Casper-based Neural Network Modelling with Fuzzy Logic Data Pre-processing for Classification of User Search Task Snippet length with Timing-related User Search Behavior Attributes

Ziwei Liu

College of Engineering and Computer Science, Australian National University
u6380075@anu.edu.au

Abstract. Based on previously published research, there has been significant differences shown between the amounts of screen fixation time of users who are shown different lengths of internet search result snippets. Timing-related data of user search behavior may be used to construct a model to predict their assigned snippet lengths. The model chosen was the Casper model, which is a modification on the cascade-correlation neural network architecture. Numerous further modifications were made on the Casper model such as the employment of mini-batch gradient descent with the RMSProp algorithm, and different metrics for determining neuron addition times. Fuzzy logic pre-processing was also performed on the data. Fuzzy membership functions were devised by splitting the data based on the labels and analyzing the features' distributions for each label. All features were then fuzzified, and their unfuzzified versions were removed. The Casper-based model was compared to a standard 2-layer neural network model. Both models performed relatively poorly, with accuracies of 0.38 and 0.35 on testing respectively. It was shown that the Casper-based model underperformed compared to the standard neural network model, and that the fuzzy logic pre-processing provided little to no benefit to the model. It was concluded that due to the better performance and the simplicity of implementation, the standard neural network model was the better choice for the classification task.

1 INTRODUCTION

1.1 Background and motivation

Searching for resources is one of the most common usages of the internet today, with studies revealing it to be the second most popular online activity [1]. Search engines on the internet display on their search engine result pages (SERPs) several different forms of representation of the resulting webpages. Most commonly, websites are displayed with a link in the form of an URL, a title and a text snippet. A snippet is typically a summary or preview of the webpage result, and often occupies a larger space in the SERP than the URL and title. Some search engines may control the snippet length to match the user's intended goals through the analysis of the user's activities and input queries [2]. On mobile devices, some snippets are able to be expanded with extra onscreen button prompts [2]. These features are implemented with the intention of reducing the time and effort it takes for a user to find their desired web resources, and enhancing their overall web searching experience [2].

A previous study has been conducted by Kim *et al.* [2] in which different users are given two different types of tasks to complete using search engines displaying different snippet lengths. The tasks were either informational tasks or navigational tasks. Various aspects of the users' performance were measured including time to first click, whether or not the user completed the task successfully, and the amount of time spent looking at certain elements on the page such as the URL, title and snippet [2]. Significant differences in the users' performance were noted between the two task types, and the three snippet lengths, including differences in the task completion time, time to first click, search accuracy, and users' attention fixation durations on the various elements of the SERP [2].

From the results of the study by Kim *et al.* [2], it can be hypothesized that there exists sufficient correlation between the users' search behavior and the lengths of the snippets they were assigned for a task to construct a predictive model, which can predict the latter using data from the former. Since the amount of time spent on a task was seen to be significantly affected by the lengths of snippets given to the user, there may be a correlation between the time related features in the dataset, such as the fixation times, with the length of the snippets given to the users. This paper will detail a predictive neural network model constructed using the data from Kim *et al.* [2] which will predict the users' assigned snippet length in the study based on various timing-related user performance attributes.

1.2 Technique

The model used will be based on the Casper neural network algorithm, which is based on the Cascade-Correlation (Cascor) algorithm [3]. The Cascor algorithm is a constructive algorithm which adaptively constructs a neural network in which each neuron is connected to all inputs and all previous neurons. It begins training with a single input layer connected directly to the output layer. Training is conducted until the network's error stops decreasing by a certain amount, at which point, the weights in the network are frozen and a single new neuron is added to the network [3]. Before being added, the new neuron is initially disconnected from the network and its input weights are trained to maximize the neuron's output with the network error. Cascor trains a pool of these neurons before selecting the best one to insert into the network. This results in a very efficient network as frozen weights can be cached and backpropagation only applies to a singular layer at any one time.

