İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS

PREDICTING PURCHASING INTENT AND NEXT ITEM RECOMMENDATION FOR SESSION BASED RECOMMENDATION SYSTEMS

Graduation Project Final Report

Ahmed Burak Gulhan
150160903

Department: Computer Engineering
Division: Computer Engineering

Advisor : Assist. Prof. Dr. Yusuf YASLAN

February 2021

Statement of Authenticity

I/we hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by my/our individual authenticity.

İstanbul, 31/01/2021

Ahmed Burak Gulhan

SESSION BASED RECOMMENDATION SYSTEMS

(SUMMARY)

Session based recommendation systems are extensively used in a large amount of websites. These websites are mainly e-commerce sites, such as Amazon, AliBaba, GittiGidiyor, etc. However session based recommendation is not limited only to e-commerce and is also used in sites such as YouTube or Spotify.

The goal of session based recommendation is to make inferences for a user based on their behavior. Since we assume that users are completely anonymous, all inferences must be made from user’s current session. For example, a recommendation system for a user browsing and e-commerce website will use the items the user has viewed in the session to make some predictions. There are two goals in session based recommendation systems. The first goal is to predict purchasing intent, that is whether or not a user has an intent to purchase an item. By making accurate predictions on purchasing intent, an e-commerce website can give incentives to specific customers in order to make them purchase, thereby improving profits. The second and more important goal is to make item recommendations to customers based on their session history. By showing items that a user has interest in, an e-commerce website can recommend items that the user doesn’t know about but would have an interest in which again would increase profits. Furthermore, next item recommendation is not only used in e-commerce, it is used in sites such as YouTube to recommend videos, Spotify to recommend music and even in online advertisements to show relevant ads.

I have used the YooChoose 2015 dataset to train my models. This dataset contains over 9 million anonymous sessions from an e-commerce website. This dataset was used in the RecSys 2015 challenge and has since been used as a baseline dataset to measure model performance in recommendation systems. This dataset contains information on items that have been clicked on, how long an item was viewed and which items were purchased. However all items are anonymized and use a random number in place of the item name.

In order to predict purchasing intent, I have used a convolutional neural network together with embeddings. The convolutional neural network model I have used was originally made for sentence classification and since sessions used in the dataset are similar to sentences, this model gives good results. For embeddings I have used Google’s Word2Vec embeddings, which turns the items into 100 dimensional vectors. The goal of using embeddings is to extract meaning from the items and use these to improve the performance of the model. This means that items that are purchased together will have vectors that are close to each other.

For predicting next item recommendation I have used a recurrent neural network (RNN), more specifically the GRU4REC model. Recurrent neural networks are a type of neural network that are well suited for processing sequential data, which is the type of data used in session based recommendation systems. There is a vast amount of different models and variations that have been developed for session based recommendation systems. Many of these are based on the GRU4REC model, which became very popular at the time of its publication and has since been used as a baseline model for comparisons. In this project I have also used the GRU4REC model. In addition to this model I have used an attention mechanism and included time data called dwell time (which is rarely used in session based recommendation models) with the goal of increasing baseline performance. An attention mechanism is used to give weight to a sequence of data to improve performance. Attention mechanisms can be used in images, sentences and temporal data.

The results obtained for predicting purchasing intent show promise. My proposed model did exceed the state of the art model’s performance. Additionally the state of the art model uses hand crafted features for the Yoochoose dataset, which makes it perform worse on other datasets, however my proposed model does not need any handcrafted features and should show similar a performance for other datasets. The results of the model proposed for next item recommendation is better than baseline GRU4REC, but more research is needed. The attention mechanism did not show any improvements for a GRU network by itself, but has shown improvements when using it in conjunction with dwell time.

SESSION BASED RECOMMENDATION SYSTEMS

(ÖZET)

Oturuma dayalı öneri sistemleri, çok sayıda web sitesinde yaygın olarak kullanılmaktadır. Bu web siteleri ağırlıklı olarak Amazon, AliBaba, GittiGidiyor gibi e-ticaret siteleridir. Ancak oturum bazlı öneri sadece e-ticaretle sınırlı olmayıp, YouTube veya Spotify gibi sitelerde de kullanılmaktadır.

Oturuma dayalı tavsiyenin amacı, bir kullanıcı için onun davranışlarına göre çıkarımlar yapmaktır. Kullanıcıların tamamen anonim olduklarını varsaydığımız için, tüm çıkarımlar kullanıcının mevcut oturumundan yapılmalıdır. Örneğin, bir e-ticaret web sitesi bir öneri sistemi, tahminlerde bulunmak için sadece kullanıcının o oturumda görüntülediği öğeleri kullanacaktır. Oturum bazlı öneri sistemlerinde iki amaç vardır. İlk amaç, satın alma niyetini, yani bir kullanıcının bir ürünü satın alma niyetinin olup olmadığını tahmin etmektir. Bir e-ticaret web sitesi, satın alma niyetiyle ilgili doğru tahminlerde bulunarak, belirli müşterileri satın almaları için teşvik edebilir ve böylece karlarını arttırabilir. İkinci ve daha önemli amaç, müşterilerin oturum geçmişlerine göre ürün önerileri yapmaktır. Bir e-ticaret web sitesi, kullanıcının ilgilenebileceği elemanları göstererek, kullanıcının bilmediği ve ilgi duyacağı elemanlar önerebilir, böylece yine kar arttırılabilir. Üstelik bir sonraki ürün önerisi sadece e-ticarette değil, YouTube gibi sitelerde video önermek için, Spotify müzik önermek için ve hatta online reklamlarda alakalı reklamlar göstermek için kullanılmaktadır.

Modellerimi eğitmek için YooChoose 2015 datasetini kullandım. Bu dataset, bir e-ticaret web sitesinden gelen 9 milyondan fazla anonim oturumu içeriyor. Bu ancak yine karı artıracak, RecSys 2015 yarışmasında kullanıldı ve o zamandan beri başka öneri sistemlerindeki model performansını ölçmek için standart datset olarak kullanıldı. Bu dataseti, tıklanan elemanlar, bir elemanın ne kadar süreyle görüntülendiği ve hangi elemanların satın alındığı hakkında bilgi içeriyor. Bununla birlikte, tüm öğeler anonimleştirilmiştir ve isim yerine rastgele bir sayı kullanılmıştır.

Satın alma niyetini tahmin etmek için, düğünlerle birlikte evrişimli bir sinir ağı kullandım. Kullandığım evrişimsel sinir ağı (CNN) modeli, ilk olarak cümle sınıflandırması için yapılmıştı ama datsette kullanılan oturumlar cümlelere benzediği için bu model iyi sonuçlar veriyor. Embedding için, elemanları 100 boyutlu vektörlere dönüştüren Google'ın Word2Vec embeddingi kullandım. Embedding kullanımının amacı, elemanlardan anlam çıkarmak ve bunları modelin performansını artırmak için kullanmaktır. Bu, birlikte satın alınan elemanların birbirine yakın vektörlere sahip olacağı anlamına geliyor.

Bir sonraki öğe tavsiyesini tahmin etmek için yinelenen bir sinir ağı (RNN) kullanan GRU4Rec modelini kullandım. Yinelenen sinir ağları, oturum tabanlı öneri sistemlerinde kullanılan veri türü olan sıralı verileri işlemek için çok uygun olan bir sinir ağı türüdür. Oturum bazlı öneri sistemleri için geliştirilmiş çok sayıda farklı modeller vardır. Bunların büyük kısmı, yayınlandığı sırada çok popüler hale gelen ve o zamandan beri karşılaştırmalar için temel model olarak kullanılan GRU4REC modeline dayanmaktadır. Bu projede GRU4REC modelini de kullandım. Bu modele ek olarak, temel performansı artırmak amacıyla bir dikkat mekanizması kullandım ve zaman verilerini de dahil ettim. Performansı artırmak için bir dizi veriye ağırlık vermek için bir dikkat mekanizması kullanılır. Dikkat mekanizmaları resimlerde, cümlelerde ve zamansal verilerde kullanılabilir.

Satın alma niyetini tahmin etmek için elde edilen sonuçla iyi gözüküyor. Önerdiğim model, şu ana kadar en iyi modelin performansını aştı. Ek olarak, en iyi performans veren model Yoochoose veri kümesi için el yapımı özellikleri kullanır, bu da diğer veri kümelerinde daha kötü performans göstermesine sebep olur, ancak benim önerdiğim modelim herhangi bir el yapımı özelliğe ihtiyaç duymuyor ve diğer veri kümeleri için benzer bir performans göstererebilir. Bir sonraki öğe önerisi için önerilen modelin sonuçları temel GRU4REC performansından daha iyi, ancak daha fazla araştırmaya ihtiyaç vardır. Dikkat mekanizması kendi başına bir GRU ağı için herhangi bir gelişme göstermedi, ancak zaman verisi ile birlikte kullanıldığında gelişmeler gösterdi.

Contents

1 Introduction and Project Summary........1

2 Comparative Literature Survey........3
2.1 Recommendation Systems........3
2.2 Predicting Purchasing Intent........3
2.3 Next Item Recommendation........4

3 Developed Approach and System Model........8
3.1 Predicting Purchasing Intent Structural Model........8
3.2 Next Item Recommendation Structural Model........11
3.2.1 Recurrent Neural Network........11
3.2.2 GRU4REC Model........12
3.2.3 Including Dwell Time........13
3.2.4 Attention Mechanism........13

4 Experimentation Environment and Experiment Design........15
4.1 Predicting Purchasing Intent Experimentation........15
4.2 Next Item Recommendation Experimentation........17

5 Comparative Evaluation and Discussion........20
5.1 Predicting Purchasing Intent Model Evaluation........20
5.2 Next Item Recommendation Model Evaluation........20

6 Conclusion and Future Work........22
6.1 Predicting Purchasing Intent........22
6.2 Next Item Recommendation........22

7 References........24

1 Introduction and Project Summary

Session-based recommendation systems are an emerging field in recommendation systems. Session based recommendation systems are extensively used on commercial websites, especially e-commerce websites, such as Amazon, AliBaba, GittiGidiyor, etc. These systems allow the user a better experience and serve to increase profits on such websites. Session-based recommendation is a method of predicting purchasing intent and the next item to recommend by using an anonymous sequence of customer behavior. These types recommendation systems are used primarily in e-commerce platforms. By using a session based recommendation system an e-commerce platform can predict and recommend items that will interest the customer and can predict the customer's buying intent and provide incentives for the customer to purchase. This can improve the profits of the e-commerce platform.

