Comparing Search Engine Scrolling and Pagination Using Artificial Neural Networks and Genetic Programming

Mark Glanville
u5727146@anu.edu.au

Research School of Computer Science, Australian National University, Canberra, Australia

Abstract. This paper presents the classification of overlapping datasets using two-layer neural networks, distinguishing data points for vertical scrolling and horizontal pagination in search engines for mobile devices. Four neural networks were produced using data preprocessing techniques to extract meaningful information, and using varying thresholds to equate false positive and false negative rates. Similarly, four genetic programming models were produced using the same techniques. These models underwent hyperparameter selection and K-fold cross validation to optimise their classification accuracy. The models had a maximum accuracy of 66.7% and a highest mean accuracy of 59.7%, indicating that the scrolling and pagination control types had unclear decision boundaries and did not have statistically significant distinctions. The preprocessing techniques increased the classification accuracies, although the thresholding techniques had mixed impacts. Future work includes replication studies with more data, more complex solution representations, and further investigation into the genetic models' interpretable results.

Keywords: Neural Network, Genetic Programming, Pagination, Scrolling, Classification, Preprocessing, Variable Threshold

1 Introduction

1.1 Background

Mobile devices are extremely common in everyday life - as of 2017, 91% of internet-connected households in Australia using mobile or smart phones [4]. Small optimisations of their performance and user interfaces can be vital to engage customers and retain user satisfaction. Kim et al. [5] have presented an analysis of users' ease of use and satisfaction when using search engines with two different control types. Their study compares vertical scrolling, which involves users dragging a finger vertically to imitate holding and dragging a scroll bar on a desktop browser, with horizontal pagination, which involves users dragging a finger horizontally to imitate turning a page when reading a book. This study concludes that pagination increased the likelihood of users finding and examining relevant documents, and decreased the amount of time spent searching for relevant results. If this is correct, it has notable ramifications for the mobile device industry, where increased usability and search speed could cause devices designed to facilitate pagination to be designed. Consequently, it is paramount to verify the conclusions drawn and support either the introduction of pagination or the continued prevalence of vertical scrolling.

1.2 Previous Experiment

In this study, we present a further analysis to confirm the veracity of the conclusions drawn by Kim et al. [5]. If pagination provides significant improvements when compared to scrolling, it will produce significantly different results, so it should be possible to categorise each type separately in an unlabelled dataset with data points corresponding to both scrolling and pagination. To do this, we want to establish that a simple classifier is capable of distinguishing between these control types. This study involves the creation of a two-layer neural network to achieve this classification task, taking an input data point produced in the study performed by Kim et al. [5], and predicting whether it is more likely to be an instance of vertical scrolling or horizontal pagination. It also will consider using an evolutionary learning classifier to solve the same problem.

Assuming these models are effective, an accurate ability to classify the input data supports the idea that the two types produce different results, and allows further comparisons to be drawn about the typical results and benefits of each type. For instance, providing inputs to maximise the activation of each output of the neural network will provide an exaggerated comparison between the types [10], and the rule produced by the evolutionary algorithm allows the decision boundary to be understood and implemented. On the contrary, if it is incapable of accurately classifying the input data, then it is unlikely that the two types produce distinct results, suggesting that there is not an objective benefit of using pagination over scrolling or vice versa.

1.3 Technique Influence

The neural network classifier has been influenced by two techniques suggested by Milne et al. [8] in their paper describing their approach to constructing a network to classify types of forest from geographical data. Firstly, they undertake extensive preprocessing of their data in order to convert it into a meaningful and interpretable form that allows their neural network to learn effectively. Secondly, they produce a variable threshold which allows some control over the ratio of false positives to false negatives. This is particularly effective in situations where false positives and false negatives have unequal consequences, and minimising one or the other is imperative to diminish any negative outcomes. Whilst the consequences of misclassifying scrolling and pagination are not severe, the concept of varying the classification threshold can still be applied in an attempt to increase the accuracy of the neural network.

