



12/2/2018

MIE324 FINAL REPORT

Team CheckThisOut!

Zhi Cheng 1003343900

Yudong Xu 1003337935



Word Count: 1989 Penalty: 0

Goal and Motivation

Technology has been developing at an increasingly fast rate for the past few decades, with the current era being called the information age, there is a huge amount of content being generated constantly and it is becoming harder and harder for people to filter through. Therefore, a recommender system that can conveniently retrieve information for people is becoming more necessary than it has ever been.

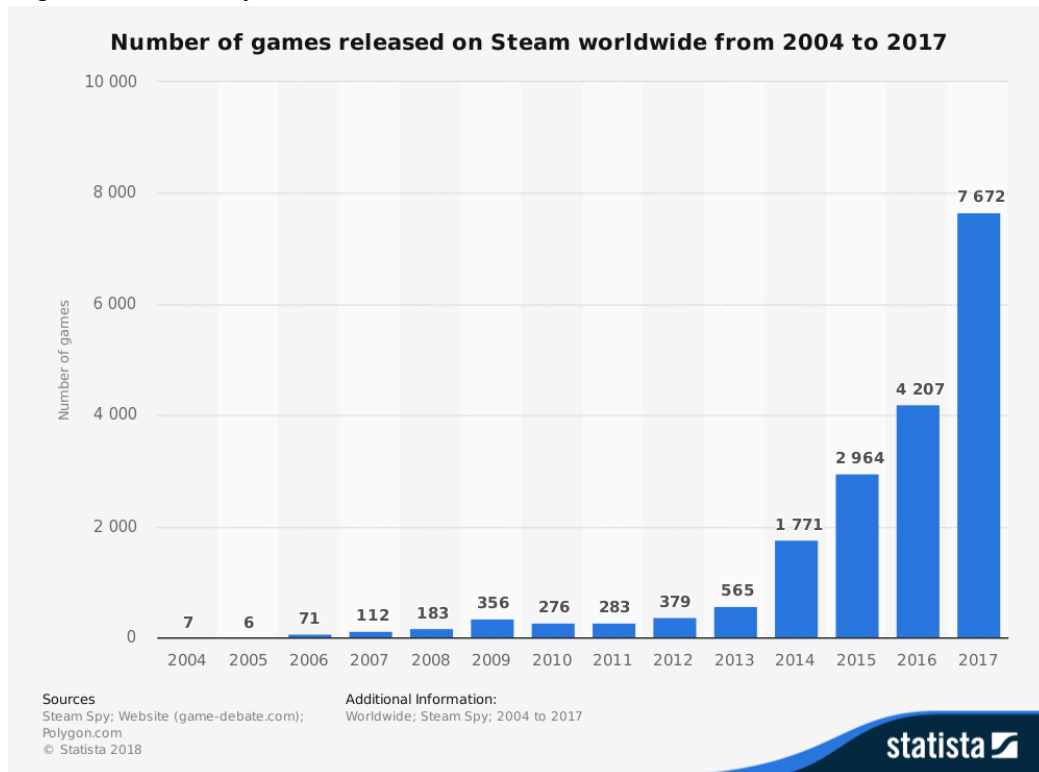


Figure 1 Number of game released on steam from 2004 to 2017

This project aims to tackle a specific type of content that is Video Games. As seen in figure 1, there has been reported a massive increase in the number of video games released every year since 2004 with almost 8000 new games released in 2017 alone[1][2]! A software that can recommend new games will therefore be very useful, saving people a significant amount of time and effort.

Check This Out! is a Natural Language Processing (NLP) based recommender system which takes in as input some games a person has played and the time they have spent playing them, as well as a new game the player have never played before, the model will predict how much the player will like the new game by predicting the amount of time they will spend play that new game.