Cascor does come with its setbacks however. Earlier inserted neurons may perform poorly, and there is no further way to train them due to the frozen weights, so the network may end up with a large number of neurons, a portion of which are poor performing. The use of correlation within the Cascor algorithm also results in the algorithm performing poorly in certain tasks such as regression and some classification problems [4].

The Casper algorithm aims to address Cascor's issues and uses a modified version of the RPROP algorithm, which adaptively sets and adjusts different learning rates for different weights in the network. Casper networks function similarly in principle to Cascor, they start with the input and output layers as well as 1 hidden neuron and progressively builds the cascade network one neuron at a time. Compared to Cascor, Casper does not freeze the network weights. Instead 3 different training rates are assigned to the weights in the network depending on their location. The learning rates are designated L1, L2 and L3, with L1 having the largest magnitude, followed by L2, then L3 [4]. L1 is assigned to all weights connecting to the input of the newest hidden neuron in the network, L2 is assigned to the connection between the newest neuron and the output neurons, and L3 is assigned to all other weights. The weights allow old neurons to still be trained albeit at a much slower rate compared to new neurons, this rectifies Cascor's issues while retaining its benefits. Casper also makes use of weight decay to improve the network's generalization [4].

New neurons are inserted into Casper networks after a decrease of RMS error of at least 1% has occurred in a given time period. The time period is defined by the formula: 15+P*N, where P is a user defined parameter, and N is the number of hidden neurons currently in the network [4].

In addition to the Casper network, additional pre-processing of features in the data using fuzzy logic will also be employed. Fuzzy logic can be used to represent a certain value's membership in certain categories. Compared to binary logic, where one value is strictly in one of two categories based on a certain criterion, fuzzy logic uses membership functions to determine the degree of membership between 0 and 1 of a certain value in a certain category [8]. Membership functions are defined over the range of possible input values, and are typically triangular or trapezoidal in shape [8]. A value is said to be "fuzzified" when it is transformed into degrees of membership in different categories or fuzzy sets by the membership functions. For example, a height of 188 cm might be fuzzified into 0.6 degrees of membership in the "tall" fuzzy set and 0.4 degrees of membership in the "short" fuzzy set. Fuzzification may be applied to some inputs of a neural network in order to enhance their usefulness to the network.

This paper describes the implementation of a Casper-based neural network model with fuzzy logic data pre-processing to predict the snippet lengths assigned to users based on the timing-related performance attributes outlined in the Kim *et al.* [2] study. Training and testing of the model will be done with the original data used in the study. The model will also contain several modifications compared to the original Casper implementation detailed by Treadgold and Gedeon [4], and will be compared to a standard fully connected neural network model featuring only a single hidden layer.

2 METHOD

2.1 Model implementation

The Casper-based neural network model used in this study will be implemented using the Python package PyTorch. In the implementation, the model keeps track of a list of hidden neurons, all with sigmoid activation functions and 1 output only. The input of hidden neurons varies depending on the neuron's position, the very first hidden neuron is connected to all network inputs, and each subsequent hidden neuron is connected to all network inputs and all the previous hidden neurons. So each new neuron added must have an input size of I + N, where I is the model's input dimension and N is the number of hidden neurons already in the network. The output layer is connected to all network inputs and all hidden neurons, so it has an input size of I + N, and each time a new hidden neuron is added, the output layer's input size has to grow by 1. As there are three snippet lengths in the study by Kim *et al.* [2], the task this model solves is a multiclass classification problem, therefore the output layer's has 3 outputs. The growth of the output layer is done by initializing a new (3, 1) weights tensor (with PyTorch's default initialization method for a linear layer weight tensor: the Kaiming uniform initialization) and attaching the new weights vector to the old output layer weights vector.

The model's forward pass loops through all the hidden neurons. To account for the cascade architecture of the network, for each neuron, the input data is passed in and then the output of the neuron is concatenated to the input data and passed to the next neuron. This continues all the way through all hidden neurons until the output layer. The output layer now receives an input of I + N dimensions, and outputs its final predictions.

