Exploring the Effectiveness of Neural Network in Predicting User Satisfaction with Web Searches Based on Task-Related and Users Behavioural Features

Haoting Chen

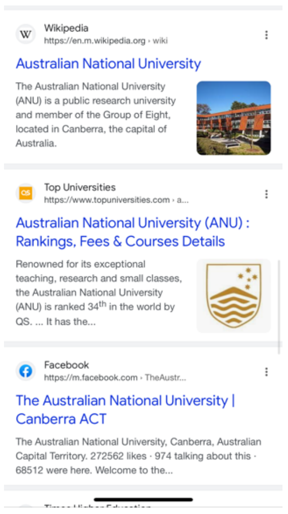
School of Computing, Australian National University
u7227871@anu.edu.au

Abstract. In modern days, web search has become an inseparable daily activity for everyone. While web search allows us more easily to access information around the world, the experience of web search is also important. However, the experience matrices, such as user satisfaction, usually require conducting surveys to obtain, which can make it hard for search engine providers, like Google, to receive instant user feedback on their products. This study aims to address the issue by building up neuron networks to predict user satisfaction. The dataset used for training is collected by Kim et al. [1] study, which captures not only user satisfaction with web searches but also a range of metrics of behaviour, such as eye-tracking metrics and various scenarios, such as different snippet sizes provided in the result page, that may be correlated with user satisfaction. Two types of neural models are used to solve this prediction problem. One is a simple 2-layer feed-forward fully connected network. The other is a constructive network, called Casper [2], in which the size of the network can grow during training to dynamically adapt to problems varying from simple to complex. The results show that the performance of the two models is very close after fine-tuning their hyperparameters. Both models have 70% average accuracy in capturing the true satisfaction number (an integer ranging from 1 to 7), with a tolerance of one unit difference. However, both models experience the overfitting issue to the training dataset, which stops the test accuracy from further increasing.

Keywords: web search, Casper, neuron network, satisfaction

1 Introduction

In the digital age, web searches have woven themselves into the very fabric of our daily routines. For search engine providers, ensuring that users have a satisfactory experience is essential. Traditionally, satisfaction can be obtained through user surveys, but this can face some challenges, such as prolonged feedback and dishonest answers from the participants. In addition to directly obtaining satisfaction through the user's opinion, satisfaction may also be estimated by investigating and modelling the correlation to the user's behaviours during web searches. For example, the time a user spends on a specific search task, the movement of their eye may somehow reflect their satisfaction. Therefore, a regression model can be built to take user behavioural and environmental matrices as parameters and output an estimated satisfaction number. Some parameters may be implicitly obtained through the background user behaviour tracking system in users' devices after granting their consent, such as the during it takes for a user to make their first click after being presented with the search results measured in Kim et al. [1] study. The search engine provider then can use the uploaded data from the users' apps and the model to predict the satisfaction level. However, the question is about how effective this kind of model is and how far it is from practical use still requires more research. Section two of this paper provides a brief description of the dataset, the neuron network, specifically, about what information the dataset provides, and how the simple 2-layer neuron network and Casper are built and trained. Section three contains the methods used, and the decisions made during data pre-processing, model validation, and testing process. Section four summarises the summarise the results through model testing and Section five concludes the paper.


Wikipedia
https://en.m.wikipedia.org › wiki
Australian National University
The Australian National University (ANU) is a public research university and member of the Group of Eight, located in Canberra, the capital of Australia.

Top Universities
https://www.topuniversities.com › a...
Australian National University (ANU) : Rankings, Fees & Courses Details
Renowned for its exceptional teaching, research and small classes, the Australian National University (ANU) is ranked 34th in the world by QS. ... It has the...

Facebook
https://m.facebook.com › TheAustr...
The Australian National University | Canberra ACT
The Australian National University, Canberra, Australian Capital Territory. 272562 likes · 974 talking about this · 68512 were here. Welcome to the...

Figure 1: Example of snippets on the result page, the last snippets are shorter than the first two.