Overall Software Structure

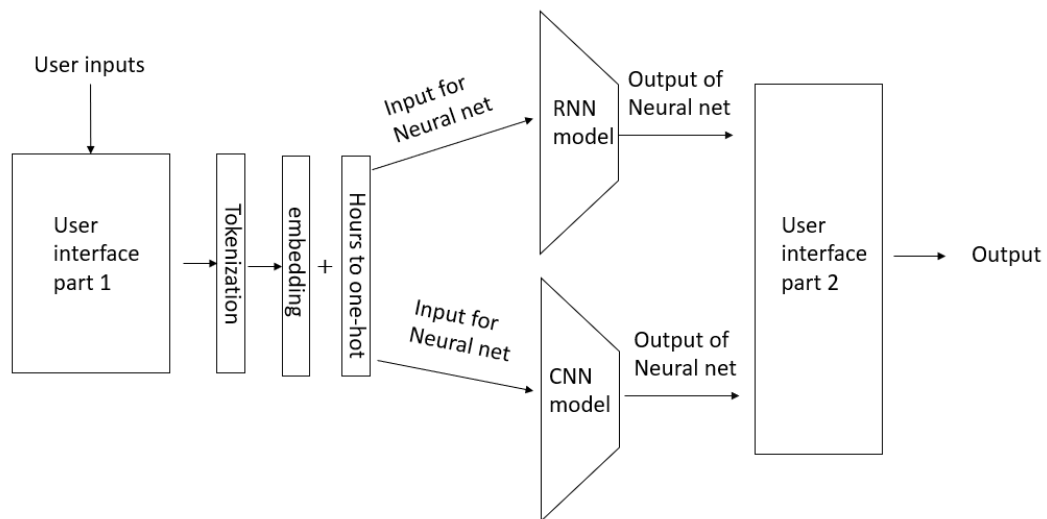


Figure 2 General Structure of the Software

The software can be divided into 3 parts, which will be referred to as “user interface”, “machine learning models” and “input preprocess”.

The “user interface” retrieves inputs from, and present the outputs to the user. It will ask for the names of the top 10 video games the user has been playing and the number of hours they have played them, as well as the name of the new game that we want to make predictions of. After the computation is finished, the user interface will present the predictions made by the machine learning models.

The “input preprocess” converts the user input into appropriate format for the two models used in the software. The name of video games will be sent into a lookup table, so that the corresponding abstracts are obtained. The abstracts will then be tokenized and turned into word embeddings. The number of hours played for the games are also one-hot-encoded into 4 categories (0-10 hours, 10-35 hours, 35-85 hours and above 85 hours).

The “machine learning models” consists of two models in parallel. The preprocessed information will be fed to both models and the models will output a one-hot-encoded prediction of number of hours that the player will be playing the new game.

Sources of Data

The dataset used for training the neural networks will be built on top of a previously collected dataset that was uploaded to the Kaggle Website called Steam Video Games[3]. As seen in figure 3, each entry in the pre-existing dataset has 5 columns: the first column is user ID of a specific player, the second column is the name of a video game, the third column is the action that the specified player performed on the video game, the action can be either purchase or play, if the action is purchase, the fourth column is 1 or 0 representing if the player have purchased this game, and if the action is play, the fourth column represents the amount of hours the player have played the game. The last column is filled with value 0 and have no apparent usage

76767	Counter-S	purchase	1	0
76767	Counter-S	play	365	0
76767	Call of Du	purchase	1	0
76767	Call of Du	play	271	0
76767	Total War	purchase	1	0
76767	Total War	play	207	0
76767	Call of Du	purchase	1	0

Figure 3 Sample entries to the original dataset

Not all of the information were useful from the original dataset, so the dataset was first cleaned by removing all entries containing the action ‘purchase’ in the third column, and then the third column entirely. Entries of players with less than 11 total games played were also discarded since the models requires 10 games as inputs and 1 game as the label.

Afterwards, the dataset was reshaped by combining the most played 11 games as well as the played hours for every player into one single entry, in the form of: [Game #1, Hours Played #1], [Game # 2, Hours Played #2] ... [Game #11, Hours Played #11] so that each entry to the dataset represents one player. However, the new dataset turned out to be highly unbalanced since for most players, their number 11 most played game often had a much lower played time, therefore the label for most entries turned out to have very small values, furthermore, since the label is the hours played for the 11th game, it will always be less than the hours played from the inputs, which may result in biases within the models. To account for this, data augmentation was performed by rotating every entry to the dataset 10 times, creating 10 new entries for every playe.

Lastly, a web scraper was created and used to extract the abstracts for all games appeared in the first dataset, they were stored into a separate dataset where the first column contains the names of the games and the second column contains the abstracts of the games. The second dataset was used as a lookup table for the software.

