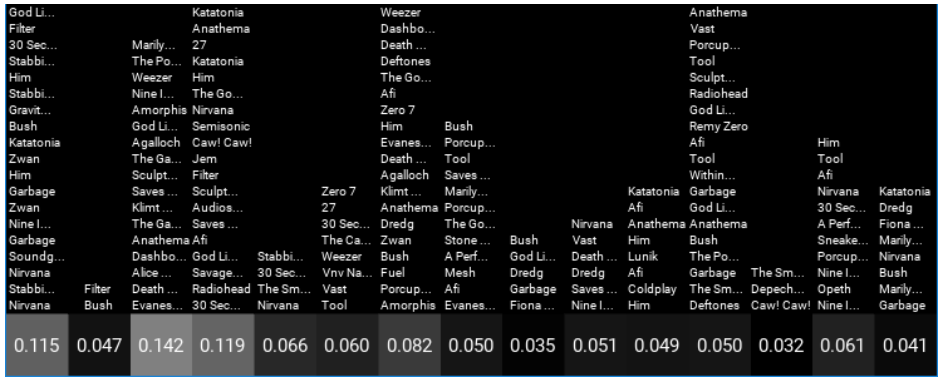
Figure 2: Another example of the Last.fm intra-session attention mechanism showing that more recent sessions are prioritized as well as the de-prioritization of sessions of length 2.

2 Intra-Session Attention Weights for Reddit

	WTF pics science AskReddit funny AskReddit atheism	pics AskReddit	funny WTF aww	AskReddit aww	aww todayi...	funny WTF videos AskReddit atheism politics funny pics AskReddit atheism	atheism aww pics	worldnews Advice... videos pics WTF aww funny	AskReddit atheism AskReddit atheism	aww Music aww AskReddit	Permac... todayi...	todayi... AskReddit todayi...	AskReddit todayi...	AskReddit atheism	atheism pics
atheism	0.1523	0.1605	0.0905	0.0703	0.0596	0.0569	0.0535	0.0514	0.0537	0.0454	0.0458	0.0446	0.0419	0.0385	0.0344
AskReddit	0.2000	0.1864	0.1005	0.0823	0.0640	0.0540	0.0492	0.0431	0.0405	0.0326	0.0329	0.0311	0.0283	0.0289	0.0253
AskReddit	0.2612	0.1961	0.0823	0.0637	0.0578	0.0447	0.0417	0.0354	0.0381	0.0276	0.0292	0.0311	0.0296	0.0330	0.0278
AskReddit	0.1722	0.0911	0.0842	0.0781	0.0738	0.0563	0.0556	0.0509	0.0516	0.0508	0.0500	0.0494	0.0460	0.0457	0.0436
pics	0.2473	0.1809	0.1263	0.0949	0.0709	0.0499	0.0366	0.0294	0.0294	0.0227	0.0229	0.0230	0.0217	0.0233	0.0201
AskReddit	0.1828	0.1049	0.1126	0.1058	0.0749	0.0667	0.0584	0.0455	0.0449	0.0371	0.0373	0.0360	0.0321	0.0329	0.0275
funny	0.1722	0.1973	0.0940	0.0904	0.0696	0.0557	0.0528	0.0353	0.0390	0.0286	0.0293	0.0323	0.0324	0.0377	0.0326
AskReddit	0.1603	0.1200	0.1121	0.1071	0.0842	0.0716	0.0613	0.0445	0.0406	0.0401	0.0366	0.0341	0.0298	0.0318	0.0252
AskReddit	0.2144	0.2175	0.1125	0.0897	0.0647	0.0457	0.0442	0.0281	0.0311	0.0222	0.0233	0.0257	0.0253	0.0299	0.0249
atheism	0.1316	0.2001	0.1554	0.1346	0.0849	0.0496	0.0417	0.0266	0.0287	0.0235	0.0233	0.0252	0.0240	0.0269	0.0232
AskReddit	0.2441	0.1318	0.1070	0.0874	0.0722	0.0593	0.0543	0.0368	0.0359	0.0325	0.0310	0.0304	0.0268	0.0282	0.0217
AskReddit	0.2583	0.1690	0.0826	0.0843	0.0686	0.0521	0.0448	0.0329	0.0375	0.0250	0.0256	0.0292	0.0289	0.0335	0.0269
AskReddit	0.2280	0.1194	0.0914	0.0884	0.0724	0.0518	0.0467	0.0417	0.0444	0.0358	0.0351	0.0379	0.0356	0.0389	0.0317
AskReddit	0.1954	0.0971	0.0783	0.0818	0.0723	0.0546	0.0545	0.0507	0.0530	0.0418	0.0415	0.0450	0.0443	0.0473	0.0418
AskReddit	0.1963	0.1129	0.0917	0.0837	0.0700	0.0561	0.0525	0.0483	0.0504	0.0381	0.0385	0.0410	0.0399	0.0426	0.0371
AskReddit	0.1950	0.0835	0.0848	0.0826	0.0687	0.0589	0.0560	0.0521	0.0542	0.0426	0.0431	0.0453	0.0449	0.0464	0.0413
AskReddit	0.2002	0.0912	0.0849	0.0816	0.0679	0.0597	0.0551	0.0505	0.0523	0.0418	0.0421	0.0439	0.0433	0.0448	0.0398
AskReddit	0.2051	0.0771	0.0754	0.0763	0.0643	0.0602	0.0561	0.0528	0.0546	0.0455	0.0456	0.0474	0.0473	0.0480	0.0437
AskReddit	0.2079	0.0794	0.0735	0.0760	0.0627	0.0601	0.0554	0.0525	0.0536	0.0460	0.0459	0.0474	0.0473	0.0477	0.0439