1.4 Genetic Programming

Genetic programming is a type of evolutionary algorithm that mimics biological evolution in order to find the most appropriate representation to solve a given problem [7]. In this study, genetic programming is used to find a suitable classifier to distinguish between the two control types. This is done by iteratively producing generations of parse trees that represent an expression for the decision boundary, which is evaluated against the data before producing the next generation from the highest-performing expressions. This technique is particularly effective when there is not a known representation of the known solution, which is the case in this problem, and its ability to test a range of conditions is ideal for searching for such an unknown decision boundary [2]. In this study, the genetic programming model will also be considered with and without preprocessing, and considered with and without thresholding.

In summary, this study will conclude whether it is possible to consistently distinguish between the two types, and hence whether the difference between scrolling or pagination in mobile search engines is significant. Crucially, this can be used to inform future design decisions of mobile devices and web browsers, which have wide-reaching impacts on their extensive range of consumers.

2 Methodology

In order to compare the preprocessing and variable thresholding techniques presented by Milne et al. [8], four neural networks were produced. This includes neural networks including and excluding both the preprocessing steps and the variable thresholding steps. Including a neural network with neither preprocessing nor variable thresholding acts as a control for comparing the other neural networks against it and verifying that the preprocessing steps taken are both appropriate and effective. Meanwhile, producing all four permutations of the techniques ensures that the source of any changes in outcomes can be known, with no doubt about which changes in implementation have produced results. Additionally, the production of four neural networks increases the likelihood that one of them will produce appropriate results to distinguish the scrolling and pagination control types, or if it is not possible, that they will corroborate this with each other. Each of the networks produced follow similar two-layer architectures and are trained with the same dataset and the same methodology.

In a similar vein, four genetic programming models were produced, including and excluding both the preprocessing steps and variable thresholding steps. These act to both increase the likelihood of creating a model that can distinguish the control types, as well as to corroborate the absence of a clear decision boundary if none of the models can produce one.

This study selects two-layer neural networks to be designed since they provide a simple mechanism for classification, producing reasonable results with most simple classification problems. Similarly, genetic programming models are investigated to determine the existence of a straightforward, interpretable classification mechanism for the problem. By creating eight models in total, it is likely that any decision boundary of reasonable complexity can be found, and the failure to find one is indicative of its likely non-existence.

2.1 Dataset

The dataset used throughout this report is the same as the dataset used by Kim et al. [5] in their previous study, consisting of the experimental results produced by their methodology. In their experiment, they successfully recorded the time spent on the search engine, time spent on websites, accuracy, and user satisfaction for 24 participants. This was done for six scrolling tests and six pagination tests, for a total of 12 data points for each participant and 288 data points in total. Crucially, each scrolling test has a corresponding pagination test, resulting in a balanced dataset. A description of each of the fields in the dataset is provided in Table 1.

Table 1. Data fields presented in the original dataset, with brief descriptions.

Data Field	Description
Type	Control type used – either V (vertical scrolling) or H (horizontal pagination)
Target Position	Rank of the search result on the search engine
Subject	Number identifying the subject
Time to First Click	Time between task start time and first click time
Time to Right Click	Time between task start time and correct click time
Total Time on SERPs	Total time spent on the search engine
Task Completion Duration	Total time spent to complete the subject's task
Accuracy	Whether the first click was the correct website
Satisfaction	Subject's satisfaction on the control type
Task Start	Raw time that the task started
First Click	Raw time that the first click was performed
Right Click	Raw time that the correct click was performed
Task End	Raw time that the task was completed
Scroll	Whether a scroll occurred before the first click
Time with Wrong WebDoc	Total time spent on incorrect websites

From Table 1, only the "Type" data field is non-numeric, with either an "H" or a "V" denoting horizontal pagination or vertical scrolling respectively. "Accuracy" and "Scroll" are Boolean, with '1' denoting a correct first click or a scroll, and '0' denoting an incorrect first click or the absence of a scroll. "Target Position", "Subject", and "Satisfaction" are each represented on a constrained integer scale, with the former two representing a specific search ranking or subject, and the latter representing the subject's satisfaction as self-nominated with a 7-point Likert scale. The remainder of the data fields contain positive but otherwise unconstrained floating point numbers, representing times or time durations.