At this point, there are several differences in this paper's Casper implementation compared to the original publication [4]. First, the criteria by which new neurons are added are slightly different. This paper's implementation does not rely on a 1% decrease in RMS error, and instead looks for a 1% decrease in cross-entropy loss. This modification was due to the fact that RMS error would not be ideal in a multiclass classification problem, as the target and predicted values are only 0, 1 and 2. Cross-entropy is widely used in classification problems and have been shown to lead to faster training and improved generalization [5], so it is used here. As such, every 15+P+N epochs, the cross-entropy loss is recorded and compared to the previous time period's loss, if it has decreased by more than 1%, a new neuron is added to the network.

Every time a new neuron is added, the output layer input is expanded as detailed above, and all network learning rates need to be re-adjusted according to the new neuron's position. This point also deviates from the original implementation, the optimization algorithm used is not the RPROP algorithm but rather the unpublished RMSProp algorithm proposed by Hinton [6] in his lecture series. RMSProp is an adaptation of the RPROP algorithm for mini-batch gradient descent (which is also used in this implementation in an attempt to increase convergence stability, and is another point of difference from the original Casper paper). RPROP does not take into account the property where when learning rates are small, the gradient is effectively averaged over successive mini-batches. Many small adjustments in one direction then one big adjustment in the opposite direction equal in magnitude to the sum of the small adjustments would lead to the weight staying in roughly the same value, but RPROP would push the weight in the first direction much more than in the second. RMSProp tries to address this problem by accounting for the magnitude of the gradient during weight adjustments as well [6]. RMSProp also has a weight decay parameter, which is used in the implementation instead of the Simulated Annealing term in the original paper.

The L1, L2 and L3 learning rates are applied via parameters provided to the RMSProp optimizer. Their values are 0.2, 0.005 and 0.001 respectively, as detailed in the original Casper paper [4]. The newly added neuron's linear layer's weight tensor learning rate is set to L1, and the default learning rate for the network is set to L3 in the optimizer. The L2 learning rate presents some challenges as it is located at the new (3, 1) weight tensor added to the output layer's weight tensor. Due to this location, an approximation must be used here as PyTorch is unable to set the learning rate of only a part of a weight tensor. The approximation is as follows: following the calculation of gradients in the network and before stepping the optimizer, a scalar value equal to L2 divided by L3 is multiplied to the right most column of the output layer's gradient tensor (the gradient for the connections to the output of the newest hidden neuron). Since the optimizer's default learning rate is L3, the approximations hopes that L2 learning rate can be simulated when L3 learning rate is multiplied to a gradient which has been already been multiplied by L2/L3.

Lastly, the loss function used for this model is cross-entropy loss, as previously mentioned. The number of epochs was set to 200, and the batch size was set to 20.

To compare to this Casper-based model, a standard neural network model with only 1 hidden layer is implemented. The hidden layer contains 5 hidden neurons as this is approximately the number of neurons added by the Casper model given the other parameters. The standard neural network also uses mini-batch gradient descent and uses the same number of epochs and batch size as the Casper-based model. The standard network's loss function was also cross-entropy loss and its learning rate is set to the L1 learning rate from the Casper-based model.

The two models are compared on the basis of their confusion matrices and its associated measures, and loss values.

2.2 Dataset

The dataset used in this study is the eyegaze search snippets dataset generated by Kim *et al.* [2] during their study. The main tab of the dataset is used, and the "Snippet_length" attribute is used as the prediction target. Only 6 attributes were used as input features, this includes "Time to first click", "Fixation_snippet", "Fixation_total", "Task_type", "Fixation _title" and "Fixation_URL". All input features except for "Task_type" are based on time and measure the number of seconds elapsed for some aspect of the participants' performance. For "Time to first click", the seconds before the participants made their first click in the task was recorded. For all other tasks with "Fixation" in their title except "Fixation_total", the amount of seconds the participants' eyes fixated on certain aspects of the webpage were recorded. "Fixation_total" is the sum of all other "Fixation" attributes. The "Task_type" attribute was included even though it was not a timing-related attribute because according to Kim *et al.* there were significant differences in user performance times between different task types, so it is included to try and model this aspect of the data.