Figure 3: An example of the Reddit intra-session attention mechanism. This one shows more recent sessions being prioritized. It does not de-prioritize short sessions.

3 Inter-Session Hidden Attention Weights for Last.fm



(a)

(b)

Figure 4: Attention weights for the hidden inter-session attention mechanism using the Last.fm dataset. These show similar patterns as the intra-session Last.fm, with short sessions being de-prioritized. The sixth session from the right in Figure (b) seems to be heavily de-prioritized even though it is very long, but this is an exception to the rule.

4 Inter-Session Delta-t Attention Weights for Last.fm

Δt	attn
4	0.901
7	0.064
9	0.029
11	0.005
13	0.001
23	0.000
28	0.000
36	0.000
37	0.000
51	0.000
53	0.000
55	0.000
57	0.000
61	0.000
73	0.000

(a)

Δt	attn
8	0.336
9	0.380
10	0.241
12	0.029
14	0.006
15	0.003
16	0.003
17	0.001
19	0.000
21	0.000
23	0.000
31	0.000
32	0.000
35	0.000
36	0.000

(b)

Δt	attn
26	0.477
28	0.225
29	0.101
31	0.072
35	0.117
47	0.003
51	0.002
59	0.002
60	0.000
71	0.000
74	0.000
96	0.000
98	0.000
100	0.000
109	0.000

(c)

Figure 5: Attention weights for the delta-t inter-session attention mechanism using the Last.fm dataset. This shows similar patterns to the same attention weights for the Reddit dataset, with more recent sessions being heavily prioritized. When there are no recent sessions, the attention weights spread more evenly.

5 Session Representation Attention Weights for Reddit

(a) (b) (c) (d)

Figure 6: Attention weights for the hierarchical attention when creating session representations. Shown here are several examples of the attention weights for sessions containing the subreddit AskReddit. In figures (a) and (b) AskReddit has a very small weight and larger weights are given to smaller subreddits. For reference, AskReddit has as of 07.06.18 roughly 19,346,000 subscribers while Tinder has 967,000 subscribers and keto has 647,000 subscribers. It does not always have a low weight however. In Figure (c) you can see that the weight is close to that of pics, which has roughly the same number of subscribers as AskReddit. Sometimes, AskReddit will also get a higher weight than smaller subreddits such as in Figure (d) (nfl has 778,000 subscribers), but this is uncommon.