2 Dataset and Model Used

2.1 Dataset

The dataset collected by Kim et al. [1] provides many useful features for training the model. The primary objective of the paper is to investigate the influence of snippet size on users' satisfaction, confidence, and performance when interacting with mobile web search results. Snippets in this context refer to the textbox holding the description text of a web page found. As shown in Figure 1, the data collection process involves asking participants to use mobile devices to conduct searches and make relevant judgments based on the search results and the snippets provided. The snippet's size varies from short, medium and long. In addition to recording satisfaction, confidence, and performance, data about user behaviour is also collected, such as the type of task performed, time spent on tasks, and eye-tracking results. These data should also correlate with the data such as snippet size and satisfaction. For example, doing a specific task when a specific snippet size is shown on the web page may lead to specific user satisfaction, which is exactly what the model should learn. Therefore, the dataset not only helps estimate snippet size but also provides features and labels for the model to learn a function to predict satisfaction. In the training process, 15 features from the dataset were selected and the satisfaction column served as ground true. The satisfaction consists of 7 levels encoded as an integer ranging from 1 to 7 in the dataset. Since the output of the model is a float number, the predicted result will be rounded and bounded with 1 to 7.

2.2 Feed-Forward Neuron Network with a Single Hidden Layer

One model used to solve the satisfaction prediction task is a simple feed-forward neuron with only the input layer of size 15 accepting all the features, a hidden layer of some size with sigmoid as the activation function, and the output layer of size 1 outputting a floating number of the satisfaction score. Mean squared error (MSE) is used as the loss function which is the average square difference between the prediction satisfaction score and the ground true. The network is learned through a common learning process with backpropagation, mini-batch gradient descent, and Adam optimiser with L2 regularisation. Choosing such a network is due to its simplicity but also the potentiality. According to the Universal Approximation Theorem, a single hidden layer feed-forward network can approximate any continuous function using a finite number of neurons, under mild assumptions on the activation function. The theorem is proved by George Cybenko (1989) by using sigmoid [3] and Kurt Hornik et al. (1989) further make the theorem generalise to a boarder class of activation functions [4]. Even though a deep network is more popular it can represent complex functions in a more compact way and form feature hierarchies. No matter when training a shallow or deep network, selecting the number of neurons and layers is still a thorny problem.

2.3 Casper

The other model used is Casper proposed by Treadgold and Gedeon (1997) [2]. Casper is a constructive neuron network in which size and structure grow during the training process instead of defining before training. In this way, the problem of selecting the number of hidden layers and neurons is automatically handed during the training and the programmer can focus more on tuning other hyperparameters. The output of the algorithm of training a Casper is in the following.

- The network initialises a minimal structure in which there is only an input layer connecting to an output layer.
- Train the network until the training error has fallen below 1% of its previous for a time period calculated in Formula 1, where P is defined by programmers and N is the number of currently installed neurons.
- Add a new neuron connecting to all input neurons and hidden neurons in lower layers and the output neuron, as shown in Figure 2. The newly added weights should be initialised but the old weights in the network remain unchanged.
- Repeat 2 until the training result is satisfied.

$$Period = 15 + P * N \qquad (1).$$

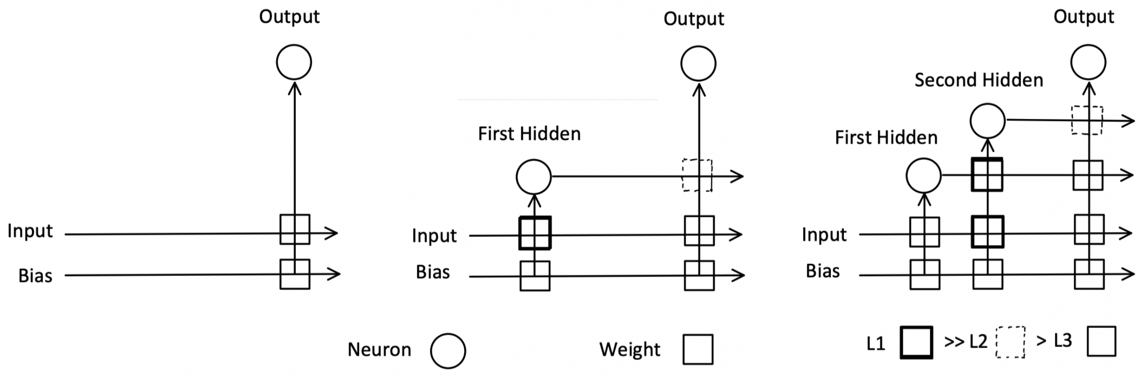


Figure 2: initial network, network after one hidden neuron is added, and network after two hidden neurons are added.