A few of the attributes needed recoding, this included the target attribute "Snippet_length", which was recoded to 0, 1 and 2 for short, medium and long tasks respectively. The attribute "Task_type" was also recoded to 0 and 1 for informational tasks (info) and navigational tasks (nav) respectively. The 6 features were to be given as inputs to the standard neural network model, while additional fuzzy logic pre-processing was to be performed on the 5 timing-related attributes before they were given to the Casper-based model. Normalization was performed on the unfuzzified features

which were given to the standard model. Normalization was not needed for the fuzzified features for the Casper model as the fuzzification process already transformed values to a range between 0 and 1. When normalization was required, Z-score normalization was used. Lastly, a train/test split of 80/20 was used.

2.3 Fuzzification pre-processing

The fuzzy logic pre-processing consisted of constructing fuzzy membership functions representing the degree of membership of a numeric value in each of the labels, then fuzzifying all numerical input features. This is done with the hopes of breaking down a simple numeric value into more meaningful indications of which label it may correspond to. The membership functions used were triangular, and so needed 3 values to define the left and right corners of the triangle, as well as its peak. These values were found for each label and feature by first separating the data depending on which label they correspond to. Then for all the features in each label, check what their min, max and mean values were, these would form the left, right and peak values for the membership function for that label. The range was also found for the feature and used as the universe of discourse for the membership function. This was done for every feature so that each feature had membership functions for all 3 labels, then all values in that feature column were fuzzified using the membership functions to obtain 3 new columns containing values between 0 and 1 representing degrees of membership in each label. The 5 input feature besides "Task_type" were fuzzified, each generating 3 new columns representing their membership in each label. Their unfuzzified columns were then discarded from the dataset. This means after the fuzzy logic pre-processing, there were 16 input features in total being fed into the network. The visualized fuzzy membership functions for all fuzzified features can be found in Appendix 1.

3 RESULTS

As a standard for comparison, the hyper-parameters for both the Fuzzy + Casper-based model and the standard model were both set to 200 epochs and 20 batch size. The user-defined P parameter for the Fuzzy + Casper-based model was set to 5 as this was determined to be the optimal value from a previous study [7], and the number of hidden neurons in the standard model was set to 5. The standard model was given the 6 original, unfuzzified features, while the Fuzzy + Casper model was given the 16 features resulting from the fuzzification. Each model was run 5 times, each time with randomized train and test split datasets. The accuracy, precision and recall for each label, and the macro average precision and recall values for the training and testing datasets are presented in Tables 1 and 2 respectively.

Table 1. Training accuracy, precision and recall for each label, and the macro average precision and recall values for the standard and Fuzzy + Casper models

Metric	Standard model	Fuzzy + Casper model
Accuracy	**0.56**	**0.51**
Label 0 precision	0.42	0.74
Label 1 precision	0.73	0.35
Label 2 precision	0.58	0.56
Average precision	**0.58**	**0.55**
Label 0 recall	0.73	0.51
Label 1 recall	0.47	0.52
Label 2 recall	0.77	0.60
Average recall	**0.66**	**0.56**

Table 2. Testing accuracy, precision and recall for each label, and the macro average precision and recall values for the standard and Fuzzy + Casper models

Metric	Standard model	Fuzzy + Casper model
Accuracy	**0.38**	**0.35**
Label 0 precision	0.32	0.51
Label 1 precision	0.52	0.21
Label 2 precision	0.28	0.40
Average precision	**0.37**	**0.37**
Label 0 recall	0.51	0.32
Label 1 recall	0.33	0.32
Label 2 recall	0.52	0.51

Average recall	**0.46**	**0.39**

In both Tables 1 and 2, the standard model appears to have outperformed the Fuzzy + Casper model, with generally higher accuracy, and higher macro average precision and recall in both the training and testing datasets. The standard model also seems to almost outperform the Fuzzy + Casper model in all individual label precision and recall values with the exception of Label 0's and Label 2's precision in testing and Label 1's recall in training.