Machine Learning Models

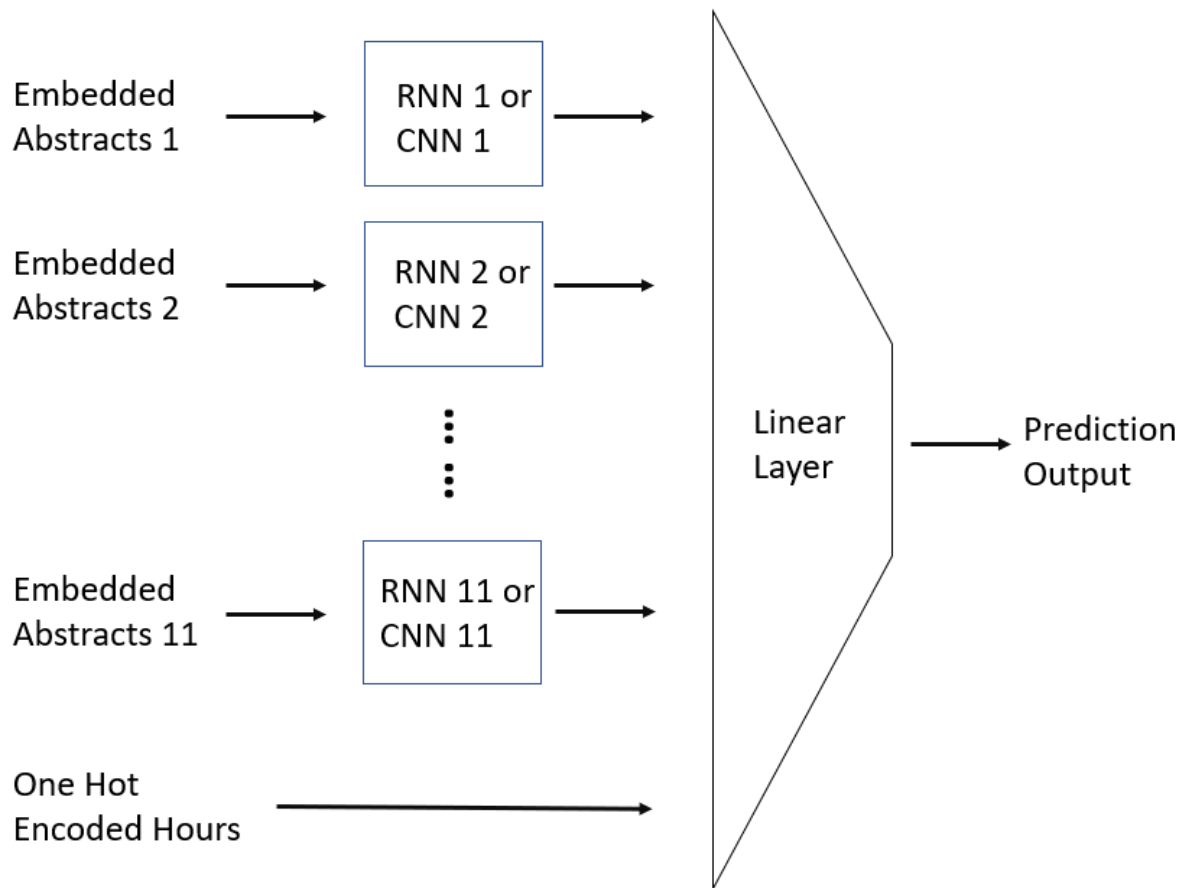


Figure 4 Model Overall Structure

When designing the neural network model, two different approaches were used. As seen in figure 4, the two approaches share a similar overall structure that contains two layers, the first layer focuses on NLP and has 11 separate networks, each network individually takes in the word embedded abstract for one of the 11 games and encodes them, the second layer is a linear layer, which takes in the outputs from the first layer along with the one-hot encoded hours data and outputs a final prediction.