In this project I have focused on the two parts of a session-based recommendation system: predicting the next item a customer will view and predicting the purchasing intent of the customer. For predicting the purchasing intent of the customer I have used a convolutional neural network (CNN) along with item embeddings. By using embeddings we can map items into a low dimensional space in order to calculate their similarity. This means that similar items and items that are frequently purchased together will have similar vectors. By using embeddings we can improve the performance of a recommendation system [7]. For item embeddings I have used a skip-gram network, specifically Google's word2vec tool. Each item in the e-commerce dataset will be treated as a word, and by training with sessions in the dataset, which are similar to sentences for embedding purposes, a 100 dimensional vector is generated for each item. Since our problem is similar to a sentence classification problem and that CNN's have very good performance at classifying sentences, I used a CNN model together with word2vec embeddings [9]. The goal for predicting purchasing intent was to surpass the current state of the art performance which has an area under curve (AUC) score of 0.853 [7]. I have achieved a score of 0.855 using 10 fold cross validation with my model. The results in this project were obtained by using the YooChoose 2015 dataset and all computations were done in Google Colab.

For predicting the next item a customer will view, I have used a recurrent neural network (RNN). I have conducted tests with a GRU-based RNN. A RNN is useful for training with sequential data, such as the data we will be using. A problem that RNN models faces is that it suffers from short-term memory. In long sequences there are problems carrying information from earlier steps to later steps. A solution for this problem is using LSTM or GRU. LSTM and GRU are used to regulate the flow of information and can learn which data in a sequence is important to keep. Both LSTM and GRU accomplish the same thing, however GRU is faster to train then LSTM as it contains fewer gates and GRU has a similar performance as LSTM. One of the best performing models for next item recommendation uses, called GRU4REC, uses GRU. In the GRU4REC model the results are a Hit Rate for the top 20 items (HR@20 or Recall@20) score of 0.7211 and a Mean Reciprocal Rank for the top 20 items (MRR@20) score of 0.3170 [3]. These results are obtained by using the YooChoose 2015 dataset [5]. My proposed model uses the

GRU4REC model with a self attention mechanism in addition to incorporating dwell-time (how long a customer has viewed an item), which is rarely used in other models.

The result of my predicting purchasing intent model achieved a AUC score of 0.855 for the YooChoose 2015 dataset. This score is higher than the state of art score of 0.853, also it should be noted that the state of the art solution uses extensive hand crafted features made specifically for the YooChoose 2015 dataset and may not have as good of a performance in other datasets as it has in the YooChoose dataset [7]. However, since the code for the state of the art solution is not public, this cannot be tested. In contrast, my solution does not involve any hand crafted features, there are only three hyperparameters, which are the three different sampling sizes used in the CNN and these hyperparameters can be modified for any specific dataset. This also makes this model simplr to use in other datasets. Additionally, due to computational time constraints in Google Colab, all of the training dataset could not be used for generating word2vec embeddings in each k-fold during cross validation, instead instead 1/8 of the YooChoose 2015 dataset was isolated and used for training embeddings and the rest of the database was used for training the model. Training embeddings in each k-fold iteration may give better performance for this model.

The model that achieved the best performance for next item recommendation had a Recall@20 score of 0.7706 and a MRR@20 score of 0.6379. The model used for this score was a modified GRU4REC model with an added attention mechanism and incorporated dwell time. This score is surpasses the GRU4REC model performance, but note that only 1/64 of the YooChoose 2015 dataset was used for this result, tests using 1/32 and 1/16 of the YooChoose 2015 datasets showed worse performance with this model.

2 Comparative Literature Survey

2.1. Recommendation Systems

In 2015 ACM Conference on Recommender Systems made a session-based recommendation challenge called the RecSys Challenge 2015 [5]. In this challenge a database from an online retailer YOOCHOOSE was provided, which contained over 9.2 million user sessions. The goal of this challenge was to determine whether a user will purchase anything in the current session and what they will purchase. The winner of this challenge used an ensemble learner with over 400 hand crafted features. The results of the winning model was an AUC score of 0.85 for predicting purchasing intent and an Jaccard measure of 0.765 for next item recommendation[6]. This was about %50 of the maximum possible score which was considered an impressive result.

A problem with e-commerce data for session-based recommendation is that there are comparatively very little amount of buyers. For the YooChoose 2015 dataset about %5.5 of all 9.2 million sessions were buyers. Another problem is that the dataset is sparse, some items are visited much less frequently compared to others and a significant number of sessions are very short, with a session length less then or equal to three.

2.2. Predicting Purchasing Intent

Another part of the problem in session-based recommendation systems is to predict whether a user has the intention of buying or not. This is called predicting purchasing intent. The performance of predicting purchasing intent is usually measured using Area Under Curve (AUC). The winner of the RecSys 2015 challenge had and AUC score of 0.853 for the validation set using a GBM model, which is still the state of the art solution [6]. However the code for this solution is not publicly available.

In 2018 Sheil et al. proposed a LSTM model to predict purchasing intent [7]. The YooChoose 2015 and Retail Rocket Kaggle datasets were used with this model. The authors were able to obtain the state of the art model, which was the winner of the 2015 RecSys challenge to compare results with their own model. The model Sheil et al. developed uses some new ways of data preparation that increases performance. One being event unrolling, which uses the time the user has viewed an item in training. The other being sequence reversal, that is reversing the input sequences before giving it to the model. This seems very simple, but since the most important item in predicting purchasing intent is the last item, therefore if we reverse the input sequence we give the most important item first. Sequence reversal improves the test AUC on the YooChoose 2015 dataset by 0.005 - from 0.834 to 0.839. However this improvement is negligible and may have been caused by randomness during training. Additionally the LSTM model proposed in this paper also made use of dwell time by using event unrolling. The model developed in this paper had a result of 0.839 AUC for YooChoose 2015 dataset which is 98% of the state of the art solution, which is 0.853 AUC. However for Retail Rocket Kaggle dataset, the result of the model is 0.838 AUC which is better then the state of the art model’s solution of 0.834.

The model used in this project is based on the simple CNN model proposed by Zhang and Wallace in 2016 [9]. This model was proposed as a method for sentence classification using a convolutional neural network with tokenized sentences. The goal of Zhang and Wallace was to identify which settings in a convolutional neural network should be tuned for the best performance and the ranges of the hyperparameters to be changed [9]. Zhang and Wallace concluded that filter region sizes have a large effect on model performance and should be tuned, furthermore they concluded that 1-max pooling regularly outperformed other pooling strategies. The architecture of this model is composed of three filter regions of different filter sizes with each filter region containing one hundred feature maps. Each region's feature maps are concatenated using 1-max pooling into a single feature vector. This feature vector is sent into a dense layer to obtain a binary output. The amount of feature maps in each filter region have some effect on performance. Zhang and Wallace found that increasing feature maps increases performance up to a certain point [9]. One hundred feature maps is sufficient for this project's purposes, and adding more feature maps will only increase training time.

2.3. Next Item Recommendation

One part of the problem in session-based recommendation systems is predicting the next item the customer will choose by examining the previous items the customer has visited. The evaluation criteria of this is, Hit Rate (HR or Recall) and Mean Reciprocal Rank (MRR). A major breakthrough in this area was in March 2016 where Hidasi et al. developed a Recurrent Neural Network using Gated Recurrent Units (GRU) on session-based data, specifically the YooChoose 2015 dataset [2]. The basic architecture of this model is shown in Figure 2.1. This model has a HR@20 (or Recall@20) and MRR@20 for performance evaluation. HR@20 is the proportion of cases having the desired item among the top-20 items in all test cases and MRR@20 is the average of reciprocal ranks of the desired items. MRR is set to zero if the rank is above 20. The results were a MRR@20 score of 0.2693, which is a 31.49% improvement over the previous best baseline result and a HR@20 score of 0.6322 which is a 24.82% improvement over the previous best baseline result. This model was called the GRU4REC model.

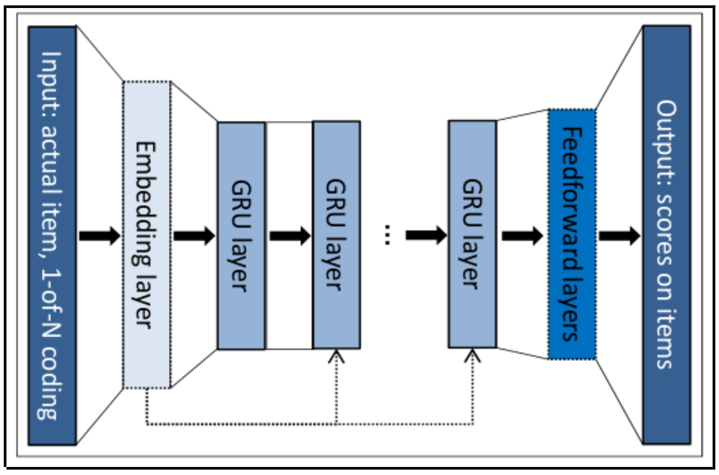


Figure 2.1: General architecture of the GRU4REC network

One of the challenges for session-based recommendation is that each session differs in length. Hidasi et al. proposed a method called session-parallel mini-batches for the RNN model input as is shown in Figure 2.2 [2]. In this method, first the sessions are ordered. Then, first event of the first X sessions are used to form the input of the first mini-batch where the desired output is the second events of these sessions. The second mini-batch is formed from the second events and this continues for each mini batch. If any of the sessions end, the next valid session is used in its place. The appropriate hidden hidden state of the RNN is reset when this change occurs, since each session is independent.