In order to determine the exact cause of the diminished performance in the Fuzzy + Casper model, the fuzzy pre-processing was removed from the Casper model. The 6 original features were normalized and fed into the Casper model. The model was run 5 times, each time with randomized instances in the train and test datasets. The accuracy, precision and recall for each label, and the macro average precision and recall values for the training and testing datasets are presented in Table 3.

Table 3. Testing accuracy, precision and recall for each label, and the macro average precision and recall values for the Casper model with no fuzzy pre-processing

Metric	Training	Testing
Accuracy	**0.53**	**0.38**
Label 0 precision	0.57	0.38
Label 1 precision	0.52	0.36
Label 2 precision	0.53	0.42
Average precision	**0.54**	**0.39**
Label 0 recall	0.58	0.45
Label 1 recall	0.47	0.27
Label 2 recall	0.60	0.53
Average recall	**0.55**	**0.42**

By comparing Table 3 to Tables 2 and 1, it can be seen that the performance of the Casper model alone is still roughly the same or worse compared to the standard model, with generally the same or lower accuracies and averages in both the training and testing results. Some individual labels' precision or recall for the Casper model appears to be higher than the standard model's, such as Label 2's testing precision and Label 0's training precision. Even so, the Casper model seems to be a lot lower in some other individual label's precision/recall values such as Label 0's training recall.

When compared to the Fuzzy + Casper model's results, the Casper model appears to perform almost identically during training, with a less than 0.05 difference between the accuracy and macro average precision/recall. Individual differences in the label precisions and recalls exist, but there are no clear trends in the differences to directly indicate improvement or deterioration. In testing, the Casper model appears to have better accuracy and macro averages compared to the Fuzzy + Casper model, but once again, it is difficult to tell as there are no clear trends in how individual labels' precisions and recalls have changed.

Lastly, the standard model had an average training loss of 9.13 while the Fuzzy + Casper model had an average training loss of 9.97 and the Casper model alone had an average training loss of 9.91. Again the standard model outperforms either of the Casper models with a lower loss value, indicating a more fitting model for the data. While the Casper model seems to have roughly the same amount of loss regardless of whether fuzzy logic pre-processing is done, with only a 0.06 decrease in loss without the pre-processing.

4 DISCUSSION

From the performance metrics, it can be concluded that there is no added benefit from using a Casper model with fuzzy logic pre-processing. The standard model was able to effectively outperform the Casper model both with and without the fuzzy logic pre-processing. Even though the Casper model may predict some individual labels with better precision or recall than the standard model, it often has other labels with significantly worse performance. Taking into account the considerably more complex implementation of the Casper model and the fuzzy logic pre-processing compared to a standard neural network model, it is clear that the standard neural network model is a better choice for the task.

There may be several reasons why the Casper model was not able to offer additional benefits; it may be possible that the data itself is not enough to successfully predict the labels. No model within this experiment was able to achieve an accuracy or average precision/recall of above 0.7. All models within this experiment also had a relatively similar level of training loss, all at around 9. This indicates that even though there were differences between the different models, they generally were not able to perform any better than a certain threshold. This may be indicative of a problem with the data and the problem itself. It may be possible that the features used in this experiment are simply not enough to properly classify the target, and as a result, none of the models were able to perform well.

The fuzzy logic pre-processing appeared to have added no benefit, and at times even seemed detrimental to the performance of the Casper model. This may be due to the definitions of the fuzzy membership functions for each of the labels. For each of the features which were processed, their range of values for each label had significant overlaps. This significantly reduces the usefulness of the fuzzy encoding as some values may be encoded with very similar degrees of membership for more than 1 label, which provides little to no new information about how it should be classified. An example of this overlap can be seen in Fig. 1.

