Effects of reduction by distinctiveness on the performance of search strategy prediction long short-term memory (LSTM) network

Yuanshan Chen

Research School of Computer Science, Australian National University
u6704997@anu.edu.au

Abstract. People use search strategies when searching for useful information with search engines on computers and mobile phones. This search procedure of search engine users is highly relevant to time series. Also, predicting user's search strategy can be regarded as a classification problem in machine learning. Combining the perspectives mentioned above, this classification task can be performed by training a long short-term memory (LSTM) network with a fully connected linear layer added to its output. The network reduction technique has been proved effective in reducing the size of a well-trained neural network, with its performance close to original. This network reduction technique is based on distinctiveness. In order to show the impact of reduction technique on the performance of an LSTM network, the comparison of the training accuracies, test accuracies, and confusion matrices before and after network reduction are adopted as performance measurements. It is shown that for an LSTM network not well-trained, network reduction technique can reduce the size of the network. However, for a well-trained LSTM network, network reduction technique cannot reduce the size of the network. The network will stay the same before and after reduction.

Keywords: Search strategy, long short-term memory (LSTM) network, Network reduction technique, Distinctiveness, Test accuracy, Confusion matrix

1 Introduction

The Internet has become an important medium for people to look up and exchange information. People use computers and mobile phones to search for useful and high-quality information on the Internet to provide guidance to their work, study and decision-making. The most common tool for searching information on the Internet is the search engine [3][4][5][6]. In order to obtain the required information from the search engine efficiently, users will naturally use certain search strategies instead of blindly browsing the search results page. Meanwhile, since browsing all the search results at the same time is not practical, the search behaviour automatically complies with time series.

In order to optimize the search engine design, it is very important to understand user's search strategy. This requires recognizing user's search strategy first, which can be regarded as a classification problem in machine learning. Moreover, since long short-term memory (LSTM) network [7][8] specializes in dealing with time series data, classification tasks can be performed by a well-trained LSTM neural network. But as Gedeon and Harris mentioned in their 1991 paper "Network Reduction Techniques" [2], there may be redundant neurons in these trained neural networks. It is necessary to remove these redundant neurons to reduce the network scale.

The network reduction technique is based on distinctiveness. Distinctiveness is measured by the angle between the vectors whose elements are the outputs of hidden neurons. This network reduction by distinctiveness has been proved to be effective in reducing the size of the network, with the performance of the reduced network close to the original.

I choose the eyegaze-search1 dataset [1] for web search strategy prediction. The neural network I designed consists of one input layer, one output layer and one LSTM layer. I then apply network reduction by distinctiveness to this trained neural network [2]. In order to show the impact of reduction technique on network performance, test accuracy and confusion matrices are adopted.

The structure of this paper is organized as follows. Section 2 describes the dataset, the reduction technique and other methods I adopted. Section 3 demonstrates the experimental results and discussions. Section 4 states the conclusions and the direction to future work.

2 Methods

The task of predicting the search strategy of participants on eyegaze-search1 dataset can be modelled as a classification problem in machine learning. The details about the original dataset itself is described in section 2.1. Data pre-processing for the input data and the training targets is described in section 2.2. The LSTM mechanism and its representation in Pytorch [9] is described in section 2.3. Distinctiveness and its application in LSTM network reduction is described in
The classification method for search strategy prediction task is described in section 2.5. The measurement for guiding the training process of the neural network is described in section 2.6. The evaluation method for the impact of the reduction technique on the performance of the neural network is described in section 2.7.

2.1 Dataset Description

In this paper, I choose the eyegaze-search1 dataset [1] for search strategy prediction. This dataset was produced by recording the eye movement, as well as the scrolling and clicking behavior of the participants while they were performing given web search tasks using search engines. I choose this dataset for two main reasons.

First, the design for the experiment that generate this dataset covers the main scenarios in which people may use a web search engine. For instance, people may search on the large screen of a computer, or search on a small screen of a mobile phone. Meanwhile, the experiment design for eyegaze-search1 dataset considers the two common possible types of search tasks people need to finish when they seek help from web search engines. The tasks fall into two categories of reaching a specific web page and finding the answer to the specific question. These two categories are named navigational and informational, respectively.