The "Type" column of the dataset is used as the target labels, being predicted by the models produced. Meanwhile, the rest of the dataset is used as the input data, which the models are trained and evaluated upon.

It is worth noting that the dataset's statistics provide some insight into the differences between scrolling and pagination. Kim et al. [5] produced results stating that pagination had shorter average time durations for each interaction, as well as a greater correct answer rate on the second page of results (for "Target Position" greater than 5). However, these were not all statistically significant, and none of them had a p-value less than 0.01, indicating that these differences may not always be capable of distinguishing between the two types in a simple manner.

Another brief consideration is that of the individual subjects and their effects on the dataset. Since each subject may take different average amounts of time to complete their tasks, the dataset can be adjusted to prevent the influence of any individual subject on the classification results. Each subject's individual statistics will be considered during the preprocessing stage in order to reduce their influence, in favour of unveiling the underlying trends in the data between control types and target positions.

2.2 Preprocessing

As previously mentioned, two of the four neural networks and genetic models had no preprocessing applied and underwent learning from the entire original dataset. The other two models of each architecture underwent extensive preprocessing on the dataset before training commenced. The steps involved in this preprocessing stage and their justifications for their use are presented in Table 2 [11].

Table 2. Preprocessing steps and justification for each of their inclusions.

Step Taken	Justification
Removing the "Task Start", "First Click", "Right Click", and "Task End" data fields	All information encoded in these fields is retained by "Time to First Click", "Time to Right Click", and "Total Time on SERPs".
Removing the "Task Completion Duration" and "Time with Wrong WebDoc" data fields	The amount of time spent on correct or incorrect websites should have no relevant effect on the amount of time spent on the search engine outside of what is encoded by the "Accuracy" data field.
Normalising the "Target Position" and "Satisfaction" data fields against all other data entries	Converting all values to be between 0 and 1 allows all inputs to have equal effects on the network's learning, and the constrained range of these fields allows standard normalisation to retain their meaning.
Normalising the "Time to First Click" and "Total Time on SERPs" against all of that subjects' other data entries	Normalising the entries for the same subject minimises the effects of the subjects themselves, and retains the relative amount of time taken between target positions and control types.
Subtracting the "Time to First Click" from the "Time to Right Click", then normalising against all of the other data entries for the subject	Taking the difference produces a measure of the time spent before the entry was found whilst reducing the repetition of the data collected.

2.3 Neural Network Model

The neural networks used are each two-layer neural networks trained with the standard backpropagation model. They consist of a linear hidden layer with a sigmoid activation function and a linear output layer that also has a sigmoid activation function, ultimately returning a classification likelihood between 0 and 1. This facilitates the thresholding process by ensuring that the neural network output is within a known range of values and can be modified to accommodate the threshold. Each network has two output neurons, one for each classification category, and has a number of input neurons corresponding to the number of input features in the dataset – 14 for the original dataset, and 7 for the preprocessed dataset. All of the other hyperparameters were decided by iterating through various values of each, and selecting values with the best results for the validation set.

Each network was trained with a number of hidden neurons in the range of 5 to 15, a learning rate between 0.05 and 0.0005, and a number of epochs between 200 and 2000. This was iteratively decided, alternating the hyperparameter being changed until the network had the greatest reported accuracy from the validation set. For the two networks using thresholds, their values were also iteratively decided alongside the other hyperparameters, with values ranging from 0 to 1. These were designed to balance the false positive rate with the false negative rate, which should indicate that the neural network cannot be improved for classification of one control type without being detrimental to the other. Modifications to the threshold were done by subtracting a flat coefficient from the second neural network output. These modifications were done throughout the training, validation, and testing procedures to train and test the network with the threshold incorporated. As an aside, it is appropriate and expected to use different threshold values for the two neural networks designed with them, instead of using the same threshold for both, since they use different datasets and will learn different relationships between neurons.