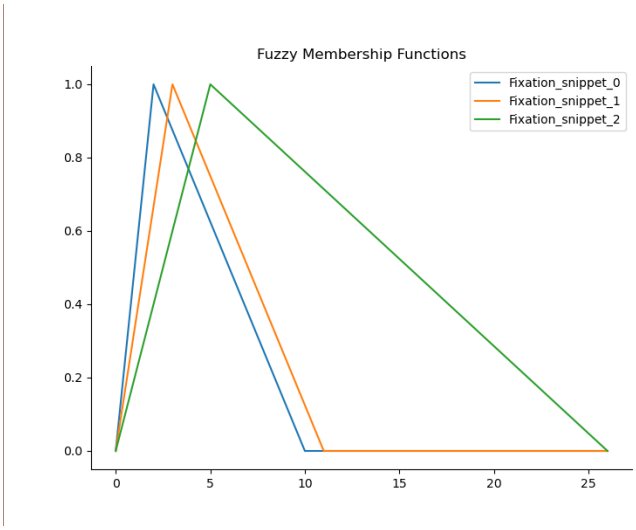

Fig. 1. Example fuzzy membership function visualization for the "Fixation_snippet" feature.

Due to this overlap, the current fuzzy encoding method used in this study is ineffective and should not be used in the future.

Another potential issue in this study may simply be the use of Casper algorithm or issues with its implementation. The usage of the Casper algorithm may not have been appropriate for this problem. The Casper algorithm was originally tested on the much more complicated two spirals dataset [4], so the results are not comparable with this study's results. But it's very likely that the algorithm would be much more suited to complex problems which have been shown to be difficult for standard backpropagation [4], rather than a problem which could be very plausibly solved by standard backpropagation, such as the one in this study.

One other obvious flaw with this study's Casper algorithm is the differences in implementation from the original. It's highly probable that the approximation of the L2 learning rate was ineffective, or that RMSProp is not as effective as RPROP and its weight decay is not as effective as the simulated annealing weight decay. All these differences in implementation may have caused significant discrepancies in the performance of this study's Casper model and the originally described Casper model [4].

5 CONCLUSION

This study has shown that based on the eyegaze search snippets dataset obtained by the Kim *et al.* [2] study, a neural network model which can accurately predict the user's assigned snippet size from their timing-related performance attributes was difficult to construct. It was demonstrated that on average a standard 2-layer neural network model outperform Casper-based models with fuzzy logic pre-processing on this task. This suggests the possibility that Casper-based models offer little to no advantage in this study's classification problem and that a standard neural network should be used instead.

It was also shown that the fuzzy logic pre-processing technique used in this study offered little to no benefits to the performance of the Casper model, as the overlap in fuzzy membership functions for the labels meant that the fuzzy encodings provided little additional information for distinguishing different labels.

Some avenues for future work may be to investigate more advanced methods of fuzzy encoding, perhaps using fuzzy clustering to assign probabilities of each instance being a label [8].

For the Casper algorithm, as the optimization algorithm RMSProp is an unpublished modification on the RPROP algorithm central to the Casper algorithm, there is still much to research in terms of fully understanding its characteristics or improving its performance [6]. A more efficient variation of the RPROP algorithm may aid in the further improvement of Casper-based algorithms, and may be a worthy avenue of research.