When training the network against the gradient of the loss function, a modified version of resilient backpropagation (R-prop) is used. R-prop is proposed by Riedmiller and Braun in 1993 [5]. Compared to normal backpropagation that updates weight based on their derivatives. Weights are updated based on their signs and dynamic step size. The Simulated Annealing is used as weight decay (regularisation) which makes the amount of weight decay proportional to its squares [6], as shown in Formula 2, where Hepoch is the number of epochs passed since the addition of the last hidden neuron.

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta w_{ij}} - k \cdot sign\left(w_{ij}\right) \cdot {w_{ij}}^{2} \cdot 2^{-0.01*Hepoch} \qquad (2)$$

In addition to R-prop to weight decay, the Casper optimiser also applies different learning rates to 3 different parts of the network. As shown in Figure 2, L1, the weights connecting the last hidden neuron and input neurons to the new hidden neurons should learn the fastest, followed by L2, the weights connecting the last hidden neurons to the output neurons, and the rest of the weights, L3.

3 Methodology and Implementation

All implementations mentioned in this section can be found in the source code associated with this paper. For example, several Jupyter Notebook files have recorded how data is processed step by step and how the best hyper-parameter of a model is selected through several trials. Please refer to the readme file for details.

3.1 Data Pre-Processing

Relevant File: data_preprocessing.ipynb

(1) While the original dataset has provided 26 columns, i.e., 26 possible features, not all of them are useful for prediction. For example, the id-relevant columns are dropped, such as 'Task_num'. (2) There are also redundant columns such as column TTF (Time to first click) and column log (TTF). For a feature occupying two columns by its original value and its logarithmic transformation values. If the original data is skewed, use the log one, and vice versa. The log transformation helps a single feature of data to become more normally distributed and normally distributed data is preferred by the machine learning model since many assuming on training is made based on normal distribution. (3) The categorical variables, such as nav/info in column “Task_Type” are encoded into integer numbers. (4) Finally, all column except the target column is normalised by subtracting their mean and dividing by their standard deviation. This ensures features with large number values are equally important as the features with small number values. More advanced techniques can be assigned to different weights to different features after finding out their difference in significance, but this was not implemented in the study.

3.2 Building Neuron network with a single hidden layer in PyTorch 2.0

Relevant File: simplenet_v0.py

The building of the network is followed by several key steps. First, define the model class by specifying the layer and the interactions between layers. Then, define the optimiser and loss function, in this problem, they are Adam optimiser and MSE. Next, write a training function. Insides, perform forward pass and use loss function to calculate the loss. Then, perform a backward pass using “loss.backward()” to calculate the gradients and use gradients to update weights by calling “optimiser.step()”. Stop until a pre-defined epoch number has been reached.

3.3 Building Casper in PyTorch 2.0

Relevant File: casper_v0.py, casper_spiral.ipynb

In order to have a changeable structure, in the class definition of Casper, the hidden layers are stored in a list. In a forward pass, the input is passed to the first hidden layer in the list and then keeps updating while transferring a list of hidden layers. In the end, the last hidden layer sends the result to the output layer to output. All hidden layers use sigmoid as their activation function. In the Casper model class, there is also a helper function that can add a neuron to the model when it is called.

The training function consists of two loops, the inner loop trains the model through a number of epochs and stops until the loss no longer decreases for some time. The outer loop manages the addition of hidden neurons into the model. After the inner loop is complete, the outer loop adds a neuron to the model and then updates the optimiser function as the number of trainable parameters has changed. The outer loop stops when the number of hidden neurons has exceeded a certain number.

The optimiser is the key part of the model. Casper uses different learning for different parts of the model. Luckily, PyTorch supports making model weight parameters into groups and for each group a specific learning rate is assigned. PyTorch also supports R-prop, but the Simulated Annealing weight decay is not natively supported. To address this issue, a child class of Rprop is made to modify the behaviour of the original Rprop class. The logic is to access the gradients of weights of the model and subtract them with the weight decay term.

To demonstrate the correctness of the implementation of Caspor, the model is run on a classification problem, called two spirals, which consists of two groups of data points lying on two spirals, respectively, as shown in Figure 3. The model performed quite well on this problem with a high accuracy of 97.48% with a standard deviation of 3.92%. A similar benchmark is done on the original paper of Casper [2] and a similar accuracy of 97.80% is measured.