Second, the measurements used when collecting data – such as search speed (measured by the time passed before the first click and the total time taken to finish the task), fixation duration (measured by the time users spend gazing at a particular area on the screen), scanpath (indicating the sequence of fixations of the user on the search result page), maximum gaze position (showing the region on the search result page to which the user pays the most attention) – effectively reflect people's eye behavior when using the web search engine, as well as people's clicking and scrolling behavior. Therefore, this dataset forms excellent foundation for predicting web search strategy.

Most importantly, the dataset has recorded the time users spent on every area of interest (AOI) on each search results web pages. Each of these time numeric values construct a time series together. The relationship between the time users spent gazing at each region is a good indicator for this user’s search strategy on this specific task.

The entire dataset consists of 640 samples, collected from 32 participants, each completing 20 search tasks. 320 search tasks were performed on the big screen, and the other 320 were performed on the small screen. 160 informational tasks and 160 navigational tasks were performed on the large screen and on the small screen, respectively. Among all 640 samples, 247 samples fall in the category of depth-first (DF) search strategy, 362 samples fall in the category of mixed (MX) search strategy, and 31 samples fall in the category of breadth-first (BF) strategy.

2.2 Data Pre-processing

To begin with, it can be shown from the observation that different participants use different search strategies based on different search tasks, indicating the search strategy has little to do with specific participants. So the column indicating the participant is removed.

Most of the features in the eyegaze-search1 dataset are numerical, but still some are not. These include the size of the screen and the type of search task, as well as the search strategy. It is necessary to convert these features into a form that can be represented numerically since neural networks only accept numerical inputs.

In order to run the classification task on the search strategy, which will be the function of the designed neural network, each item in the search strategy column also needs to be converted to a numeric type. The depth-first (DF) search strategy will be represented by 0. The mixed (MX) search strategy will be represented by 1. The breadth-first (BF) search strategy will be represented by 2.

Since the hidden neurons in the network designed for the search strategy classification task use the tanh activation function (The reason will be shown in Sec 2.3 and Sec 2.4), and since all numerical values are non-negative, each dimension of the input data is better normalized to range 0 to 1 in order to avoid the gradient vanishing circumstance. The time spent on gazing each area of interest (AOI) to complete the task don’t differ much among different members. Therefore, it will be adequate to use the min-max normalization. Min-max normalization first subtracts the minimum value in the column from the original value, then divides the result by the difference between the maximum value and the minimum value in the column. Other columns whose range is not 0 to 1 are normalized by min-max normalization.

Most importantly, LSTMs in Pytorch accept three dimensional tensors as the input only. Therefore, the original data must be re-organized to a tensor of specific shape.

The first dimension of this tensor is the length of the sequence. Since every web search result page has 10 areas of interests (AOIs), the length of the first dimension should be 10.

The second dimension of this tensor is the size of the batch. Since there are 32 participants, each performing 20 search tasks, the length of the second dimension is the number of training samples or the number of test samples.
Moreover, because of this batch specification, only when the original dataset is split in half can both the training set and the test set function properly in this network, otherwise batch size problem will occur. Therefore, the second dimension of the training set and the second dimension of the test set are both 320.

The third dimension of this tensor is the length of each of the elements in the sequence. Although there are 10 AOIs for each task, the participant can only gaze at 1 AOI in each time step. This makes the length of the third dimension be 1.

2.3 LSTM mechanism and its representation in Pytorch

An LSTM [7][8] consists of four parts. The first part is the forget gate. The two matrices corresponding to this part are denoted as \( W_f \) and \( W_{hf} \), respectively. The formula of the first part is shown in equation (1), where \( \sigma \) denotes the sigmoid function.

\[
f_t = \sigma(W_{fi} x_t + b_{fi} + W_{hf} h_{t-1} + b_{hf})
\]

(1)

The second part is the input gate. The two matrices corresponding to this part are denoted as \( W_i \) and \( W_{hi} \), respectively. The formula of the second part is shown in equation (2), where \( \sigma \) denotes the sigmoid function.

\[
i_t = \sigma(W_{ii} x_t + b_{ii} + W_{hi} h_{t-1} + b_{hi})
\]

(2)

The third part is the tanh layer. The two matrices corresponding to this part are denoted as \( W_g \) and \( W_{hg} \), respectively. The formula of the third part is shown in equation (3).

\[
g_t = \tanh(W_{gi} x_t + b_{gi} + W_{hg} h_{t-1} + b_{hg})
\]

(3)

The fourth part is the output gate. The output from this gate is passed on to the next layer as well as the next time step. The two matrices corresponding to this part are denoted as \( W_o \) and \( W_{ho} \), respectively. The formula of the third part is shown in equation (4), where \( \sigma \) denotes the sigmoid function.