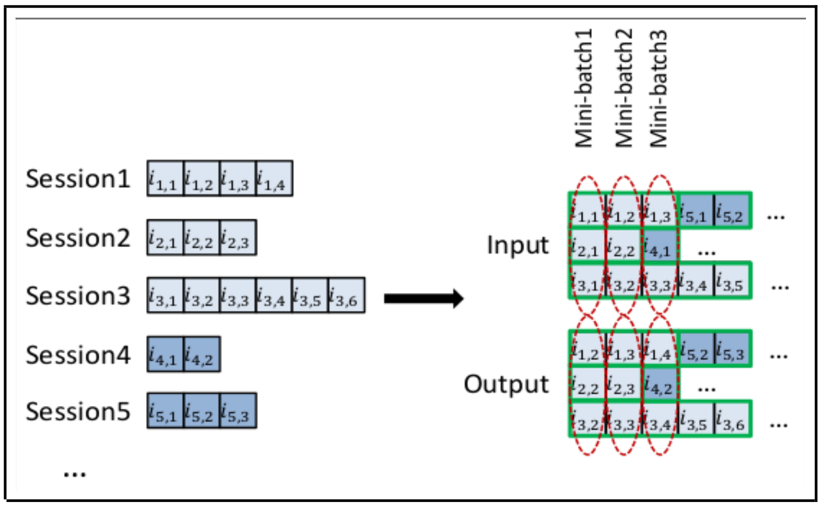


Figure 2.2: Making session-parallel mini-batches

In September 2016 Tan et al. made improvements on the previously mentioned RNN model using data augmentation [8]. For data augmentation two different methods were used. The first method is treating prefixes of the original input sessions as new training sequences. Given an input training session [x_1 , x_2 , ..., x_n], the sequences and corresponding labels ([x_1], V (x_2)), ([x_1, x_2], V (x_3)), ... ,([x_1, x_2, . . . , x_{n-1}], V (x_n)) are generated for training. The second method is using embedding dropout, which is a form of regularization applied to inputs. This is equivalent to randomly deleting items in a sequence, which makes the model less sensitive to noisy clicks. An example of these methods is shown in Figure 2.3. This model achieved a HR@20 result of 0.7129 and a MRR@20 result of 0.3091 which are a 12.8% and 14.8% improvement over previous results on the previous GRU4REC model.

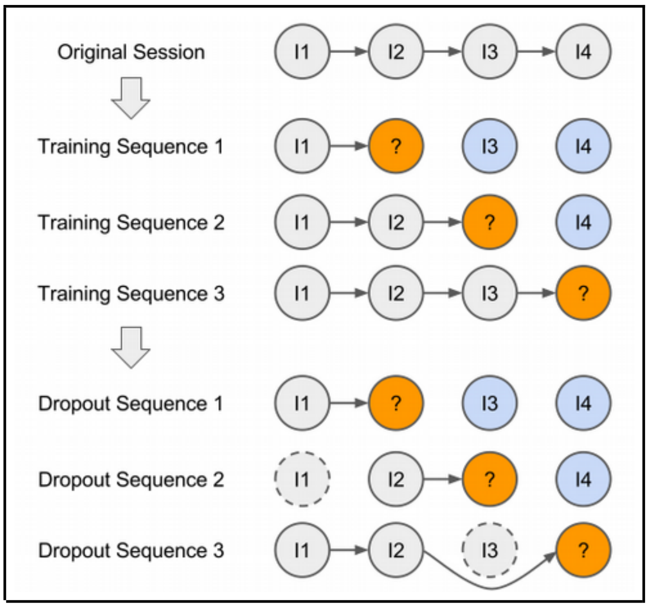


Figure 2.3: Two different data augmentation methods

In 2018 Hidasi and Karatzoglou improved upon their previous GRU4REC model published in 2016 [3]. The changes made were using a improved input sampling method called negative sampling and improving the loss function. In the new sampling method a different sample is used as negative sampling for each sample in the mini batch, as shown in in Figure 2.4. The results were a HR@20 score of 0.7211 and a MRR@20 score of 0.3170 compared to the previous GRU4REC results of MRR@20 score of 0.2693 and HR@20 score of 0.6322. This model was used in this project and modified in an attempt to improve its performance. The code for this model is available publicly.

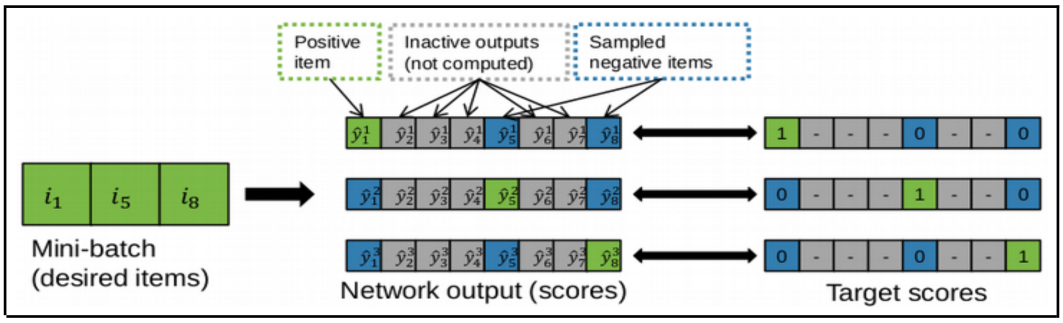


Figure 2.4: Negative sampling used by Hidasi and Karazoglou [3]

In 2017 Bogina and Kuflik published a paper on using dwell-time with the GRU4REC model and the YooChoose 2015 dataset [1]. Bogina and Kuflik determined that dwell time plays a significant role in predictions. They proposed a method, named item boosting, in order to include dwell-time in sessions. In an e-commerce dataset (YooChoose 2015) we have sequences of clicks and each click has a dwell time dt_i. The proposed method, item boosting, boosts items that have a dwell time greater than a predefined threshold t. Each item in a session that passes this threshold is multiplied such that the number of occurrences of these items are $(dt_i/t + 1)$. This method was used in conjunction with the session-parallel mini-batches method used in GRU4REC. This can be seen in Figure 2.4. The improvements made by Hidasi and Karatzoglou for the GRU4REC model (called GRU4REC with sampling) was used by Bogina and Kuflik [3].

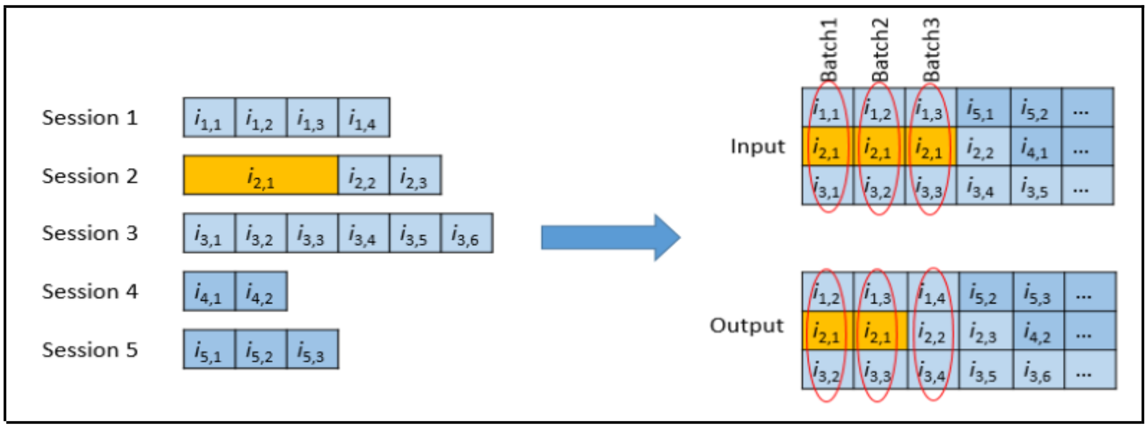


Figure 2.5: Dwell-time based session-parallel mini-batches using the items boosting method

The results obtained by Bogina and Kuflik, shown in Table 2.1, show a very large improvement to both Recall@20 and MRR@20. The optimum dwell time threshold was found to be 75. The MRR@20 score more than doubled with a score of 0.636 compared to 0.308 and the Recall@20 score also had a significant improvement with a score of 0.853 compared to 0.7117. The items boosting method used for this result was used in this project, however the large amount of improvement reported by Bogina and Kuflik were not able to be replicated, but there was still significant improvements in performance.

Table 2.1: Results obtained by Bogina and Kuflik using dwell-time

Method	Recall@20	MRR@20
GRU4REC with sampling	0.7117	0.308
GRU4Rec with sampling and dwell time threshold 75	**0.853**	**0.636**

There are several models different from GRU4REC that employ an attention mechanism. One such model is the Neural Attentive Session-based recommender (NARM) that uses a RNN encoder and decoder with a attention mechanism in between [4]. Another such model is the Variational Session-based Recommender which uses a GRU network with an attention mechanism using normalizing flows [10]. Both of these models are reported to achieve better performance on the YooChoose 2015 dataset compared to GRU4REC [4], [10]. Therefore I decided to incorporate an attention mechanism for the GRU4REC model with hopes of increasing performance for this project. The type of attention mechanism I used is a self attention mechanism.

3 Developed Approach and System Model

The work done for both predicting purchasing intent and next item recommendation consist of building and training a machine learning model in Keras and almost all data structures used were matrices and vectors for storing training and testing data. Therefore a dynamic model and data model are not relevant for this project. I will only present the structural model of the machine learning models by describing the workflow and the algorithms used.