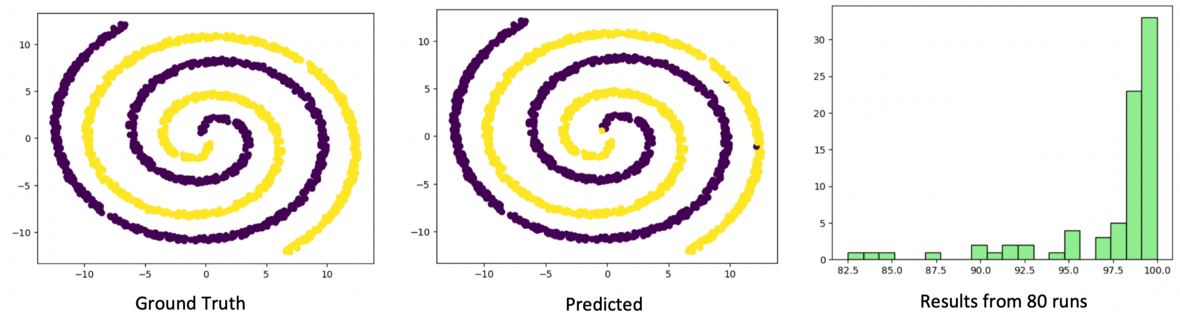


Figure 3: Performance of model on two spiral problems

3.4 Method of Validating the Single-Hidden-Layer Network

Relevant File: simplenet_validation.ipynb

The hyperparameters needed to search to find out the best model setting are:

- Optimiser used, (default = Adam), candidates = [Adam, SGD]
- Hidden layer size, (default = 500), candidates = [5, 50, 100, 1000]
- Batch size, (default = 10), candidates = [1, 5, 10, 30]
- Number of epochs, (default = 500), candidates = [100, 200, 500, 1000]
- Learning rate, (default = 0.001), candidates = [0.01, 0.001, 0.0001]
- Weight decay strength, (default = 1e-5), candidates = [1e-5,1e-4,1e-3,1e-1]

The study uses a sequential approach to hyperparameter pruning. It is sometimes considered as a greedy approach. The approach starts by initialising all hypermeter to some default value based on experience. Then loop through all hypermeters. In each loop, vary only one hyperparameter with all others held default value. Record the best one leading to the lowest loss and overwrite its default value. Proceed to the next hyperparameters until all hyperparameters are inspected. As mentioned, this method is greedy a hyperparameter is optimised based on the immediate best choice. Therefore, the scenario in which a specific combination of hyperparameters can lead to a better result may be ignored. However, this is more computationally efficient compared to Grid Search, which loops through all the possible combinations of hyperparameters.

The method used to test the model with different hypermeter layouts is 10-fold cross-validation. Since the dataset only contains 261 samples, further splitting 20% out for validation may influence the training. Also, the distribution of training and validation tests may not be the same since the sample size is small. Therefore, 10-fold cross-validation is used. Instead of training and validating a single hyperparameter 10 times, the whole 10-fold cross-validation is run four times with four different seeds. This mimics the randomness of modal initialisation. Therefore, for a single hyperparameter, the model is trained 4 * 10 = 40 times.

The primary metric used to compare the effectiveness of the model under different hyperparameter settings is MSE. Although the predicted labels are discrete (integers from 1 to 7), the problem is essential a regression tasks, where MSE is used instead of cross-entropy loss. For example, if the prediction is 4 and the ground true is 5, in classification, it is considered totally wrong, but in regression problems, 4 is still somehow close to the ground true and should have a smaller loss than having a prediction of 3. The second metric used is accuracy calculated based on the rounded output of the model and the ground truth. However, since it is a regression problem, the accuracy may not be able to evaluate the performance of the model well. If the ground truth data points scatter around the regression line, the accuracy may even go to zero, but the model may generalise the data well. Therefore, a one-unit tolerance is used when calculating the accuracy. For example, if the ground true is 5 and the predicted result is 4 or 3, they are still considered correct. Even though, the accuracy aims to give people a more intuitive view of the effectiveness of the model. The best metric to summarise the performance of the model is still MSE and its variance.

3.5 Method of Validating Casper

Relevant File: casper_validation.ipynb

The hyperparameters needed to search to find out the best model setting are:

- P, (default = 10), candidates = [0.001,0.1,1,3,5,10]
  P controls how long it lasts until the weight does not drop before entering the next step and adding neurons. Have an effect on the number of epochs the model runs.
- Number of hidden neurons, (default = 5), candidates = [0,1,2,10]
- D, (default = 0), candidates = [0, 0.05]
  Effectiveness of weight decay
- Learning rate, default = [0.001, 0.005, 0.2], candidate = [[0.0001, 0.0005, 0.02]]

The method of selecting the best same as 3.4, which tweaks each parameter one by one and uses the best one on further hyperparameter selections. The validation method and metrics are also the same, which trains the model 40 times on a simple hyperparameter and uses MSE primarily when evaluating a hyperparameter.

