Neural Scoring of Logical Inferences from Data using Feedback

Allmin Susaiyah[1]*, Aki Härmä[2]*, Ehud Reiter[3], Milan Petković[1]

[1] Eindhoven University of Technology, Eindhoven (Netherlands)
[2] Philips Research, Eindhoven (Netherlands)
[3] University of Aberdeen, Aberdeen (Scotland)

Received 16 October 2020 | Accepted 31 January 2021 | Published 15 February 2021

Abstract

Insights derived from wearable sensors in smartwatches or sleep trackers can help users in approaching their healthy lifestyle goals. These insights should indicate significant inferences from user behaviour and their generation should adapt automatically to the preferences and goals of the user. In this paper, we propose a neural network model that generates personalised lifestyle insights based on a model of their significance, and feedback from the user. Simulated analysis of our model shows its ability to assign high scores to a) insights with statistically significant behaviour patterns and b) topics related to simple or complex user preferences at any given time. We believe that the proposed neural networks model could be adapted for any application that needs user feedback to score logical inferences from data.

Keywords

Artificial Intelligence, Feedback Learning, Neural Network, Self-supervised Learning, Transfer Learning, Logical Inference, Natural Language Generation, Statistical Learning.

DOI: 10.9781/ijimai.2021.02.004

I. Introduction

Technological advancements in this century have led to a rise in the number of applications that claim to improve human lifestyle. A few important examples of these are popular health, diet and fitness mobile applications that have stormed the market. These applications obtain data from activity trackers or loggers and play the role of an artificial health or fitness agent by generating actionable insights of users' behaviour [1], [2]. For this purpose, it is desired that insights should be valid, represent a significant pattern of the user behaviour and should align with their interests. Insights can be of different levels of complexity. An example of a simple count-based absolute insight is: "You need to take 100 more steps to reach your daily goal!". Whereas, a more complex comparison based insight is "You sleep less when the room temperature is above 20 degrees than when it is lower.". Henceforth, in this paper, we consider only the comparative insights that are more complex and challenging. In general, these insights talk about a measure in two contexts, for example, by stating that a measure X is larger in context A than in context B, see [3]. This statement requires the test for statistical significance on two distributions, one from each of the contexts that are being compared.

For the purpose of testing the statistical significance of comparative insights, parametric and nonparametric significance tests have been widely used. However, there has been no specific technique to understand user interests in an intuitive manner. The biggest challenge in performing this is the very nature of user interests: they keep changing. Hence, a highly flexible model is required for this purpose. The artificial neural network (ANN) model, commonly described as a universal function approximator has shown great ability to learn, unlearn and transfer knowledge from one domain to another. Additionally, the ability of ANNs to learn multiple input characteristics encourages us to model multiple domains at once, such as the statistical significance domain and *interestingness* domain. This makes ANNs to be a favourable choice to model user preference.

Insight generating systems shouldn't produce inferences that may harm the goal of the application. We can best guarantee this, by a system where all texts are selected from a pre-generated and manually curated collection of *validated insight candidates*. This is similar to the PSVI method introduced in [3]. These validated insights form the base for the rest of this paper. The statistical significance domain considered in this paper corresponds to a well-known nonparametric significance test, namely, the Kolmogorov-Smirnov (KS) test. The interestingness domain incorporates how much a user is interested in knowing about a particular comparative insight.

In this work, firstly, we develop an insight generation system that generates validated insights using a behaviour insight mining pipeline. Secondly, we train a self-supervised neural network that can upscale and replace traditional nonparametric tests (with 92% accuracy at 5% alpha). Lastly, we show how this ANN can also be used to learn user preference using an interactive learning strategy. For this, we use a evaluate using a single user-preference scenario and multi user-preference scenario.

The characteristics of insights that are considered by our model are essential for highly scalable behaviour insight mining (BIM) systems.

* Corresponding author.
E-mail addresses: a.p.s.susaiyah@tue.nl (A. Susaiyah), aki.harma@philips.com (A. Härmä).

Applications of this can be in fitness coaching, office behaviour [4], behaviour change support systems [5], [6], business insight mining systems [3], and other relevant systems.

The structure of this document is as follows: section II gives a brief background about insight with examples, section III provides a in-depth explanation of how we developed the neural network architecture and how the online learning system is implemented. The results of our ANN and the simulated user scenarios are covered in section IV, discussion is included in section V while section VI presents the conclusions.

II. Background

In this section, we provide provide more context to the concepts discussed in this paper.