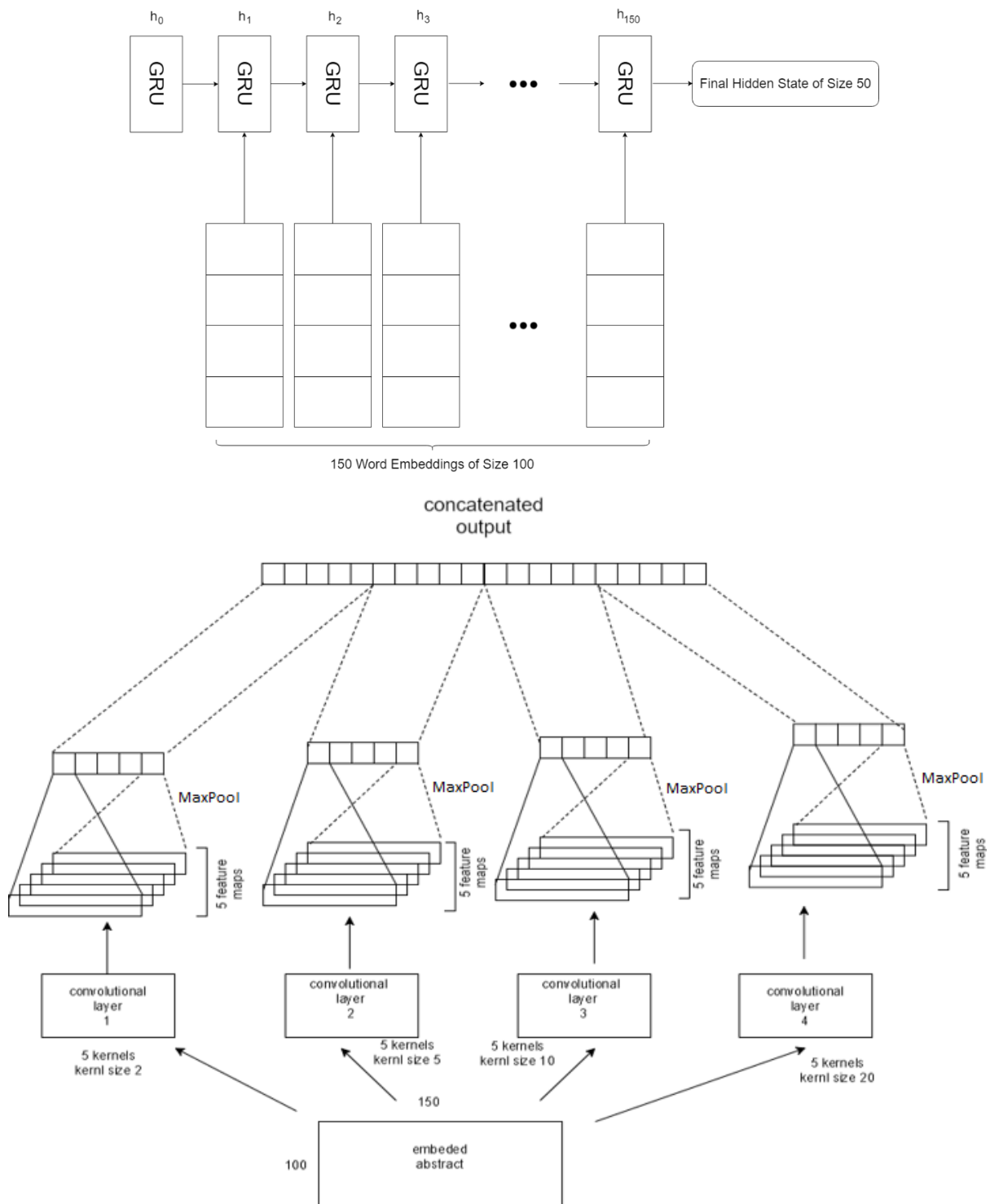


Figure 5 NLP Layer: model 1 (top), model 2 (bottom)

The key difference between the two models is that the first model uses recurrent neural networks in the NLP layer (first layer) and the second model uses convolutional neural networks instead. The details of the two NLP networks can be seen in figure 5.

In the first model, after an abstract is fed to a recurrent neural network, the sequence of word embeddings are put through a gated recurrent unit with a hidden state size of 50 and the final hidden state was used as the output for that particular network.

In the second model, the sequence of word embeddings are fed into four 1D convolutional layers in parallel, each layer has 5 kernels but with different sizes (sizes of 2, 5, 10, 20). Each convolutional layer outputs 5 feature maps, which would then be fed into a maxpool layer, so that the element with the largest activation would be extracted. By combining the output of each maxpool layer, the convolutional section of the model would give a 1*20 vector as its output.