3.6 Method of Comparing the two models

After selecting the best hyperparameter sets for each model. The two models are trained 80 times with different train and test data split and random seed, respectively. This makes the result more robust as it can average not only the randomness during model initialisation but also the randomness when selecting the training and testing set. After this process, 80 test loss and accuracy are generated for each model and they can form a distribution graph.

By visualising the distribution graph and conducting a two-sample t-test to obtain the P-value, we can conclude if one model is better than the other given a confidence level or we cannot reject the null hypothesis that the performance of two models has no significant difference.

3.7 Method of evaluating the effectiveness of neuron network on the satisfaction prediction task

The L1 score is used to evaluate the overall effectiveness of the model on the task since it is more interpretable. The L1 score directly indicates the average distance between the predicted result and the ground, which is what daily users more care about. The smaller the L1 score, the narrower the distance between the model output to the target and the model can more accurately predict a discrete satisfaction level after rounding the model output.

4 Results and Discussion

4.1 Validation result of the single-hidden-layer model

During the validation process, it was found that the following hypermeters combined can lead to an optimal result, with an overall accuracy of 70.38% with one unit tolerance and a loss of 1.89. Table one summarises the key process of how each of these hyperparameter is selected.

- Optimiser used = Adam
- Hidden layer size = 5
- Batch size = 10
- Number of epochs = 500
- Learning rate = 0.001
- Weight decay strength = 0.001

	Optimiser	Hidden	Batch Size	Epoch Number	Learning Rate	Weight Decay
Selected	Adam	5	10	500	0.001	1e-3
Default	Adam	50	10	500	0.001	1e-5
Notes (MSE)	Adam:1.96, SGD:2.12, Adam is better	5: 1.95, 50: 2.06, 5 is better	All other sizes have more loss	All other epoch numbers have more loss	All other learning rates have more loss	1e-3: 1.96 1e-5: 1.98 1e-3 is better

Table 1

A more restricted analysis can also be considering their standard deviation of MSE and computing a t-test to see if two parameters lead to significant differences in MSE. However, this may involve more work. In the validation process, our goal is to perform a number of trials to choose the best hyperparameters and test them instead of making judgements on their loss difference. Therefore, only considering the loss generally helps us quickly find out the optimal hyperparameters.

4.2 Validation result of Casper

For Casper, it was found that the following hypermeters combined can lead to an optimal result, with an overall accuracy of 71.88% with one unit tolerance and a loss of 1.93.

- P = 0.01
- Number of hidden neurons = 1
- D = 0
- Learning rate = [0.001, 0.005, 0.2]

	P	Number of hidden neurons	D	Learning rate
Selected	0.01	1	0	[0.001, 0.005, 0.2]
Default	10	5	0	[0.001, 0.005, 0.2]
Notes (MSE)	10: 3.21, 0.01: 2.30, 0.01 is better	5: 2.30 1: 1.94	0.05: 1.94 0: 1.94 A simpler one is preferred	All other learning rates have more loss

Table 2

4.3 Model Comparison

The benchmark result is obtained by running two models separately 80 times with a list of different seeds. However, the seed list used for both models is the same. The seed controls how testing and training data is split and model initialisation. This ensures that both models are fair, e.g., both are allocated to a bad testing and training dataset split by the same seed.

Figures 4 and 5 show the distribution of MSE and accuracy of the two models respectively and Table 3 summarises the mean, median, and standard deviation of the MSE and accuracy. By observing the graph, it seems the performance of the two models is quite similar. A p-test further confirms this estimation with a p-value calculated from the MSE of both models being 0.57, which is higher than 0.05. Therefore, we cannot reject the null hypothesis, suggesting that there isn't enough evidence to claim a statistically significant difference between the two models' MSE.