2.4 Genetic Programming Model

The genetic programming models used each produce a tree representation, with each node being formed from a set of primitive operations, that represents an expression for the decision boundary that distinguishes between the two control types. These are implemented with a strongly typed representation to ensure that all operations are performed with real numbers, with representations produced by this model accepting each of the real-numbered input features in a data point and producing a real-numbered output that is compared with the threshold to produce a classification [9]. The tree representation uses eight primitives: standard addition, subtraction and multiplication; standard division, but providing an outcome of 1 whenever the denominator is 0; negation, sine and cosine functions; and an ephemeral constant, which returns a random value between 0 and 1 inclusive. These are sufficient to produce a range of linear and non-linear expressions to represent the decision boundary – an inability to produce a reasonable decision boundary with this primitive set suggests that either the boundary is exceedingly complex to implement, or that a clear decision boundary does not exist.

Each model undergoes evolution using an initial population of 50 individuals, each of which is a tree initialises with the half-and-half method, with leaves at either depth 1 or depth 2 (and not necessarily at the same depth for each leaf). Their fitnesses are evaluated by taking a random subset with 10% of the training samples and calculating the number of correct classifications produced by the individual's expression [3]. Taking a random sample introduces more

stochasticity and reduces the amount of overfitting to the training data, as well as decreasing the amount of computation required. This evaluation process also includes the threshold being considered, with the real-numbered output value being processed in a sigmoid function and compared to the threshold to determine the control type classification. From these, the model undergoes a selection tournament to select the best third of the individuals, before performing crossover (with a rate between 0.1 and 0.6) to exchange a single subtree between two individuals, and performing mutation (with a rate between 0.01 and 0.4) to replace a subtree with a random subtree produced in the same fashion as the initialisation process. Both crossover and mutation are constrained to a maximum tree size of 90 in order to fit within the requirements of Python's stack limit. Similarly to the neural network model, each of the hyperparameters were decided iteratively until the greatest validation set accuracies were reported.

2.5 Validation and Testing

K-fold cross validation was used for validating and testing both the neural network and genetic programming models, allowing each model to be evaluated equally on a range of data points without a large loss of usable data. This involved separating the dataset into a number of subsamples and setting one set aside to be used as the testing set. The rest of the data was used for training and validation, performing cross-validation by using each of the remaining subsamples in turn as the validation set, training the model with the remainder of the data, and evaluating its performance with the validation set. Cross validation was done with 8 subsamples, chosen to split the dataset into equal groups whilst having a reasonably large number of data points in each group, so validation and testing could produce lower-variance results in each iteration. Once cross validation was completed and each of the models was trained and evaluated against their respective validation sets, the model producing the highest validation accuracy was selected and evaluated against the test set.

For the neural networks, cross entropy loss was selected as the loss function, combining the softmax function with negative log-likelihood loss to produce a score for each class and penalise larger mispredictions to accelerate the learning process. The Adam optimiser function has been selected, performing gradient descent to update the neural network in each epoch whilst being computationally efficient and effective on a range of gradients.

For the genetic programming models, the validation and test set evaluations were done in a similar fashion to the training set evaluations. This occurred by calculating the number of correct classifications performed by the model, and evaluates over the entire validation or test set instead of taking a random sampling from it.

3 Results

3.1 Hyperparameter Selection

The hyperparameter selection process described previously was undertaken to produce four neural network architectures and four genetic programming models, one for each permutation of unpreprocessed against preprocessed, and with a varying threshold against no threshold. This was performed iteratively until the validation loss was minimised. The selected neural network hyperparameters are presented in Table 3, and the selected genetic programming hyperparameters are presented in Table 4.