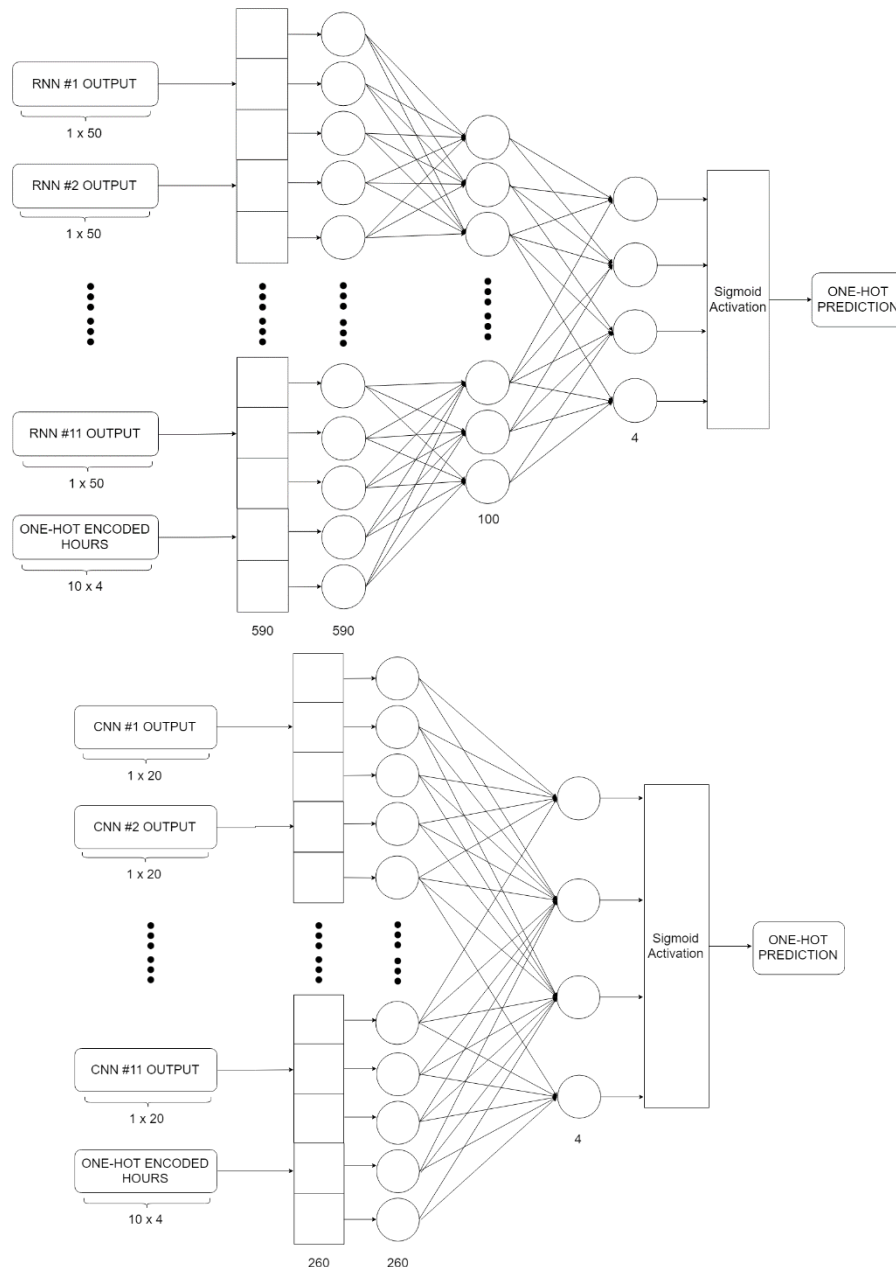


Figure 6: Linear Layer: model 1 (top), model 2 (bottom)

The linear layer (second layer) for both models serves the same purpose, they both takes the concatenated version of the 11 outputs from the first layer as well as the one hot encoded version of the 10 hours data, and outputs a final prediction. There are some differences in the actual structure of the multilayer perceptron between the two models, as seen in figure 6.

In the first model, since the output of each recurrent neural network has a size of 50, its linear layer has a input layer size of

$$11 \times 50 + 10 \times 4 = 590$$

It also has a hidden layer of size 100 and an output layer of size 4, the output layer is followed by a sigmoid activation function which gives a prediction in terms of how many hours the player will be playing the new game, in one-hot-encoded format.

On the other hand, the second model has an output of 20 for each convolutional neural network from the first layer, and therefore the linear layer has an input size of

$$11 \times 20 + 10 \times 4 = 260$$

It only has a single output layer with 4 neurons, also followed by a sigmoid activation function which gives the prediction in one-hot form.