	Mean (MSE)	Median (MSE)	Std (MSE)	Mean (acc)	Median (acc)	Std (acc)
Simple Net	1.894	1.823	0.441	70.38 %	69.23%	7.820%
Caspor	1.935	1.899	0.453	71.88%	73.08%	7.898%

Table 3

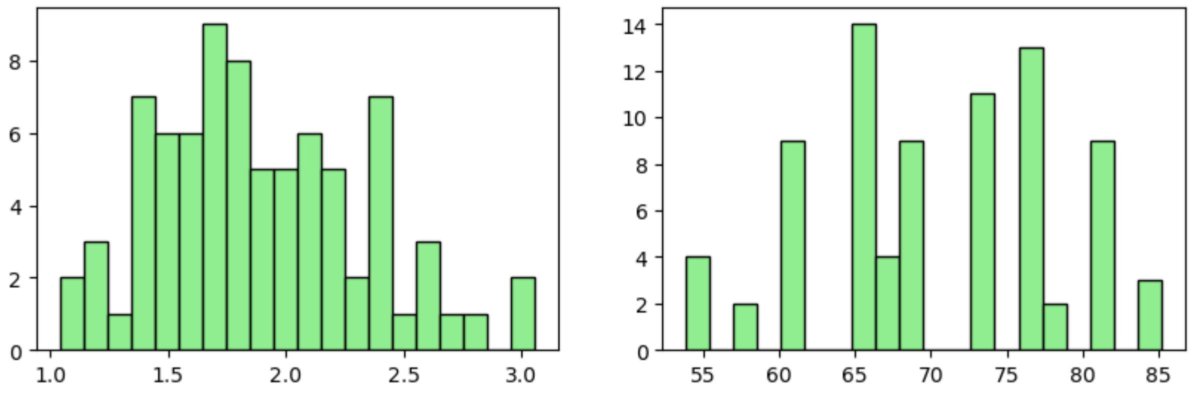


Figure 4: Distribution of MSR and accuracy of single-hidden-layer network

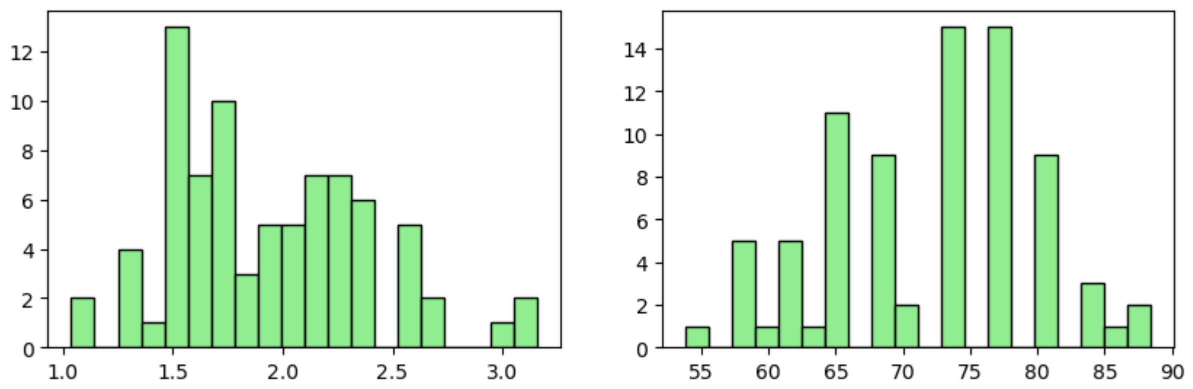
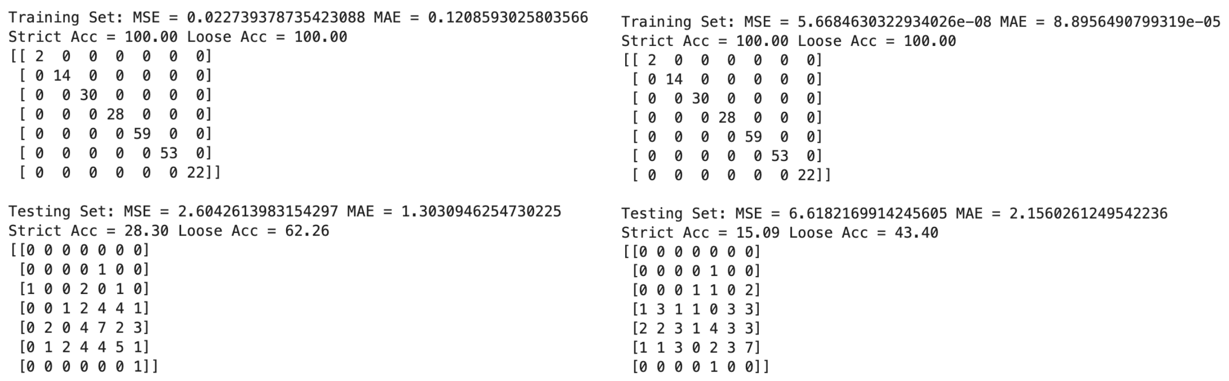