3.1 Predicting Purchasing Intent Structural Model

In the CNN model used for predicting purchasing intent, the word2vec embeddings are trained first. 1/8 of the YooChoose 2015 dataset is isolated and used for training embeddings. Session lengths of three or less are discarded since these sessions do not contain much contextual information. Each session contains consecutive numbers, each representing an item. The goal of using word2vec embeddings is to vectorize these items so that similar items will have vectors close to each other. The result of the embedding training is a 18017x100 matrix. This means that there are 18017 items each with a 100 degree vector. This embedding vector is saved and is used when training the CNN model. By using UMAP, an algorithm for dimension reduction on our word2vec embedding vectors and plotting the result we can see how the vectorized items form clusters. This plot is shown in Figure 3.1 below.

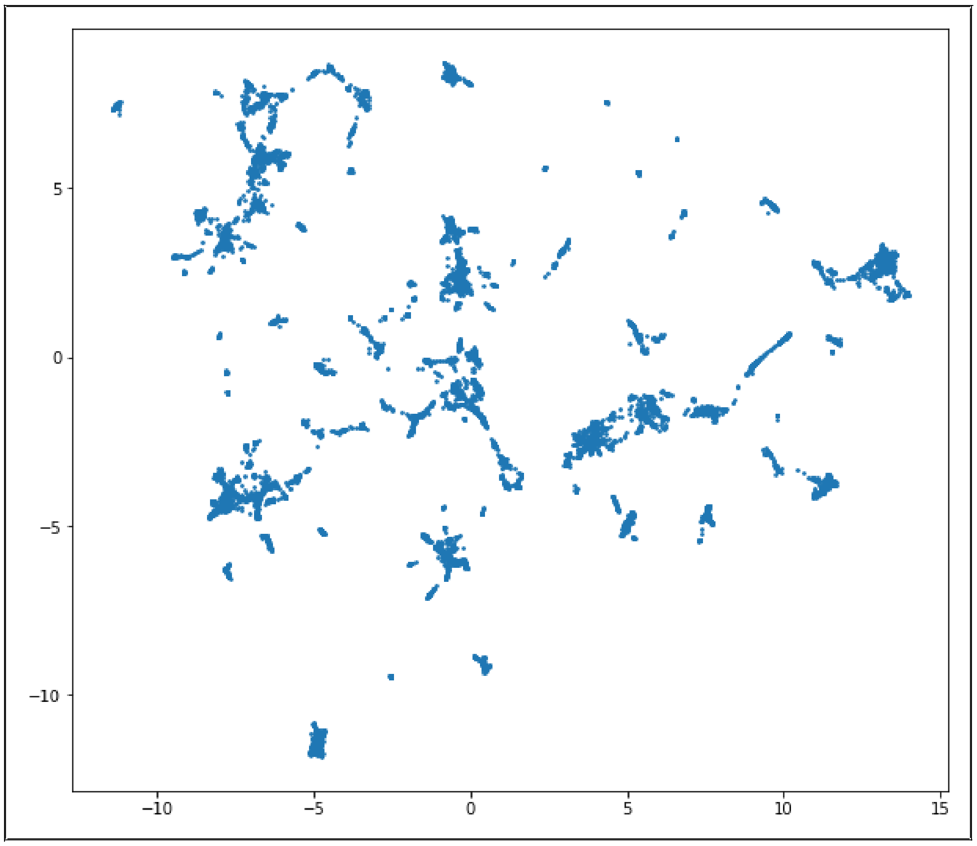
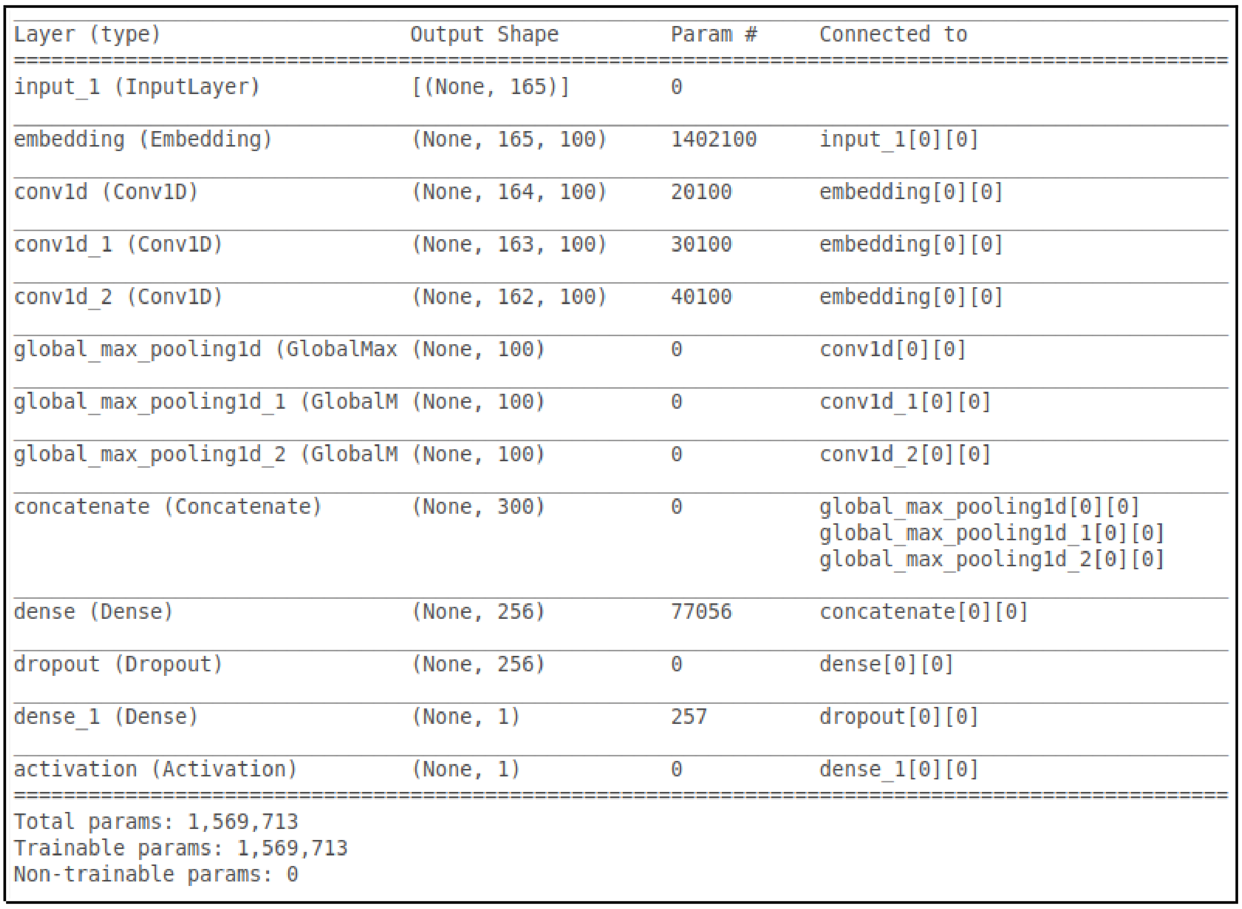


Figure 3.1: UMAP plot of word2vec embeddings

Before training the CNN model, some preprocessing occurs. The sessions are read from the YooChoose 2015 dataset and these sessions are stored in a list called ‘X’ with each item being a list of variable length containing a session. Another list is made with the corresponding indices being 0 or 1 depending on if a session has made a purchase. All session lengths are used. Then all of the sessions in list ‘X’ are padded with 0’s to make each session have a length of 165, since the maximum session length in the YooChoose 2015 dataset is 162.

Then our processed data is sent into a 10-fold cross validation loop where the input data is separated into ten sections. In this loop we calculate weights for each class and modify the class weight before training the model. Here we use our embedding matrix obtained using word2vec, and our data is sent into the embedding layer of the model. The model summary obtained from Keras can be seen in Figure 3.2. From this figure wee can see that each session is turned into a two dimensional matrix, with each item number represented by a 100 degree vector obtained from our word2vec embeddings. Then the modified inputs are sent to three different Conv1d layers, which are the filter regions. Each filter region outputs 100 different feature maps with the regions having filter sizes of 2, 3 and 4. These feature maps are concatenated after 1-max pooling and sent to a dense layer. This dense layer is connected to a sigmoid activation layer where the output is obtained. The model parameters are trained using binary cross-entropy as the loss model, adam as the optimizer, and AUC as the performance metric.



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 165)]        0

embedding (Embedding)           (None, 165, 100)     1402100     input_1[0][0]

conv1d (Conv1D)                 (None, 164, 100)     20100       embedding[0][0]

conv1d_1 (Conv1D)               (None, 163, 100)     30100       embedding[0][0]

conv1d_2 (Conv1D)               (None, 162, 100)     40100       embedding[0][0]

global_max_pooling1d (GlobalMax (None, 100)          0           conv1d[0][0]

global_max_pooling1d_1 (GlobalM (None, 100)          0           conv1d_1[0][0]

global_max_pooling1d_2 (GlobalM (None, 100)          0           conv1d_2[0][0]

concatenate (Concatenate)       (None, 300)          0           global_max_pooling1d[0][0]
                                                                 global_max_pooling1d_1[0][0]
                                                                 global_max_pooling1d_2[0][0]

dense (Dense)                   (None, 256)          77056       concatenate[0][0]

dropout (Dropout)               (None, 256)          0           dense[0][0]

dense_1 (Dense)                 (None, 1)            257         dropout[0][0]