Table 3. Finalised hyperparameters for each neural network produced.

Neural Network	# Hidden Neurons	# Epochs	Learning Rate	Threshold
Unpreprocessed, no threshold	8	500	0.002	N/A
Preprocessed, no threshold	8	1500	0.001	N/A
Unpreprocessed, threshold	9	1000	0.001	0.60
Preprocessed, threshold	10	1500	0.0015	0.45

Table 4. Finalised hyperparameters for each genetic programming (GP) model produced.

GP Model	Crossover rate	Mutation rate	# Generations	Threshold
Unpreprocessed, no threshold	0.4	0.05	300	N/A
Preprocessed, no threshold	0.4	0.05	180	N/A
Unpreprocessed, threshold	0.4	0.05	300	0.60
Preprocessed, threshold	0.4	0.1	180	0.40

The neural network hyperparameter selection in Table 3 indicate that the number of hidden neurons optimising the network's accuracy remains similar, despite the additional threshold hyperparameter and despite the changes to the dataset during the preprocessing stage. However, the greater number of epochs required for the preprocessed dataset and the decreased learning rate compared to the unpreprocessed dataset with no thresholding reflects the effects of normalising the data, with the network being trained to determine the input features with the greatest effect on producing an accurate classification.

The genetic programming hyperparameter selection in Table 4 suggests that the number of generations required is lower for the preprocessed model than for the unpreprocessed model. This reflects the simplicity of input features provided by the preprocessed model, requiring less training to produce adequate results, and with subsequent training causing the model to undergo overfitting. The crossover rate and mutation rate remain similar between the preprocessing and thresholds provided, indicating that the creation of individuals occurs in a similar manner for each model.

One design decision was to ensure that the varying thresholds were not set to 0.5, regardless of its accuracy – a threshold of 0.5 is equivalent to not using a threshold, and this acts against the purpose of this paper to judge the efficacy of the thresholding technique.

3.2 Classification Accuracy

Once the hyperparameters had been decided, each of the models was trained, validated, and tested 20 times to produce a set of classification accuracy statistics. The mean, maximum, and minimum classification accuracies produced by each of the models is reported in Table 5.

Table 5. Mean, maximum, and minimum average statistics for each neural network (NN) and genetic programming (GP) model.

Model	Mean accuracy	Max. accuracy	Min. accuracy
Unpreprocessed NN, no threshold	56.39 %	66.67 %	38.89 %
Preprocessed NN, no threshold	59.72 %	66.67 %	47.22 %
Unpreprocessed NN, threshold	56.11 %	63.89 %	50.00 %
Preprocessed NN, threshold	53.33 %	55.56 %	52.78 %
Unpreprocessed GP, no threshold	45.41%	58.33%	41.67%
Preprocessed GP, no threshold	52.78%	66.67%	30.56%
Unpreprocessed GP, threshold	51.11%	58.33%	38.89%
Preprocessed GP, threshold	52.92%	61.11%	44.44%

4 Discussion

4.1 Classification Efficacy

The results provided in Table 5 indicate that all of the two-layer neural networks and the genetic programming models struggle to produce a model that can classify more than two-thirds of the data correctly, regardless of the different architectures produced. This is a reflection of the lack of distinction between the results produced by using the scrolling and pagination control types.

Crucially, the presence of both false positives and false negatives in roughly equal proportions in the trained neural networks' results, as the networks were designed to achieve, is indicative of the crossover in statistics between the two control types, with both types being misclassified as the other. Models that classify all scrolling and some pagination instances as scrolling, or all pagination and some scrolling instances as pagination, are effective in establishing some criteria that can classify data points into one type or the other. Conversely, they can help to disprove that a data point is likely to be produced with a certain control type. Of course, a model with minimal numbers of both false positives and false negatives is ideal in correctly classifying the control types. However, since none of these statements are applicable in this problem, none of the models designed are greatly effective at the classification task.