Figure 5: Distribution of MSR and accuracy of Casper

4.4 Finding: model overfitting.

When choosing the hyperparameters for both models, it was found that making the model complex or training long can make the model nearly perfectly fit the dataset with an accuracy of 100%, as shown in Figure 6. This shows that both models are capable enough of modelling all training data points but do not generalise well. For example, by setting the hidden neuron = 1000, epoch = 1000 for the 2-layer network, and P=20 and Max Neuron = 20 for Casper and random seed = 4660. It can be found that both models learned well on the training dataset, with losses closer to 0 and the confusion matrix becoming diagonal. However, when it comes to the test set, the loss increases dramatically, especially for Caspor.

To prevent overfitting of both models, during the validation process, the complexity of the network has been adjusted to be as simple as possible to capture more general features to increase accuracy. By referring to Tables 1 and 2, it can be shown that the decrease in epoch number and P (Caspor) and, the decrease in hidden neurons for both models can improve the MSE for a small amount. However, the performance of the models seems to be stuck after trying different hyperparameters.



```
Training Set: MSE = 0.022739378735423088 MAE = 0.1208593025803566
Strict Acc = 100.00 Loose Acc = 100.00
[[ 2  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0]
 [ 0  0 30  0  0  0  0]
 [ 0  0  0 28  0  0  0]
 [ 0  0  0  0 59  0  0]
 [ 0  0  0  0  0 53  0]
 [ 0  0  0  0  0  0 22]]

Testing Set: MSE = 2.6042613983154297 MAE = 1.3030946254730225
Strict Acc = 28.30 Loose Acc = 62.26
[[0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0]
 [1 0 0 2 0 1 0]
 [0 0 1 2 4 4 1]
 [0 2 0 4 7 2 3]
 [0 1 2 4 4 5 1]
 [0 0 0 0 0 0 1]]
```

```
Training Set: MSE = 5.6684630322934026e-08 MAE = 8.8956490799319e-05
Strict Acc = 100.00 Loose Acc = 100.00
[[ 2  0  0  0  0  0  0]
 [ 0 14  0  0  0  0  0]
 [ 0  0 30  0  0  0  0]
 [ 0  0  0 28  0  0  0]
 [ 0  0  0  0 59  0  0]
 [ 0  0  0  0  0 53  0]
 [ 0  0  0  0  0  0 22]]

Testing Set: MSE = 6.6182169914245605 MAE = 2.1560261249542236
Strict Acc = 15.09 Loose Acc = 43.40
[[0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0]
 [0 0 0 1 1 0 2]
 [1 3 1 1 0 3 3]
 [2 2 3 1 4 3 3]
 [1 1 3 0 2 3 7]
 [0 0 0 0 1 0 0]]
```

Figure 6: overfitting scenario

To further overcome the bottleneck of test accuracy, several methods have been tried, including pruning, early stopping, and column-dropping based on the change in model loss after dropping this column to select the most important column. However, these trials have little effect on improving the model performance and sometimes make the model perform worse.

The problem may come from the nature of the dataset. In our task, user behaviours are used to predict the satisfaction level. However, user behaviour may contain a lot of random noise, letting the data scatter around. Therefore, even though the best regression line is found to generalise the data points. The line can still lead to a high MSE because most data points are not closer to the line.

The other potential issue that prevents accuracy from further increasing is the limited sample size. Therefore, there are only 261 samples and may not be enough to represent the population. During the testing and training dataset split. The distribution of the two sub-sets may not be the same since the sample size is small. After training in the dataset on one distribution, testing on another data distribution causes the domain shift problem and the model is not able to handle this.