A. Desirable Characteristics of Insights

Based on recent literature, an insight should have the following characteristics, namely, statistical significance [3], [7], interestingness or personal preferences [3], [8]–[11], Causal confidence [10], surprisingness [8], actionability or usefulness [8], [9], syntactic constrains [7], presentability [11] timely delivery [11], and understandability [9]. Among these, the most essential characteristics are statistical validity and interestingness.

B. Types of Insights

1. Generic insight: These are insights that talk about a rather common or scientific phenomenon. These are not grounded on the user's behaviour. For example: Excessive caffeine consumption can lead to interrupted sleep as can ingesting caffeine too late in the day.
2. Personalised (Manual/Automated) insight [12]: These are insights that are tailored to the user either by a human-in-loop or by an algorithm.
   - Absolute insights or simple insights: These insights talk about user behaviour in one context. We do not focus on such insights in this paper as they are less actionable.
   - Comparative insights: These insights compare the user behaviour between two contexts [3] as shown in Table I.

TABLE I. Examples of Comparative Insights in BIM

Comparison	Example
time-specific	On **weekdays** you walk less than on **weekends**
parameter-specific	Your **heart rate** is higher on Mondays than on other days
event-specific	**When you bike**, you spend less calories per minute than **when you run**

C. Generation of Validated Insights From Data

Thousands of insights can be generated from even a simple database by slicing and dicing the data into different views. To streamline this process, we formulated a behaviour insight mining pipeline [13]. It consists of specialised blocks to look at data (what-to-look, where-to-look, how-to-look) and to generate text (what-to-say, when-to-say and how-to-say). For example, to generate the insight "On Weekdays you sleep less than on Weekends", the database should have logs of user's sleep duration and corresponding dates (what-to-look). The rows of the database corresponding to weekdays are considered as bin A and those corresponding to weekends are considered as bin B (how-to-look). Relevant filters are used to extract these rows (where-to-look). On comparing the average user's sleep duration in each bin, we find that bin A has a significantly lower value than bin B (what-to-say). Subsequently, a statistical significance test is performed to prove its statistical validity. A text realisation block structures and generates the appropriate textual output (when-to-say and how-to-say). Similarly, many comparisons (how-to-look) could be made between two periods such as:

- Mondays and other days
- Workdays and holidays
- February and March

Generally, thousands of such insights can be generated from even a moderately sized data. We validate these insights with the help of domain experts and proceed further.

A detailed description of how insights are generated and validated is explained in [3], [13].

D. Nonparametric Statistical Significance Tests

The data extracted from the two periods mentioned above come from two nonparametric sample distributions. The two most commonly adapted techniques to determine the statistical significance of such distributions are KS test and Mann-Whitney U (MW) test. The former is based on the shape of the distributions and the latter is based on the ranks of the samples. In this paper, we use data from a sleep monitoring device that measure the duration of sleep, sleep latency, etc [14]. Although these measures follow normal distribution, when looked at different slices and dices of the data such as Mondays vs Other days, they become nonparametric. Hence, we choose the KS test in this study. However, the MW test can also be used instead.

E. Neural Statistics

Neural networks have been used for wide range applications in Machine Learning such as signal denoising [15], image classification [16], stock prediction [17], and optical character recognition [18]. The ability of the neural network to learn basically any complex function makes it a universal function approximator. The simplicity in the way by which a neural network generates an inference makes it a suitable choice for many applications. Additionally, the transfer learning capability of the network [19]–[21] allows us to transfer the pre-learned knowledge of the network to solve different and more complex problems. This inspired us to use the neural network to approximate the statistical significance test.

F. Online Learning of User Preference

By permuting different contexts one may often find a large number of statistically significant insights but not all of these insights are useful to the user. Hence, the user's preference must be considered before presenting the insights to them. The personal preferences of end-users change with time. Filtering the insights based on statistical validity alone is not sufficient to satisfy their interests. A method to learn a user's preference in a convenient and flexible manner will solve this problem. Online learning technology can train models in a flexible manner while still being deployed in a consumer product or health coaching service [22], [23]. There is no existing literature on online learning of user preference nor the neural learning of statistical validity. Such learning will be of great use in BIM applications.

In this work, we present an online learning strategy that learns user preference while simultaneously maintaining the ability to realise the statistical significance. We make use of self supervision and transfer learning techniques to achieve this. In our technique, we assume that the user is interested in $N \in \mathbb{N}$ types of insights simultaneously at any point in time. The results indicate consistent performance for various values of N.

III. Methodology

The entire methodology was carried out in two stages, namely, the self-supervised learning stage and the online learning stage. Although each stage has a different data source, model architecture, training, and validation strategy, they share an important connection. The second stage model was transfer-learned from the first stage model. In this section, we describe the above-mentioned stages in detail.