\[
o_t = \sigma(W_{io} x_t + b_{io} + W_{ho} h_{t-1} + b_{ho})
\]

(4)

In Pytorch [9], \( W_i, W_f, W_g, \) and \( W_o \) are stored sequentially in the built-in variable named weight_ih_l0 for a one-layer LSTM network; \( W_{hi}, W_{hi}, W_{hg}, \) and \( W_{ho} \) are stored sequentially in the built-in variable named weight_hh_l0 for this one-layer LSTM network. This storage mechanism is fundamental to reduction on LSTM networks.

2.4 Distinctiveness and LSTM Network Reduction

The distinctiveness of hidden neurons is measured by the magnitude of the angles between vectors, whose elements are the output of the hidden neurons from each training pattern. As described in the original technique paper, since the activation function used in every hidden neuron is tanh, the output value of each hidden neuron lies between -1 and 1. Therefore, the angle between vectors fall between 0 and 180 degrees automatically, no further normalisation is needed.

The network reduction technique mentioned in the original technique paper is based on the distinctiveness mentioned above. First, the angle between the vectors corresponding to each hidden neuron is calculated. Then, all these pairs of vectors will be considered. When the angle between the two vectors corresponding to the two hidden neurons is less than 15 degrees, one of these two hidden neurons will be deleted, with its corresponding weights added to the corresponding weights of the other one which remains. When the angle is greater than 165 degrees, both these two hidden neurons will be deleted since there is no idea which neuron is leading the classification task to the right way.

The mechanisms for applying this reduction technique to an LSTM network are shown in Figure 1 and Figure 2.
In order to perform classification, the elements to be reduced from the weight matrices are circled in red. For weight matrices \( W_{hi}, W_{hi}, W_{hp}, \) and \( W_{ho} \), the vectors indexed 2 in terms of the second dimension need to be removed. The reason for this is that the definition of an LSTM requires \( W_{hi}, W_{hi}, W_{hp}, \) and \( W_{ho} \) to be square matrices. In actual implementation of this instance, the corresponding rows of weight_ih_l0 will be removed, and the corresponding rows and columns of weight_hh_l0 will be removed.

\[
\begin{align*}
\mathbf{h}_{(t)} &\sim \mathbf{x}_{(t)} \mathbf{W} + \mathbf{h}_{(t-1)}
\end{align*}
\]

**Fig. 1.** Performing reduction to the LSTM layer. \( W_{(i)} \) denotes \( W_{hi}, W_{hi}, W_{hp}, \) and \( W_{ho} \). \( W_{(h)} \) denotes \( W_{hi}, W_{hi}, W_{hp}, \) and \( W_{ho} \). \( x_{(t)} \) denotes input vectors.

Figure 2 shows the mechanism for performing reduction in the linear layer. For instance, in Figure 2, when the second element in the output is to be reduced, the elements to be removed are circled in red. For weight matrices \( W, \) the vectors indexed 2 in terms of the second dimension need to be removed. The reason for this is that the definition of an LSTM requires \( W_{hi}, W_{hi}, W_{hp}, \) and \( W_{ho} \) to be square matrices. In actual implementation of this instance, the corresponding rows of weight_ih_l0 will be removed, and the corresponding rows and columns of weight_hh_l0 will be removed.

\[
\begin{align*}
\mathbf{h}_{(t)} &\sim \mathbf{h}_{(t)} \mathbf{W}
\end{align*}
\]

**Fig. 2.** Performing reduction to the linear layer. \( h_{(t)} \) denotes the output from the LSTM layer, and \( W \) denotes the weight matrix for the linear layer.

Figure 2 shows the mechanism for performing reduction in the linear layer. For instance, in Figure 2, when the second element in the output is to be reduced, the elements to be reduced from the weight matrices are circled in red. For the weight matrices \( W, \) the vectors indexed 2 in terms of the first dimension need to be removed, so that the output from the LSTM layer will match in size with the weight matrix of the linear layer.