4.5 Overall judgement on models' effectiveness

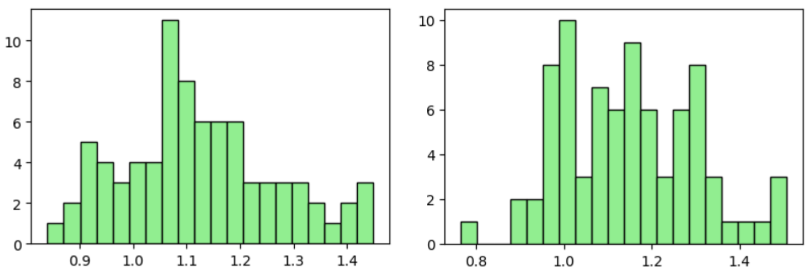


Figure 7: L1 loss for both models

	Mean	Median	Std	Max	Min
Simple Net	1.245	1.101	0.1421	1.448	0.8398
Casper	1.145	1.144	0.1525	1.505	0.7671

Table 4

Based on the L1 matrices and the distribution graph, it was shown that the model indeed has learned some features from the dataset and is able to use them to accurately predict user satisfaction given some extent of tolerance. The L1 score of the model are both around 1.1, which tells us that on average the model prediction has a 1.1 satisfaction level different from the ground truth. Even though a satisfaction level of 7 is predicted as 6, it may not make a significant difference. This is because sometimes it is hard for participants to choose between 6 and 7 when they are asked for the confiscation. The maximum of L1 loss across all samples provides some guarantee on the model performance and they are not too far from the mean loss. The standard deviation of L1 loss is very small, which means the loss is very consistent and the performance of the model is stable.

5 Conclusion

This study has explored the effectiveness of neural networks in predicting user satisfaction with web searches, using a dataset that encompassed various user behaviours. Both the single-hidden-layer network and the Casper model demonstrated the potential to predict user satisfaction, through achieving an accuracy of around 70% accuracy with a tolerance of one unit difference in satisfaction levels. However, for this dataset, both models exhibited a tendency to overfit for a little larger network complexity. Therefore, both models are set to be very small in order to achieve the best validation score. For, Casper, only one hidden neuron is added in addition to the output and input layers. Hence, the full capability of Capsor may be exhibited and this is probably why it cannot beat a simple single-hidden-layer network.

The L1 score, the average distance between the model's predictions and the ground truth, is used to measure the overall performance of models on the satisfaction prediction tasks. On average, the predicted user’s satisfaction is 1.1 units away from the true values. This shows that while the models have learned from the dataset to some extent, there's room for improvement.

6 Future Work

The conclusions drawn from this research are based on a limited dataset of 261 samples. Future studies could try using a larger data set to develop a more generalised model with higher accuracy and less overfitting. Additionally, exploring other neural network architectures to solve the satisfaction prediction task might find the opportunity to improve prediction accuracy.

To further investigate the models’ effectiveness, the model may be tested in real-world scenarios. For instance, deploying this model into a search engine help search company predict user satisfaction. The evaluation could be based on whether the model led to a positive return for the company, such as reducing the overhead of conducting user surveys and helping the company with decision-making based on user feedback.

In short, while the models in this study have demonstrated the potential to predict user satisfaction with web searches, there are ample opportunities for developing a more comprehensive model to solve the task. This paper may become a starting point for developing such as new model.

References

1. Kim, J Paul, T Sankaranarayna, R Gedeon, T Yoon, HJ 2017, 'What Snippet Size is Needed in Mobile Web Search?', *CHIIR '17: Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval, March 2017*, <https://dl.acm.org/doi/10.1145/3020165.3020173>.
2. Treadgold, NK, Gedeon, TD 1997, 'A Cascade Network Algorithm Employing Progressive RPROP', *Lecture Notes in Computer Science*, vol. 1240, <https://link.springer.com/chapter/10.1007/BFb0032532>
3. Cybenko, G 1989, 'Approximation by Superpositions of a Sigmoidal Function', *Math. Control Signals Systems*, no.2, pp.303-314, <https://web.njit.edu/~usman/courses/cs675_fall18/10.1.1.441.7873.pdf>
4. Hornik, K Stinchcombe, M White, H 1989, 'Multilayer feedforward networks are universal approximators', Neural Networks, vol.2, pp.359-366, <https://www.sciencedirect.com/science/article/pii/0893608089900208>
5. Braun, H Riedmiller, M 1993, 'A direct adaptive method for faster backpropagation learning: the RPROP algorithm', *IEEE International Conference on Neural Networks*, <https://ieeexplore.ieee.org/document/298623/authors#authors>
6. Kwok, T Yeung, D 2005, Bayesian regularization in constructive neural networks, *Lecture Notes in Computer Science*, vol.1112, <https://link.springer.com/chapter/10.1007/3-540-61510-5_95>

Acknowledgement

ChatGPT has been used to polish the writing and perform grammar checks.