A. Stage I: Self-Supervised Learning Stage

Self-supervision approach has been widely used to enrich a neural model using the input data and transformations of the input without the need for manual labelling. it has been widely used in fields like computer vision [24] language modelling [25] and speech modelling [26]. As a first stage, we conceptualised and developed a neural network model that learned rich feature representations to determine the statistical validity of comparative insights. We achieved this by training the model with highly diverse synthetic data. The data generation and model training are described below.

1. Problem Formulation

Let us consider an insight i that compares two distributions d_1 and d_2. The KS significance test can be represented as a function $f(d_1, d_2)$ that determines the p-value of d_1 and d_2. If the p-value is less than the significance level α, then, d_1 and d_2 are considered significantly different. We formulated a neural network N that approximates f as shown in Equation 1.

$$f \sim N \tag{1}$$

The neural network learns the function f by minimising the mean squared error loss function J_1 as shown in Eq 2.

$$J_1(\theta) = \frac{1}{n}\sum_{i=1}^{n}(f(d_{1i}, d_{2i}) - N_\theta(d_{1i}, d_{2i}))^2 \tag{2}$$

2. Data Generation for Base Model Selection

A dataset containing 300000 pairs of histograms of uniform distributions was generated using the NumPy-python package. The number of samples, mean and range of each distribution was chosen randomly. The ground truth labels for each pair of distribution were generated using the p-values of the two-sample KS test. The SciPy-python package was used for this. We compared it with our less optimised implementation of KS test and found it to give the same p-values. The dataset was subdivided into three equal parts, each for training, validation, and testing. We also made sure that each portion had balanced cases of significant and insignificant pairs.

3. Finalisation of Base Model Architecture

A domain-induced restriction of comparative insights is that the number of inputs is two and the number of outputs is one. Here, each input is the histogram of one of the distribution and the output is the statistical significance. Based on previous works on similar input/output constraints [27], [28], we came up with three neural network architectures, namely, a recurrent neural network (RNNA), a modified RNN (RNNB) and a siamese network (SIAM). The schematics of the RNNA architecture are shown in Fig. 1. The layers Ip1 and Ip2 are input layers, each having a fixed size of 100 elements. The layers F1 and F2, are fully connected layers, each with 50 neurons activated by a Leaky Rectified Linear Unit (ReLU) function. In fact, all layers in the network except the Final layer are activated by the Leaky ReLu function. Another level of fully connected layers, namely, F3 and F4 follow F1 and F2 respectively. We chose the number of neurons in each of these layers to be 20, which is less than the preceding layer, to have a compressed representation of the input signal. This type of step down architecture is commonly seen in the encoder part of auto-encoder neural networks [29]. This type of compression is helpful in transforming the input from spacial domain to meaningful feature domain. The layers F3 and F4 are concatenated and fed to a Simple Bidirectional Recurrent Neural Network (RNN) with 100 units. The rationale behind using an RNN is that the input needs to be considered a sequence rather than a vector as the inputs belong to two different contexts. We added another fully connected layer (F5) having 100 neurons to the output of the RNN. We believe that this layer generates rich features learned from the input data. The final layer is also a fully connected layer with one neuron activated by a thresholded ReLU activation function.

The RNNB model has every layer similar to the RNNA layer, except that it has 100 neurons in the F1 and F2 layers instead of 50. This is to see if increasing neurons would increase performance for a fixed purpose and input size. The SIAM network is also similar to the RNNA architecture, except that the F3 and F4 layers are subtracted rather than being concatenated and the RNN layer is replaced by a fully connected layer with 100 neurons.

4. Base Model Training and Testing

We trained and validated the three models in a self-supervised manner using the pairs of uniform distributions (histogram). The histogram was squeezed to 100 bins and the minimum and maximum range of histograms are fixed to be the minimum and maximum range of the dataset. This allows all the histograms to be comparable. Uniform distributions were chosen due to their close resemblance to real data that is commonly encountered in insight-mining tasks. In total, each of the training, validation and testing phases consisted of 100000 data samples. We did not go for an unequal split as we did not have that necessity due to the possibility to synthesise infinite data. The training was governed by Adam optimiser with a mean-squared-error loss function. The model that gave the best performance on the test set was considered as the base model. However, in real life, the data could also arise from complex or mixed distributions. Hence, we proceeded further with another level of fine-training.