Furthermore, the decision boundaries containing many terms produced by the genetic programming models indicate that an effective decision boundary is unlikely to exist using this primitive set in this problem. Since the evolutionary approach taken can produce a linear or non-linear classifier of almost arbitrary complexity from the operators in the primitive set (only constrained by the maximum tree size), this supports the conclusion that a decision boundary does not exist and the two control types are not sufficiently different to be distinguishable.

4.2 Comparisons Between Models and with Previous Research

In terms of the mean results, the neural network models perform slightly better than the genetic programming models for each of the permutations of preprocessing and thresholding. With the lack of interpretability from the neural networks, it is difficult to precisely identify the reasoning for their higher performance. One possible reason for this difference is the complexity of the problem causing a solution to require significant interconnection of the inputs, which the neural network model does by design whilst the genetic programming approach requires the tree model to explicitly incorporate each of the inputs together in a meaningful way [2]. Another potential reason is the need for primitive operations and coefficients outside of the evolutionary models' primitive sets to be applied, limiting the efficacy of these models to be the closest approximation of these functions, whereas the neural network would be able to determine these exactly [7].

One benefit of the genetic programming model, however, is its interpretability. Whilst the neural network model takes a black box approach, where the outputs are produced from the inputs in a way that is difficult to interpret by a reader, the genetic programming model produces individuals that represent expressions which can be retrieved from their parse trees [3]. This allows an audience to determine the key features for classifying the data samples into each control type, allowing them to understand why any particular classification decision occurred. Furthermore, this study could be extended to monitor the evolution of the most effective individuals during the training period, or to develop further representations of the decision boundary that may produce more accurate classification results.

When comparing the neural networks that use the preprocessed data against the networks that use the unmodified dataset, the network that has been trained with the preprocessed data and no threshold – or equivalently, a threshold of 0.5 – has the greatest classification accuracy on average and at a maximum. Similarly, the genetic programming models that use the preprocessed data have a greater mean classification accuracy than those using the unpreprocessed data. This suggests that the preprocessing steps included do indeed increase the efficacy of the model, causing more data points to be classified correctly. This success comes from two mechanisms: removal of superfluous data fields, and extraction of meaning from data points.

By removing the fields whose meaning is not relevant to the classification outcome, such as the "Task Completion Duration" field, the models reduce the influence of noise during training and provide more scope for the neurons (in neural networks) or individuals (in genetic programming) to be trained on meaningful data. Furthermore, reducing the number of input features during preprocessing allows fewer neural network connections to be calculated each pass, increasing the computational efficiency and allowing more epochs or more iterations to be considered within the same timeframe. For genetic programming, reducing the number of input features restricts the number of potential nodes that can be included in the parse trees, reducing the number of generations required before convergence occurs.

Extracting meaningful information from the dataset is also key to producing meaningful results. For instance, normalising the data fields corresponding to time durations can cause the entire data field to be distorted if a single outlier affects the normalisation process. By normalising the data fields corresponding to time durations for each subject, the effects of differences between subjects, such as variations in their search engine experience or tendencies to be fully informed before making a click, are minimised. This allows the general usage of scrolling and pagination in comparison to each other to be explicitly learnt.

This is reminiscent of the encoding techniques used by Milne et al. [8], converting the data into a form that clarifies the information being provided. This enables the neural network to undergo learning with the most meaningful information and removing the impact of noise, producing more accurate outputs in less time.

There are mixed results from comparing the models using thresholds against those without thresholds. Both of the neural networks trained with thresholds had lower maximum accuracies and greater minimum accuracies than their corresponding networks trained without thresholds. This suggests that designing the thresholds to balance the ratio of false positives and false negatives reduces the variance of the classification outcomes, with fewer outstandingly good or bad classifier models. This makes sense intuitively – the number of data points belonging to each class in the training and validation sets has less influence over the outcome when the ratios are balanced.