activation (Activation)         (None, 1)            0           dense_1[0][0]
==================================================================================================
Total params: 1,569,713
Trainable params: 1,569,713
Non-trainable params: 0
```

Figure 3.2: Summary of CNN model

An illustration of a similar CNN model is shown in Figure 3.3 which shows the CNN model for sentence classification proposed by Zhang and Wallace [9]. This figure is almost identical to the model used in this project with a few minor differences. The model in figure accepts and input of a sentence with each word being represented as a 5-dimensional vector, whereas in our model we input a session with each item being represented by a 100 dimensional vector. Similar to the model used in this project there are three different filter regions with sizes 2, 3, and 4. In the figure there are only two filters in each region, whereas there is one hundred in this project's model. These filters are concatenated using 1-max pooling and sent to a softmax function in the figure, whereas for the model in this project it is sent to a sigmoid function since we do binary classification.

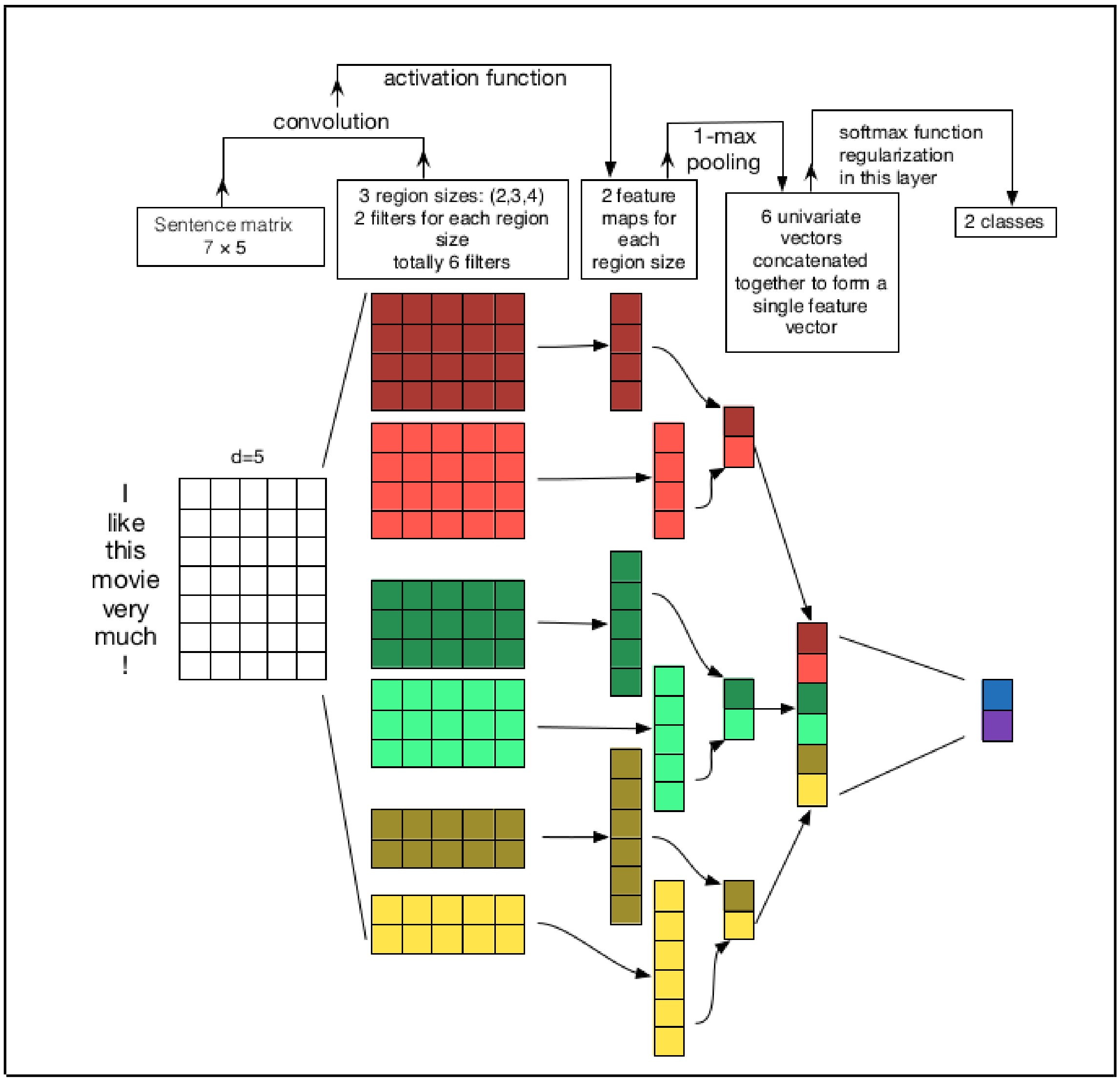

activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 x 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 3.3: CNN model proposed by Zhang and Wallace [9]

3.2 Next Item Recommendation Structural Model

3.2.1 Recurrent Neural Network

The GRU4REC model used in this project is a type of Recurrent Neural Network (RNN). A RNN is a type of neural network and the difference between a feed forward neural network and a RNN is that a RNN can handle sequential data, memorize previous inputs and consider inputs separately. This makes a RNN very suitable for session based data. An image of a RNN is shown in Figure 3.4. A RNN has a hidden layer whose output, represented by h in Figure 3.4, is sent to the next item in the sequence. This behavior allows the RNN to understand the relationship between temporal data

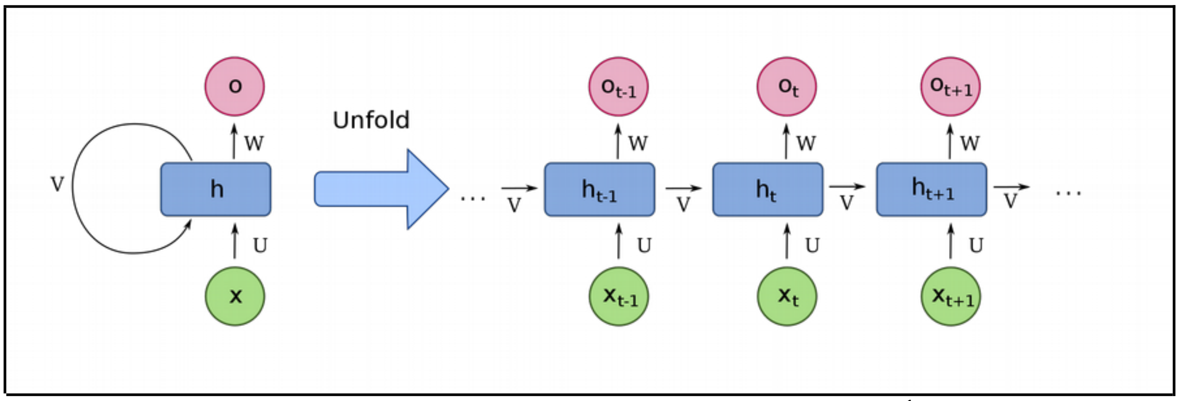


Figure 3.4: Unfolded basic Recurrent Neural Network[1]

There are some problems in a basic RNN. One of these problems is called the vanishing gradient problem, where there is a loss of information throughout time. The other problems is called the exploding gradient problem, where errors in the gradient accumulate and the network gets unstable from very large updates. To fix these problems, LSTM or GRU are used. LSTM and GRU are a type of gating mechanism used in a RNN. What an LSTM or GRU does is that it decides how much past data to remember and what information to be sent to the output. Basically they regulate the flow of information. LSTM and GRU have similar performances, but a GRU is simpler than a LSTM which allows faster training. GRUs are used in GRU4REC. An image of a GRU unit is shown in Figure 3.5

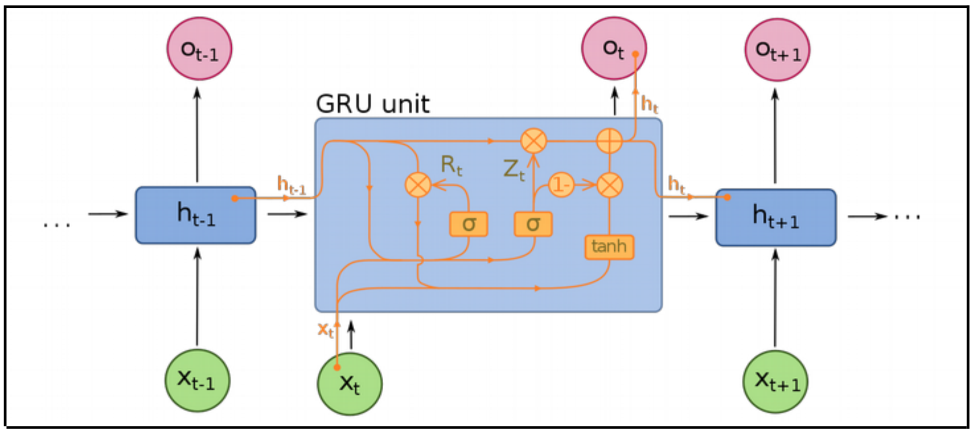


Figure 3.5: Gated Recurrent Unit

[1]Image from https://en.wikipedia.org/wiki/Recurrent_neural_network

3.2.2 GRU4REC Model

The GRU4REC model uses a GRU-based RNN network. The input of this model is an item from a session and the output the next item in the session. All items are encoded using one hot encoding. These encoded items are sent to the GRU layers as shown in Figure 3.6. Note that the embedding layer gives slightly worse results in this network so it is not used [2]. The GRU layers connect to a feed forward layer which connects to the output layer.

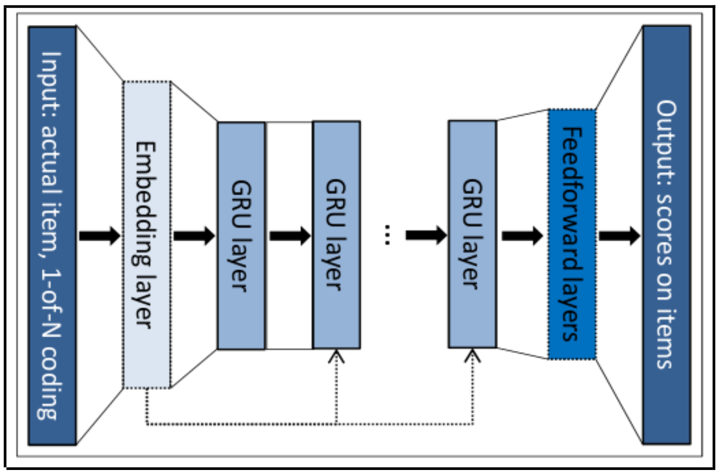
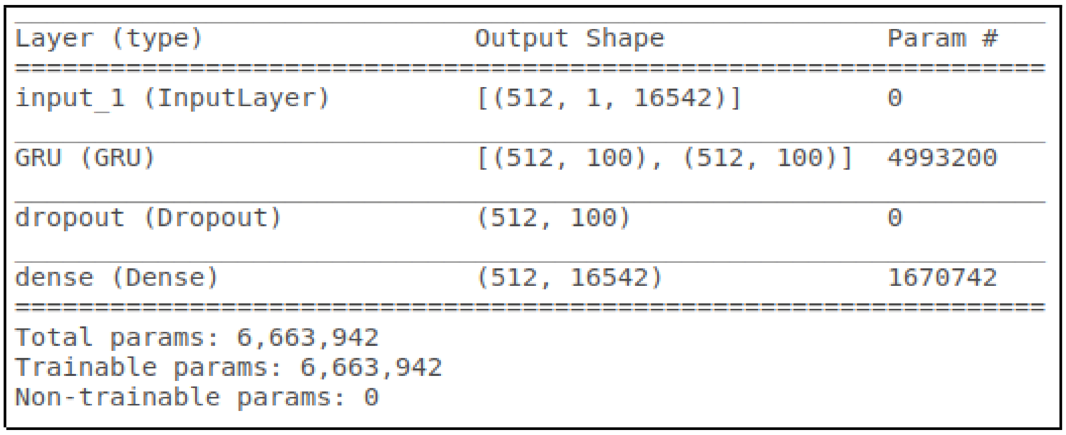


Figure 3.6: GRU4REC General Architecture

A problem with the GRU network is that sessions cannot be fed into the model one after the other since the session lengths are different. Hidasi et al. proposed a method called session-parallel mini-batches for the first GRU4REC model which is shown in Figure 2.2 [2]. In this method first the sessions are ordered. Then, first event of the first X sessions are used to form the input of the first mini-batch where the desired output is the second events of these sessions. The second mini-batch is formed from the second events and this continues for each mini batch. In the GRU4REC model the mini batch size is 512. If any of the sessions end, the next valid session is used in its place. The appropriate hidden state of the GRU layers is reset when this change occurs, since each session is independent. However, this exact method is not used in the current GRU4REC model, since improvement to this sampling method have been made. This new sampling method is called negative sampling, which is shown in in Figure 2.4. In this method, for each example in the mini batch, another example is used for negative sampling.

In this project I did not use the original GRU4REC code which is available publicly, since our testing environment did not support the Theano library as a back end, which was used in the original code. Instead a Keras implementation of this model was used[1]. The summary of the model used in the Keras implementation is shown in Figure 3.7. However, this summary does not show the one hot embeddings and the batch sampling. These operations are done in the data preprocessing stage, and the modified inputs are then sent to the model.

[1]https://github.com/paxcema/KerasGRU4Rec