The reason why the two models focused on recurrent and convolutional neural networks, is that both networks are good at obtaining the sequential information about the data, which are crucial since the goal of the models is to look through the abstracts of each video game and recognize patterns of games that users enjoy. More specifically, when processing an abstract, the recurrent neural network has a memory of previously entered words and can encode the information more accurately, the convolutional neural network having parallel convolutional layers with different kernel sizes, will be able to identify phrases of different length, and consequently capture information in the abstracts.

Training, Validation and Test

In the process of training, many designs were implemented so that a high accuracy could be achieved by the model. At the beginning of the training process, the team focused on designing and training two separate models, where the first model is an autoencoder that compresses the embedded abstracts, the fully trained autoencoder would then be transferred into the second model for making predictions.

Later, after receiving suggestions from the professor and teaching assistants of the course, the team decided to combine the two models into one and replacing the autoencoder with a convolutional or recurrent layer. This design, after performing grid search on the hyperparameters, achieved the highest accuracy.

Both the convolutional and recurrent versions of the models were trained on the training dataset for 2 epochs by which the accuracies were observed to have plateaued. The best accuracies achieved by both models as well as the confusion matrices on the test data can be observed in tables below.

	Model 1(RNN)	Model 2(CNN)
Testing Accuracy	76.3%	71.3%
Validation Accuracy	85%	75%

Table 1: Best Accuracies for the machine learning models

Prediction \ Ground Truth	0-10 hours	10-35 hours	35-85 hours	Above 85
0-10 hours	212	24	1	7
10-35 hours	33	240	4	4
35-85 hours	7	65	119	19
Above 85	26	38	39	173

Table 2: Confusion Matrix for Model 1

Prediction \ Ground Truth	0-10 hours	10-35 hours	35-85 hours	Above 85
0-10 hours	146	39	10	22
10-35 hours	1	216	0	1
35-85 hours	0	93	104	21
Above 85	0	31	32	155

Table 3: Confusion Matrix for Model 2

From the confusion matrices, it can be observed that, when the models made incorrect predictions, it was often the case that this prediction fell into the neighbouring category of the correct answer, which shows that the model have learned to make predictions that are reasonably accurate.

The models' performances seemed reasonable, however, a major issue surfaced when the team attempted to implement the model for real world uses. It was noticed that after entering 10 games and hours to the model, no matter what game was entered as the new game for prediction, the output, although with some variation in values, often belonged to the same one-hot category. Upon further exploring this issue it was realized that the models when deciding the final prediction are heavily biased towards the hours data instead of the natural language data as the team had hoped for.

In order to reduce the model's bias towards "number of hours" data, many modifications to the models were made, but the results were still not ideal. For example, a model using convolutional layers was implemented that did not take in the hours data as inputs, however, the accuracy of this model remained at around 25% throughout training, which suggest that the model underfits. The dataset was also modified to have entries with the same 10 games and hours as inputs but with different new games as the label, however, the resulting model had a much lower validation accuracy of 52% and test accuracy of 47% and although with some improvements, the model was still heavily biased towards the hours data.

In the end, the models, although with reasonable accuracies, were not able to effectively perform the task that it was originally designed for.

Ethical Issues

A high accuracy neural network that can successfully predict people's preferences must know a lot about those people, and if the model is implemented at a large scale then it will hold information about almost everyone. Someone with bad intentions may find a way to obtain this information and use them to exploit other people. Essentially, personal privacy will be at a much greater risk than before

Key Learnings

The major lesson we learned from this project is that, high accuracy of the model does not necessarily imply good performance in reality. If the project starts over again, instead of spending a large amount of time fine tuning the hyperparameters to reach a higher accuracy, we would start implementing the model at the point that an satisfactory accuracy has been reached, so that we can observe how well the model performs when it is implemented.

References

- [1] Statista. (2018). Number of games released on Steam worldwide from 2004 to 2017. [online] Available at: <https://www.statista.com/statistics/552623/number-games-released-steam/>
- [2] Kuchera, B. (2018). Report: 7,672 games were released on Steam in 2017. [online] Polygon. Available at: <https://www.polygon.com/2018/1/10/16873446/steam-release-dates-2017>
- [3] Robbins, A. (2016). Steam Video Games. [online] Kaggle.com. Available at: <https://www.kaggle.com/tamber/steam-video-games>