Both of the genetic programming models trained with thresholds had slightly higher mean accuracies than their counterparts without thresholds being applied. This is somewhat surprising – since the representations are fully learnt by the evolutionary approach, it would make sense if the threshold did not affect the classification results. However, the difference in threshold may have some effect on the parse tree formulation, with some thresholds being more suitable to the limited expressions that are able to be produced by the evolutionary models. To test whether this is the case, this study could be extended to include learnt coefficients for each of the nodes, reducing any bias from differing numerical values being produced for each expression and allowing the relationships between operations and input features to be fully learnt.

Interestingly, the preprocessed network with thresholding has lower average and maximum accuracies in comparison to the network without thresholding being applied. This suggests that the accuracy is maximised for the preprocessed network when the threshold is set to 0.5, and the classification outputs remain unmodified. From this, we can gather that thresholding is not effective in increasing the classification accuracy with this preprocessed data. One possible reason for this is the significant statistical similarities between the scrolling and pagination data entries, causing it to be difficult to define a threshold that separates them. Furthermore, since it is more imperative to balance the number of false positives and false negatives rather than reducing one in favour of the other, the alternative use of thresholding to reduce the number of false positives or the number of false negatives is not necessary in this usage case.

The final relevant comparison is between the results produced by these models and the results stated in the paper written by Kim et al. [5], where they conclude that the horizontal pagination control type has advantages over the vertical scrolling type. The models produced in this study cannot produce results indicating that one control type is better than the other, or even significantly distinguishing the two groups with a likelihood much better than random choice. This is contradictory to the paper's conclusions; in particular, it states that participants using pagination had several distinguishing outcomes from those using scrolling, such as being "more likely to find relevant documents", spending "more time attending to relevant results", and being "faster to click" [5]. It is intuitive that, if these statements were significantly true, the classifiers could use these heuristics to distinguish the two control types. The classifiers' inability to distinguish these control types suggests that the conclusion drawn in their study does not have a strong basis, and further examination would be required to draw a conclusion either for or against the use of horizontal pagination.

4.3 Limitations and Future Work

There are a number of limitations in the work presented in this study. Firstly, the rigid use of a two-layer neural network and the use of evolutionary models with a defined set of operators restricts the production of arbitrarily complex classifiers. Whilst this also limits the amount of overfitting that is possible, it may limit the models' ability to perform classification on the overlapping datasets without producing many misclassified points. Secondly, the limited amount of training data is restrictive to the ability to train the neural network. Even with 288 data points, there are only 24 data points for each control type and target position combination for the neural network to learn with. This also increases the influence of outliers or trends in the data that may not truly exist with larger amounts of accurate data, such as variations in the correct click rate. Although it may sound feasible to artificially produce more data points by adding random noise to the existing samples, the low variance between the two control types suggests that this may introduce too much bias to produce meaningful results. Finally, despite the inclusion of genetic programming techniques, the problem of interpretability still remains to an extent [3]. While it is possible to extract the expressions produced during the learning process, it is difficult to understand why these have been selected or understand the meaningful relationships between the input features. This makes it challenging to extend this learning process to manually produce a more effective decision boundary from these expressions.

In order to remedy some of these limitations, several adaptations to the methodology presented in this paper are possible. Firstly, the use of deep neural networks with a greater number of layers can increase the complexity of the classification boundaries in order to produce arbitrary complex decision regions, potentially increasing the accuracy of classifications. This could be particularly effective by applying a convolutional neural network to an extended dataset from the experiment, such as time-series data including when each of the users' actions occurred, so that the network could more effectively equate the resultant data with the control types that caused them. Alternatively, the use of other encoding techniques to produce classification outputs could be considered, such as using fuzzy logic to classify the outputs into a larger range of classes, such as likelihood to be an example of each of the control types. This would

provide further efficacy from the thresholding technique, which is designed to produce results for multi-class classification [6]. The last alternative to increase the decision boundary complexity would be to add more primitive operators to the genetic programming models, although this is unlikely to incorporate as much additional complexity as creating a separate model.