```
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(512, 1, 16542)]         0

GRU (GRU)                   [(512, 100), (512, 100)]  4993200

dropout (Dropout)           (512, 100)                0

dense (Dense)               (512, 16542)              1670742
=================================================================
Total params: 6,663,942
Trainable params: 6,663,942
Non-trainable params: 0
```

Figure 3.7: GRU4REC Keras model summary

3.2.3 Including Dwell Time

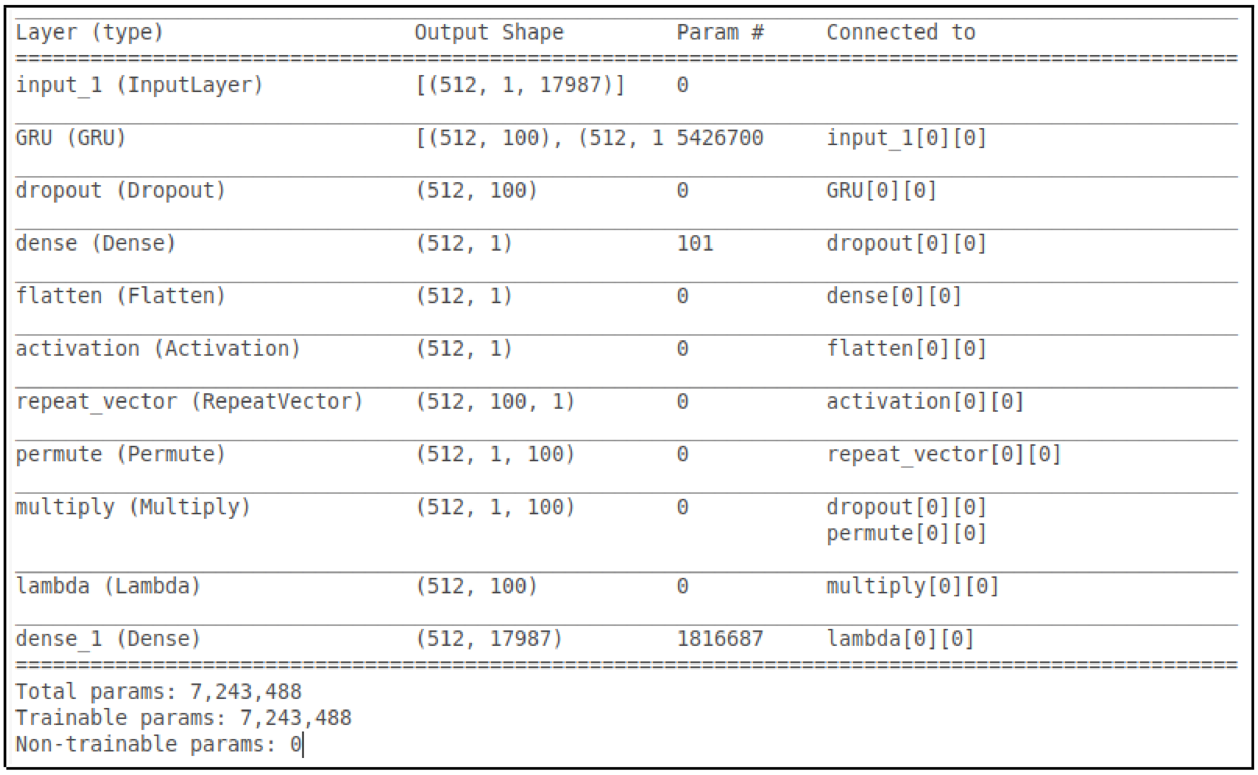
One method of improving GRU4REC performance is by using dwell time. Dwell time is how much time has passed before the next item is clicked on. The method proposed by Bogina and Tsvi was used to incorporate dwell time in the GRU4REC model, which is called items boosting [1]. The idea behind this method is that the more an item is viewed, the more the user is interested in that item. For each session there is a sequence of clicks $\{x_1, \dots, x_n\}$. For each click there exists a dwell time dt_i. For items viewed longer than a dwell time threshold t they are boosted, such that we duplicate that item by a certain amount. The amount of times an item is duplicated is determined by the formula (dt_i/t + 1). The dwell time threshold used in this model is 75 seconds since this is the optimum threshold time determined by Bogina and Kuflik [1]. An example of items boosting is shown in Figure 2.5.

The implementation of items boosting was done in the preprocessing stage in this project. After augmenting the dataset, no further modification is needed to be made.

3.2.4 Attention Mechanism

Another method of improving GRU4REC performance is by implementing an attention mechanism. The goal of an attention mechanism is to assign importance to a sequence, so that some items in a sequence get more attention and other items get ignored. There are several session based recommendation models that utilize an attention mechanism in their design [4], [10]. The attention mechanism I used in the GRU4REC model is called self attention. Basically, this type of attention allows items in a sequence to attend to previous items.

The GRU4REC with attention model summary can be seen in Figure 3.7. From this summary we can see the attention mechanism that was added between the GRU layer and the dense layer. The GRU layer is connected to a dropout layer, which then has it’s output sent to two different layers. One output is sent to the multiply layer and the other output is sent to a dense layer. This dense layer is sent to a flatten layer and a activation layer with a softmax activation. The dense layer calculate the weights of the sequences and the softmax activation layer turns these values into probabilities. These weights are sent to a repeat vector layer and a permute layer where some arrangements are made to the vector so that it can be properly multiplied. This weight vector is sent to a multiply layer where the output from the drop layer (which the GRU layer is connected to) and the weight vector are multiplied element wise. This output is the output of the GRU layers with attention applied. This output of the multiply layer is sent to a lambda layer where the shape is modified again and then this result is sent to a dense layer where the model gives it's output.



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(512, 1, 17987)]    0

GRU (GRU)                       [(512, 100), (512, 1 5426700     input_1[0][0]

dropout (Dropout)               (512, 100)           0           GRU[0][0]

dense (Dense)                   (512, 1)             101         dropout[0][0]

flatten (Flatten)               (512, 1)             0           dense[0][0]

activation (Activation)         (512, 1)             0           flatten[0][0]

repeat_vector (RepeatVector)    (512, 100, 1)        0           activation[0][0]

permute (Permute)               (512, 1, 100)        0           repeat_vector[0][0]

multiply (Multiply)             (512, 1, 100)        0           dropout[0][0]
                                                                 permute[0][0]

lambda (Lambda)                 (512, 100)           0           multiply[0][0]

dense_1 (Dense)                 (512, 17987)         1816687     lambda[0][0]
==================================================================================================
Total params: 7,243,488
Trainable params: 7,243,488
Non-trainable params: 0
```

Figure 3.7: Summary of GRU4REC with attention mechanism

4 Experimentation Environment and Experiment Design

All testing and experimentation of this project was done on Google Colaboratory. Google Colaboratory provides a Nvidia Tesla P100 PCIe 16GB GPU and 12.72 GB of RAM. Colab uses a Jupyter notebook with Python 3.6.9. Both predicting purchasing intent and next item recommendation were implemented using Keras which is an interface for Tensorflow (version 2.0 in Google Colab).