### 2.5 Classification Method

Since there are three search strategies in the dataset, the task for the neural network can be modelled as a three-category classification problem. Therefore, the number of neurons in the output layer will be 3. Also, since the task is to classify, the activation function chosen for the output layer is softmax. The formulae of the softmax function are shown in (5), (6) and (7).

\[
x = (x_1, x_2, x_3, \ldots, x_k)
\]

(5)

\[
s = \text{softmax}(x)
\]

(6)

\[
s_i = \frac{e^{x_i}}{\sum_{i=1}^{k} e^{x_i}}
\]

(7)

The function of the softmax function is to convert the original input vector into a vector with all the elements summing to 1. So that this produced vector can indicate the probability for the search strategy of each sample to be depth-first (DF), mixed (MX), or breadth-first (BF).

### 2.6 Neural Network Performance Measurement

In order to train the neural network to perform the designated task, a measurement must be introduced to guide the improvement of the network performance during training. This measurement is often called loss function in machine learning context. For this three-category search strategy classification task, softmax function is first applied to the output from the output layer. Cross-entropy loss function is then used as this guide to performance improvement. The formula for cross entropy loss is as shown in equation (8).
In this context, \( s \) is the output vector of the softmax function, and \( y \) is the vector indicating the true label of true sample corresponding to this output vector. For example, if the search strategy of the sample used for producing this particular vector \( s \) is depth-first (DF) search strategy, then the corresponding \( y \) will be \([1, 0, 0]\). If the search strategy is mixed (MX), the corresponding \( y \) will be \([0, 1, 0]\).

The reason that the combination of softmax function with the cross-entropy loss function is optimal is as follows. When we apply the chain rule, the derivative of the loss function with respect to the output vector is exactly equal to the vector obtained from using softmax function on the output layer minus the vector corresponding to the target classification value. Namely, \( s - y \), where \( s \) and \( y \) are the same as what mentioned in the previous paragraph. The calculation is shown in (9), (10), (11) and (12), which are basically simple derivatives.

\[
\nabla e(s) = \left( \frac{-y_1}{s_1}, \frac{-y_2}{s_2}, \ldots, \frac{-y_k}{s_k} \right)
\]

\[
\nabla s(x) = \begin{pmatrix}
-s_1 s_1 & -s_1 s_2 & \ldots & -s_1 s_k \\
-s_2 s_1 & -s_2 s_2 & \ldots & -s_2 s_k \\
\vdots & \vdots & \ddots & \vdots \\
-s_k s_1 & -s_k s_2 & \ldots & -s_k s_k
\end{pmatrix}
\]

\[
\nabla e(x) = \left( s_1 \sum_{i=1}^{k} y_i - y_1, s_2 \sum_{i=1}^{k} y_i - y_2, \ldots, s_k \sum_{i=1}^{k} y_i - y_k \right)
\]

\[
\frac{\partial e}{\partial x_i} = s_i \sum_{i=1}^{k} y_i - y_i
\]

Specifically, in this search strategy classification context, the only three possibilities of \( y \) are \([1, 0, 0]\), \([0, 1, 0]\) and \([0, 0, 1]\), which all satisfy formula (13).

\[
\sum_{i=1}^{k} y_i = 1
\]

Combining (9) with (8) gives the result of the derivative of the cross-entropy loss with respect to the neural network output to be \( s - y \), as mentioned above. Therefore, the combination of softmax function with the cross-entropy loss function can greatly reduce the total computation amount.

### 2.7 Reduction Technique Evaluation Measurement

In order to show the impact of reduction technique on network performance, test accuracy and confusion matrices are adopted.

Test accuracy means the ratio of the number of correct predictions in the test dataset to the total number of samples in the test dataset. In multi-classification problems, this ratio is an important indicator when measuring the performance of the classification network.

The purpose for using the confusion matrices for evaluation of the reduction technique is to study how network reduction affects the actual classification result. The way to recognize this effect is to compare each element in the confusion matrices before and after reduction. The row index of the confusion matrices refers to the actual search strategy category to which each sample belongs. The column index of the confusion matrices refers to the classification outcome predicted by the network. This means that the numbers on the diagonal of confusion matrices are the number of samples that are correctly classified into the corresponding category, respectively.
3 Results and Discussion

The task of this paper is first to perform network reduction by distinctiveness on the trained LSTM network, then to check the performance difference between the network before reduction and the network after reduction. The network cannot be re-trained after reduction.

Since no prediction or classification task is mentioned in the eyegaze-search1 dataset paper, I will compare the classification performance of the well-trained LSTM network before and after network reduction.

The discussion will focus on how network reduction affects the overall performance of the resulting classification neural network. For this discussion, I designed an LSTM network with a linear layer connected to its output. The number of input nodes is 10, given that there are 10 AOIs for each task. The number of hidden nodes in the LSTM layer is 100, and the numbers of nodes in the output layer is 3, the same as the number of classes.