Secondly, an increased amount of data points would be beneficial to reduce the effects of noise and outliers throughout the dataset. This could be achieved by replicating the original experiment with more subjects, corroborating any trends that may already exist and reducing the influence of each individual subject's data. Again, another alternative would be to extend the dataset by recording and analysing time-series data, such as when each action occurred.

Thirdly, another method of classification such as machine learning or statistical analysis could be used to produce interpretable results, similar to those produced by the evolutionary algorithms, as well as being fully replicable [1]. This could also be done by manually extending the expressions produced from the genetic programming models to create a fully-defined algorithm to improve the classification accuracy. Since the models considered in this study produce interpretable expressions that relate the input features and the classifications, it may be possible to create a problem-specific expression to improve upon these classifications. Producing an interpretable classification model would be important if a transparent model was required for an audience to confirm the efficacy and outcomes of the process, for instance when performing critical decision-making tasks.

Finally, further outcomes from the paper written by Kim et al. [5] could be analysed. This could include an analysis of the typical results of each control type by designing input features to the neural networks that maximise the output of each classification output, or designing new models to draw other conclusions from the dataset and determine the veracity of their other hypotheses.

5 Conclusion

This paper has discussed the capacity of neural networks and genetic algorithms to classify data points from two overlapping decision regions, using data from an experiment comparing vertical scrolling and horizontal pagination in a mobile search engine [5]. This concluded with four neural networks and four genetic programming models being designed, comparing the efficacy of preprocessing input data and producing variable thresholds using techniques suggested by Milne et al. [8]. The preprocessing stage was successfully used to improve the classification accuracy of both approaches by minimising the amount of superfluous data and providing dataset-specific techniques to extract meaningful information. The thresholding stage reduced the variation between the reported accuracies of the neural network, and provided an increase to the average classification accuracy for the genetic models (but not the neural network models).

The models produced in this paper were not successful in distinguishing the data points retrieved from the scrolling and pagination tests, with only a 66.7% maximum accuracy and 59.7% average accuracy. This suggests that the conclusions drawn by Kim et al. [5] are not statistically significant, an outcome supported by the statistical analysis on the data that they have provided. However, this is impacted by the limitations of the models designed in this report, such as the inflexibility of the models being produced and the limited amount of data being available from this experiment.

Future work has also been recommended, including producing a deep neural network architecture with increased complexity, producing more data for the models to undergo learning with, and extending the interpretability of the genetic programming models to provide an effective and transparent tool to perform classification. These can all be useful in optimising and justifying the use or non-use of pagination in mobile search engines, which may be a critical consideration for developers involved in the current abundance of mobile devices in the market.

6 References

1. Du, M. et al.: Techniques for interpretable machine learning. Communications of the ACM. (2019).
2. Espejo, P. et al.: A Survey on the Application of Genetic Programming to Classification. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 40, 2, 121-144 (2010).
3. Gathercole, C., Ross, P.: Dynamic training subset selection for supervised learning in Genetic Programming. Parallel Problem Solving from Nature — PPSN III. 312-321 (1994).
4. Household use of information technology, 2016-17 financial year, https://www.abs.gov.au/statistics/industry/technology-and-innovation/household-use-information-technology/latest-release.
5. Kim, J. et al.: Pagination versus Scrolling in Mobile Web Search. CIKM'16. ACM (2016).
6. Lin, C., Lee, C.: Neural-network-based fuzzy logic control and decision system. IEEE Transactions on Computers. 40, 12, (1991).
7. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). (2001).
8. Milne, L. et al.: Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood. Proceedings Australian Conference on Neural Networks. (1995).
9. Montana, D.: Strongly Typed Genetic Programming. Evolutionary Computation. 3, 2, 199-230 (1995).
10. Nguyen, A. et al.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. 29th Conference on Neural Information Processing Systems. (2016).
11. Rafiq, M. et al.: Neural network design for engineering applications. Computers & Structures. 79, 17, (2001).