There were some limitations in Google Colab which caused problems during this project. Google Colab has a computational time limitation of at most 12 hours per session. This prevented me from using all of YooChoose dataset for some time intensive computations. Additionally, Google Colab times out if it detects that the user is idle, this also caused problems during long computations so that I had to be on the computer for the entire computation duration. Google Colab also temporary disables GPU usage if you make long lasting computations using the GPU, this also caused problems.

Lastly Google Colab had problems when running the NARM model, which was going to be used in predicting next item recommendation as a baseline comparison and with running the original GRU4REC code. The publicly available NARM code was used. Attempting to run this code on Google Colab initially works, but when training the model the results for Recall and MRR were shown as ‘nan’ after the first epoch preventing me from using this model. This problem does not occur when running the code on my personal computer, but this is still infeasible due to my hardware limitations. Running the GRU4REC code works, however this code is written using theano backed and Google Colab does not offer GPU support for theano. Therefore, the computation time for the original GRU4REC code was too large and therefore this code could not be used. However a KERAS implementation of GRU4REC was used instead.

4.1. Predicting Purchasing Intent Experimentation

For the CNN model used in this project, some experimentation was done in order to tune the hyperparameters of the regions. The model contains three regions with one hundred filters each, so there are only three hyperparameters needed to tune. Each hyperparameter determines the size of the filters in a region. Zhang and Wallace found that regions with identical hyperparameters perform worse than regions with consecutive hyperparameters [9]. Therefore I did not use any identical hyperparameters for this experimentation. Additionally the results in the work of Zhang and Wallace show that high values for regions perform worse than lower values [9]. As a result I limited my experimentation to lower values. The following table shows the AUC results and the standard deviation obtained by different region values using 10-fold cross validation.

Table 4.1: Hyperparameter tuning for region sizes

Region Sizes	AUC score
1,2,3	0.8541 (+- 0.0042)
2,3,4	**0.8554 (+- 0.0037)**
3,4,5	0.8538 (+- 0.0034)
4,5,6	0.8512 (+- 0.0034)
5,6,7	0.8503 (+- 0.0036)
6,7,8	0.8502 (+- 0.0036)

From this table we observe that the optimum region sizes are 2,3,4 and that as regions get larger the AUC score steadily decreases. It should be noted that the AUC scores with sub-optimal region sizes do not have a significant decrease. In fact the non optimal region size 3,4,5 has the same AUC score as the state of the art model of 0.853 [7].

Another experimentation done on the CNN model was using class weights. The YooChoose 2015 dataset has a large class imbalance, only around 7% of all sessions have made a purchase. Therefore I used class weights when training the model to mitigate this problem. Table 4.2 shows the AUC score and standard deviation with and without weights for this model, calculated using 10-fold cross validation. From this table we can observe a small but notable difference in AUC.

Table 4.2: Effect of class weights on AUC score

No class weighting	Class weighting
0.8502 (+-0.0036)	**0.8554 (+- 0.0037)**

Furthermore, I also measured the difference word2vec embeddings make in AUC score. Instead of using word2vec embeddings I used one hot vectors and compared the results. However due to the very large size of the one hot vector matrix and as a result of the large increase in the parameters needed to be trained, the testing environment runs out of RAM and crashes, therefore I used 1/64 of the dataset to compare. From Table 4.3 we can see that using word2vec embeddings has a significant improvement to AUC score compared to using one-hot encoding. Additionally the training time of word2vec is around 10 times faster compared to one-hot encoding.

Table 4.3: Effect of embeddings on AUC score (1/64 dataset)

One-hot encoding	Word2Vec
0.7223 (+- 0.0433)	**0.7513 (+- 0.0265)**

4.2. Next Item Recommendation Experimentation

For next item recommendation the entire YooChoose 2015 dataset could not be used. Instead 1/64, 1/32, and 1/8 of the dataset were used. For measuring performance the metrics MRR@30, MRR@20, MRR@10, MRR@5, Recall@30, Recall@20, Recall@10, and Recall@5 were used.

The first experimentation done was incorporating dwell time as shown by Bogina and Kuflik [1]. The method used in incorporating dwell time was implemented in the preprocessing stage of the model. The parameter, dwell time threshold is used for this method. I used the threshold value of 75 seconds as this value shows the best increase in performance [1]. The results of dwell time are shown in Table 4.4.

Table 4.4: GRU4REC with dwell time results

	Recall@30	Recall@20	Recall@10	Recall@5	MRR@30	MRR@20	MRR@10	MRR@5
GRU4REC (1/64)	0.7151	0.6676	0.5634	0.4385	0.2871	0.2853	0.2780	0.2612
GRU4REC with dwell time (1/64)	**0.7368**	**0.7168**	**0.6825**	**0.6460**	**0.5850**	**0.5847**	**0.5823**	**0.5771**
GRU4REC (1/32)	0.6772	0.6412	0.5678	0.4745	0.3325	0.3311	0.3259	0.3128
GRU4REC with dwell time (1/32)	**0.6933**	**0.6792**	**0.6543**	**0.6245**	**0.5708**	**0.5703**	**0.5686**	**0.5645**
GRU4REC (1/16)	**0.7086**	**0.6725**	0.5943	0.4856	0.3361	0.3346	0.3291	0.3141
GRU4REC with dwell time (1/16)	0.6661	0.6546	**0.6307**	**0.5985**	**0.5462**	**0.5457**	**0.5440**	**0.5396**

From these results we observe that MRR gets a large increase when using dwell time. We also can observe that the difference between MRR@30 and MRR@5 is less when using dwell time compared to without. This means that GRU4REC with dwell time makes correct predictions with a lower rank more often compared to normal GRU4REC. An interesting observation is that the smaller datasets get higher MRR results using GRU4REC with dwell time. This may be occurring since the smaller datasets have slightly less amounts of different items compared to larger datasets and so the model can make more accurate predictions. More analysis is needed to determine the reason of this behavior.

For the Recall results we observe that GRU4REC with dwell time consistently gets a significant increase for Recall@10 and Recall@5. However Recall@30 and Recall@20 get a much smaller increase in performance for the 1/64 and 1/32 datasets. In the 1/16 dataset we see that GRU4REC performs better than GRU4REC with dwell time for Recall@30 and Recall@20. More analysis is also needed to determine the reason for such results. Additionally we can observe that the difference between Recall@30 and Recall@5 is less when using dwell time compared to without. Finally, we observe is that the smaller datasets get higher Recall results using GRU4REC with dwell time, similar to the case with MRR.

Another experiment conducted to improve GRU4REC performance was adding an attention mechanism to the model. I have added a self attention mechanism between the GRU layer and the dense layer. The results of GRU4REC with attention are written in Table 4.5 below.

Table 4.5: GRU4REC with attention results

	Recall@30	Recall@20	Recall@10	Recall@5	MRR@30	MRR@20	MRR@10	MRR@5
GRU4REC (1/64)	0.7151	0.6676	0.5634	0.4385	**0.2871**	**0.2853**	**0.2780**	**0.2612**
GRU4REC with attention (1/64)	**0.7292**	**0.6798**	**0.5751**	**0.4437**	0.2856	0.2836	0.2762	0.2583
GRU4REC (1/32)	0.6772	0.6412	0.5678	0.4745	0.3325	0.3311	0.3259	0.3128
GRU4REC with attention (1/32)	**0.6881**	**0.6525**	**0.5783**	**0.4831**	**0.3421**	**0.3406**	**0.3354**	**0.3224**
GRU4REC (1/16)	**0.7086**	**0.6725**	**0.5943**	**0.4856**	**0.3361**	**0.3346**	**0.3291**	**0.3141**
GRU4REC with attention (1/16)	0.7027	0.6665	0.5847	0.4761	0.3349	0.3335	0.3277	0.3129

From these results we see that GRU4REC with attention provides a very minor benefit in Recall at 1/64 and 1/32 datasets, but does not for the 1/16 dataset. For MRR values, there doesn't seem to be a correlation between dataset sizes and model performance. The different between these values of the two models are less than 0.01. These differences could be caused by the randomness is the GRU4REC model and therefore the attention mechanism used in this model does not give any significant performance increase.

For the final experimentation for next item recommendation was using both dwell time and the attention mechanism together. The results are shown in Table 4.6.

Table 4.6: GRU4REC with attention and dwell time results

	Recall@30	Recall@20	Recall@10	Recall@5	MRR@30	MRR@20	MRR@10	MRR@5
GRU4REC (1/64)	0.7151	0.6676	0.5634	0.4385	0.2871	0.2853	0.2780	0.2612
GRU4REC with dwell time (1/64)	0.7368	0.7168	0.6825	0.6460	0.5850	0.5847	0.5823	0.5771
GRU4REC with attention and dwell time (1/64)	**0.7910**	**0.7706**	**0.7329**	**0.6933**	**0.6383**	**0.6379**	**0.6352**	**0.6299**
GRU4REC (1/32)	0.6772	0.6412	0.5678	0.4745	0.3325	0.3311	0.3259	0.3128
GRU4REC with dwell time (1/32)	0.6933	0.6792	0.6543	0.6245	0.5708	0.5703	0.5686	0.5645
GRU4REC with attention and dwell time(1/32)	**0.7332**	**0.7197**	**0.6938**	**0.6613**	**0.6104**	**0.6099**	**0.6080**	**0.6036**
GRU4REC (1/16)	**0.7086**	**0.6725**	0.5943	0.4856	0.3361	0.3346	0.3291	0.3141
GRU4REC with dwell time (1/16)	0.6661	0.6546	0.6307	0.5985	0.5462	0.5457	0.5440	0.5396
GRU4REC with attention and dwell time(1/16)	0.6761	0.6645	**0.6382**	**0.6029**	**0.5577**	**0.5572**	**0.5553**	**0.5505**

From these results we see that GRU4REC with attention and dwell time performs better than the previously tested models for both Recall and MRR in all datasets. This is a surprising result since GRU4REC with attention did not show any significant increase in performance by itself, however with both attention and dwell time the model performs better than the sum of its parts. Similar to the GRU4REC with dwell time model, this model also performs much better at a smaller dataset compared to a larger one. This model shows an even greater difference between the 1/64 dataset scores and 1/16 dataset scores then the GRU4REC with dwell time model does.