After finishing the reduction on the LSTM layer and its subsequent linear layer, a confusion matrix for the training data will be generated. I will evaluate the impact of reduction technique on the specific classification process of the network by comparing the training data confusion matrix of the original network with the training data confusion matrix of the reduced network.

Also, I recorded the test accuracy and the test data confusion matrix of the neural network before and after forward reduction and backward reduction, respectively. By comparing the test accuracy and confusion matrices, the influence of reduction order on network classification performance can be evaluated.

The stopping criterion of training is that the cross-entropy loss reaches below $2 \times 10^{-3}$. The training loss by training epoch is shown is Figure 3.

![Fig. 3. The cross-entropy loss after each training epoch.](image)

The training and test accuracy of the neural network before and after reduction are shown in Tables 1 and 2, respectively.

<table>
<thead>
<tr>
<th>Table 1. Training accuracy before and after reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>before reduction</td>
</tr>
<tr>
<td>after reduction</td>
</tr>
</tbody>
</table>
According to Table 1, in terms of training, the training accuracy before network reduction was 100.00%, while the training accuracy after network reduction was 100.00%. After reduction, the test accuracy has not changed.

According to Table 2, in terms of testing, the test accuracy before network reduction is 67.50%, and the test accuracy after reduction is 67.50%. The accuracy before and after reduction also remains unchanged.

The training data confusion matrices before and after performing reduction on the LSTM network are shown in Table 3, and the test data confusion matrices are shown in Table 4.

In Table 3 and Table 4, the row index of the confusion matrices refers to the actual search strategy category to which each sample belongs. The column index of the confusion matrices refers to the classification outcome predicted by the network.

In Table 3 and Table 4, the training data confusion matrices and the test data confusion matrices before and after reduction are compared. There is no difference between the confusion matrices before and after performing reduction.

The reason for the results presented above is as follows. By recording the number of hidden nodes in the network after performing reduction, I discovered that for a fully trained LSTM network, reduction cannot be performed at all. All the angles between the hidden notes fall between 15 degrees and 165 degrees, making them all distinct. Only when this network is not fully trained can reduction be performed. This indicates that for small datasets, deep learning approaches is very prone to overfitting.

### 4 Conclusions and future work

In this paper, I choose the eyegaze-search1 dataset [1] for search strategy prediction. The LSTM network I designed consists of one LSTM layer, followed by one fully connected linear layer. I generalised the network reduction technique [2] to the LSTM network. In order to show the impact of reduction technique on network performance, training accuracy, test accuracy and corresponding confusion matrices are adopted. It shows that from the perspective of test accuracy and confusion matrices that for a fully trained LSTM network, reduction cannot be performed at all, since all hidden nodes are distinct from others. Only when this network is not fully trained can reduction be performed, which indicates that for small datasets, deep learning approaches is very prone to overfitting.

In future work, I will try to modify the architecture of the network with which I use to perform the classification task. I may apply cross-validation. I may apply 2 or 3 LSTM layers, or I may apply much bigger datasets of time series to examine the validity of reduction technique.

<table>
<thead>
<tr>
<th>Class</th>
<th>Before reduction</th>
<th>After reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=DF</td>
<td>129 0 0 129</td>
<td>Class=DF 129 0 0 129</td>
</tr>
<tr>
<td>Class=MX</td>
<td>0 183 0 183</td>
<td>Class=MX 0 183 0 183</td>
</tr>
<tr>
<td>Class=BF</td>
<td>0 0 8 8</td>
<td>Class=BF 0 0 8 8</td>
</tr>
<tr>
<td>Total</td>
<td>129 183 8 320</td>
<td>Total 129 183 8 320</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>Before reduction</th>
<th>After reducing the last hidden layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=DF</td>
<td>65 50 3 118</td>
<td>Class=DF 65 50 3 118</td>
</tr>
<tr>
<td>Class=MX</td>
<td>34 143 2 179</td>
<td>Class=MX 34 143 2 179</td>
</tr>
<tr>
<td>Class=BF</td>
<td>2 13 8 23</td>
<td>Class=BF 2 13 8 23</td>
</tr>
<tr>
<td>Total</td>
<td>101 206 13 320</td>
<td>Total 101 206 13 320</td>
</tr>
</tbody>
</table>
References