References

1. Teevan, J., Cutrell, E., Fisher, D., Drucker, S.M., Ramos, G., Andre, P., Hu, C.: Visual Snippets: Summarizing Web Pages for Search and Revisitation. Web Searching and Browsing. (2009)
2. Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T., Yoon, H.: What Snippet Size is Needed in Mobile Web Search? In *Proc. 2017 Conf. Human Information Interaction and Retrieval*. pp. 97-106 (2017)
3. Fahlman, S.E., Lebiere, C.: The Cascade-Correlation Learning Architecture. *Advances in Neural Information Processing Systems.* pp. 524-532 (1990)
4. Treadgold, N.K., Gedeon, T.D.: A Cascade Network Algorithm Employing Progressive RPROP. In *Proc. Int. Work-Conference on Artificial Neural Networks.* pp. 733-742. (1997)
5. Bishop, C.M.: Pattern Recognition and Machine Learning, Springer, New York (2006)
6. Hinton, G. (2012) Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude, Lecture, University of Toronto
7. Liu, Z.: Casper-based Neural Network Modelling for Prediction of User Search Task with User Search Behaviour Attributes In *Proc. 4th ANU Bio-inspired Computing Conference*, pp. 1-9. (2021)
8. Ross, T.J.: Membership Functions, Fuzzification and Defuzzification in *Fuzzy Systems in Medicine*, Physica, Heidelberg. (2000)

Appendix 1

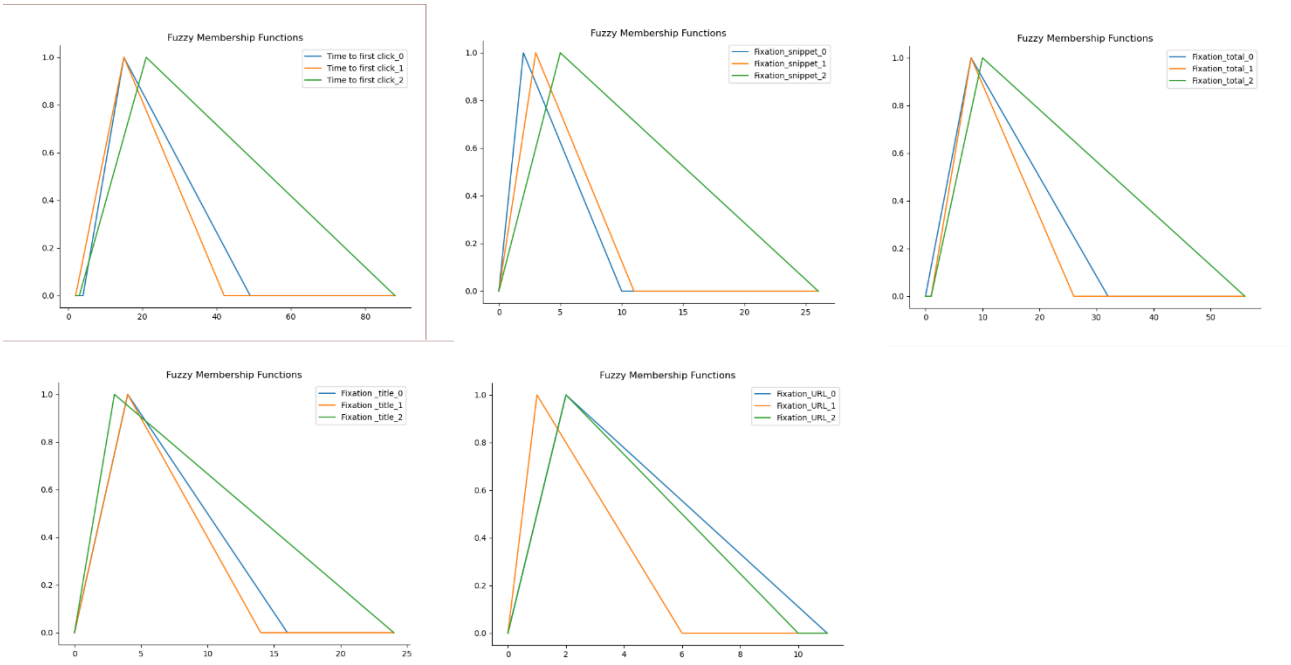

Fig. 2. Examples of fuzzy membership function visualization for all pre-processed features.