5 Comparative Evaluation and Discussion

5.1. Predicting Purchasing Intent Model Evaluation

My goal for predicting purchasing intent, stated in my interim report, was to surpass the state of the art AUC result of 0.853 for the YooChoose 2015 dataset which was made by Romov and Sokolov [5], [6]. My model was able to surpass this result with a score of 0.855 which was obtained using 10-fold cross validation, meaning that this result wasn't obtained by chance. Additionally the previous state of the art model makes extensive use of feature engineering. Shiel et al found that the state of the art model made by Romov and Sokolov does not perform well on datasets with a large percentage of short session lengths [7]. However since a CNN model can 'learn' features by itself, my model should not have problems with other datasets.

5.2. Next Item Recommendation Model Evaluation

My goal for next item recommendation was, stated in my interim report, was to surpass the current best GRU4REC performance of 0.7211 HR@20 and 0.3170 MRR@20 for the YooChoose 2015 dataset by combining different methods from literature. However making comparisons with this original metric would not be completely accurate, since the entire dataset and the original GRU4REC[1] code could not be used due to testing environment restrictions. Instead 1/64, 1/32, and 1/8 of the dataset was used because of computational time restrictions and instead of using the original GRU4REC code implemented in Theano, a Keras implementation[2] of this model was used.

The two methods used to improve the GRU4REC model was using dwell time and adding an attention mechanism. The final results of these methods are shown in Table 5.1. The first method used to improve performance, using dwell time, was the method presented in Bogina and Kufliks paper [1]. Using this method in this project gave a very high increase in performance of the model, but not as much as reported in this paper. However, this may be caused by not using the entire database. The second method used to improve GRU4REC performance was using an attention mechanism. This method did not provide good results, as both models' performance metrics had less than a 0.01 difference. However using both dwell time and the attention mechanism gave much better results, this model gave the best results in almost all cases. The only cases where GRU4REC with attention and dwell time did not give the highest performance was Recall@30 and Recall@20 for the 1/16 YooChoose dataset. The GRU4REC with attention and dwell time has the highest scores for the 1/64 dataset and suffers a loss in performance as the dataset size increases.

The final model achieves a Recall@20 score of 0.7910 and MRR@20 score of 0.6379 using 1/64 of the dataset which surpasses the stated goal of HR@20 score of 0.7211 and MRR@20 score of 0.3170 in the interim report. The results of Bogina and Kufliks original method of including dwell time were a Recall@20 score of 0.853 and MRR@20 score of 0.636 [1]. GRU4REC with attention and dwell time surpasses this MRR@20 score, with a score of 0.6379 but does not surpass this Recall@20 score only getting 0.7910.

[1]https://github.com/hidasib/GRU4Rec
[2]https://github.com/paxcema/KerasGRU4Rec

Table 5.1: Results of next item recommendation

	Recall@30	Recall@20	Recall@10	Recall@5	MRR@30	MRR@20	MRR@10	MRR@5
GRU4REC (1/64)	0.7151	0.6676	0.5634	0.4385	0.2871	0.2853	0.2780	0.2612
GRU4REC with dwell time (1/64)	0.7368	0.7168	0.6825	0.6460	0.5850	0.5847	0.5823	0.5771
GRU4REC with attention (1/64)	0.7292	0.6798	0.5751	0.4437	0.2856	0.2836	0.2762	0.2583
GRU4REC with attention and dwell time (1/64)	**0.7910**	**0.7706**	**0.7329**	**0.6933**	**0.6383**	**0.6379**	**0.6352**	**0.6299**
GRU4REC (1/32)	0.6772	0.6412	0.5678	0.4745	0.3325	0.3311	0.3259	0.3128
GRU4REC with dwell time (1/32)	0.6933	0.6792	0.6543	0.6245	0.5708	0.5703	0.5686	0.5645
GRU4REC with attention (1/32)	0.6881	0.6525	0.5783	0.4831	0.3421	0.3406	0.3354	0.3224
GRU4REC with attention and dwell time(1/32)	**0.7332**	**0.7197**	**0.6938**	**0.6613**	**0.6104**	**0.6099**	**0.6080**	**0.6036**
GRU4REC (1/16)	**0.7086**	**0.6725**	0.5943	0.4856	0.3361	0.3346	0.3291	0.3141
GRU4REC with dwell time (1/16)	0.6661	0.6546	0.6307	0.5985	0.5462	0.5457	0.5440	0.5396
GRU4REC with attention (1/16)	0.7027	0.6665	0.5847	0.4761	0.3349	0.3335	0.3277	0.3129
GRU4REC with attention and dwell time(1/16)	0.6761	0.6645	**0.6382**	**0.6029**	**0.5577**	**0.5572**	**0.5553**	**0.5505**

6 Conclusion and Future Work

6.1. Predicting Purchasing Intent

The model used in this project for predicting purchasing intent surpasses the state of the art model's performance for the YooChoose 2015 dataset with an AUC score of 0.855. The previous state of the art score had an AUC score of 0.853 which was achieved by Romov and Sokolov [6], [7]. Although, it would be prudent to compare these models on other datasets. Sheil et al. managed to obtain the code of the state of the art solution and compared the performance of this model with their proposed LSTM model for predicting purchasing intent [7]. This LSTM model scored less that the state of the art model for the YooChoose 2015 dataset with AUC scores of 0.839 and 0.853 respectively, but these two models were also tested for the Retail Rocket dataset where the LSTM model scored higher than the state of the art model, with AUC scores of 0.838 and 0.834 respectively. This shows that the state of the art model does not generalize well to other datasets. Since the performance of Romov and Sokolov's model is available for the Retail Rocket dataset, the model proposed in this paper could also be tested with that dataset to compare its performance. Furthermore, the LSTM model proposed by Sheil et al also achieves very good results, but it might be further improved by including word2vec embeddings as was used in the CNN model proposed in this project.

Additionally, the CNN model used in this project does not incorporate dwell time. Finding a way to incorporate dwell time in this model may further improve performance. Lastly, due to computational limits, the word2vec embeddings could not be trained for every k-fold iteration, instead 1/8th of the dataset was used for training embeddings beforehand and the rest of the dataset was used for training the model. If better computational resources could be obtained so that both the embeddings and the model could be trained in every k-fold, then the model may show better performance.

Lastly, the word2vec embeddings used in predicting purchasing intent may give better performance with additional tuning. The results obtained in this project for predicting purchasing intent used the default settings for word2vec, which made embeddings in a one hundred dimensional space. Modifying settings, such as vector dimensions, may effect the performance of the model.

6.2. Next Item Recommendation

The model used for next item representation surpasses the state of the art solution for the standard GRU4REC model. However the entire YooChoose 2015 dataset could not be used. The results of the GRU4REC with attention and dwell time model implemented in this project using 1/64 of the YooChoose 2015 dataset is 0.7706 Recall@20 and 0.6379 MRR@20, compared to the Keras GRU4REC model's performance of 0.6725 Recall@20 and 0.3346 MRR@20. However using 1/16 of the dataset for GRU4REC with attention and dwell time shows significantly worse performance compared to using only 1/64 of the dataset with 0.6645 Recall@20 and 0.5572 MRR@20. This behavior is also seen in the GRU4REC with dwell time model. Furthermore, the results obtained Bogina and Kuflik using the same dwell time method achieved a higher Recall@20 performance. Therefore more research is needed to be done on this dwell time method to see what causes these problems.

Furthermore, the attention mechanism used in this project did give not give any significant performance increase without using dwell time. However attention mechanisms are used in models that perform better than GRU4REC and therefore attention mechanisms should give more of a benefit for the GRU4REC model. The low performance of this project’s attention mechanism may have resulted from an improper type of attention mechanism used. A different type of attention mechanism may give better results. Maybe the attention mechanism was improperly implemented in the GRU4REC model. Further research is also needed on attention mechanisms.

7 References

[1] V. Bogina and T. Kuflik, “Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks,” *SIG Proceedings Paper in Word Format. In Proceedings of RecTemp Workshop co-located with ACM RecSys’2017, Como, Italy*, Aug. 2017.

[2] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based Recommendations with Recurrent Neural Networks,” Mar. 2016.

[3] B. Hidasi and A. Karatzoglou, “Recurrent Neural Networks with Top-k Gains for Session-based Recommendations,” Proceedings of the 27th ACM International Conference on Information and Knowledge Management – CIKM 18, 2018.

[4] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural Attentive Session-based Recommendation,” *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017.

[5] P. Romov and E. Sokolov, “RecSys Challenge 2015,” Proceedings of the 2015 International ACM Recommender Systems Challenge on - RecSys 15 Challenge, 2015.

[6] P. Romov and E. Sokolov, “RecSys Challenge 2015: ensemble learning with categorical features,” Proceedings of the 2015 International ACM Recommender Systems Challenge on - RecSys 15 Challenge, pp. 1–4, Sep. 2015.

[7] H. Sheil, O. Rana, and R. Reilly, “Predicting purchasing intent: Automatic Feature Learning using Recurrent Neural Networks,” Jul. 2018.

[8] Y. K. Tan, X. Xu, and Y. Liu, “Improved Recurrent Neural Networks for Session-based Recommendations,” DLRS 2016: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Sep. 2016.

[9] Y. Zhang and B. C. Wallace, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification,” Apr. 2016.

[10] F. Zhou, Z. Wen, K. Zhang, G. Trajcevski, and T. Zhong, “Variational Session-based Recommendation Using Normalizing Flows,” *The World Wide Web Conference on - WWW 19*, pp. 3476–3482, May 2019.