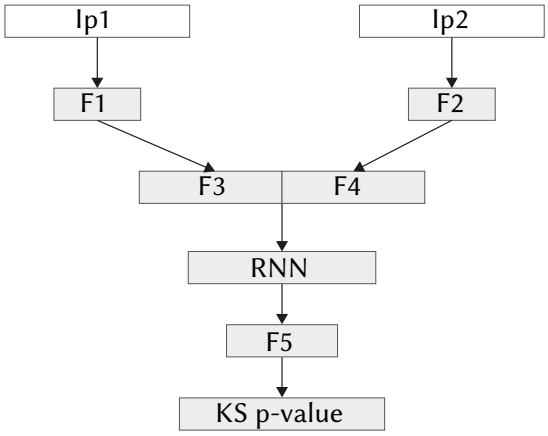

Fig. 1. Self-supervised neural network architecture for significance testing.

5. Improving the Base Model

In reality, the models encounter complex and nonparametric data distributions. For example, The distribution of hours of sleep on Mondays may not be normally distributed, but might follow a nonparametric distribution. Such scenarios are not considered during the base model training. Hence, we further train it with more diverse pairs of distributions (histogram) such as Gamma, Gumbel, Laplace, Normal, Uniform and Wald. On the whole, a total of 360000 pairs of distributions were generated and were equally split into training,

validation and testing sets. Each of these sets consists of 120000 pairs of distributions (20000 pairs of each distribution). Both inputs of the network are always fed the same type of distribution, but with different parameters. For example, if one input of the network is a normal distribution, the other input is also a normal distribution but with different mean, range, and cardinality. The training labels are generated earlier. The training was governed by Adam optimiser with a mean squared error loss function. Once trained, the model can be used as a smart alternative to statistical significance testing to filter significant insights among all insights.

B. Stage II: Online Learning Stage

In the second stage, we transformed the base model to detect interesting insights while preserving its ability to detect significant insights.

1. Problem Formulation

In this stage, apart from the two distributions d_1 and d_2, we are also interested in the user model $\emptyset$. The user's preference can be represented by a function $p_u(k)$ that generates an interestingness value for a given insight k. This function can also be considered as a user interestingness/preference model. We formulated a transfer learning approach that uses a portion of network N i.e, N' and augments it with features representations generated from another neural network Δ that uses the state vector s of the insight k. Finally, the augmented network drives the overall network O that approximates $p_u(k)$ shown in Equation 3.

$$p_u(k) \sim O(N^{'}(d_1, d_2), \Delta(s)) \tag{3}$$

The neural network learns the function p_u by minimising the mean squared error loss function J_2 as shown in Eq 4.

$$J_2(\theta) = \frac{1}{n}\sum_{i=1}^{n}(p_u(k) - O_\phi(N^{'}(d_{1i}, d_{2i}), \Delta(s_i))^2 \tag{4}$$

In this work, we show that any improvement in approximating p_u does not have an impact on the approximation of f in Equation 1.

2. User Model Acquisition

The online learning strategy detects interesting insights with-out being instructed by the user explicitly. It uses a feedback form in a mobile application that displays a few insights that were scored high by the base model. We simulated a user who may choose the insights that they are interested in and the neural model learns from it. A sample feedback form is shown in Table II. In this work, the preference of the simulated user changes every month.

This feedback is equivalent to "labelling" in traditional online learning theory. To generate the insights to validate our online learning system, we obtained sleep and environmental sensor-data collected from a bedroom of a volunteer over a period of 4 months from May 2019 to August 2019. We logged various parameters such as the timestamp of the start of sleep, sleep duration, sleep latency, ambient light, ambient temperature, ambient sound and timestamp of waking-up. We generated insights for each day of the user using the procedure explained in [3]. The insight texts talk about the two contexts that it compares and an expression of the comparison such as "less than", "longer than", etc. The number of insights per day varied between a few hundred to few thousand. We simulated the user preference given below by automatically filling the feedback form for each day.

1. May: The user is interested in Insights related to Weekdays.
2. June: Weekend insights are interesting to the user.
3. July: The user prefers to know more about his sleep duration.
4. August: The user is again interested to know if he/she is doing well on weekends.

TABLE II. A Sample Insight Feedback Form

Insight	**Are you interested to see more of these type of insights?**
On Weekdays you sleep less than on Weekends	◯
...	...
Your take longer to fall asleep on Mondays than other days	◯

Collecting daily-feedback from a real user is expensive and time-consuming. Hence, we simulated the above monthly user-preference pattern. With this, we forced the model to adapt to abrupt changes in preferences; posing significant challenges to the network. Initially, all insights were initialised with an interestingness score of 0. The simulator re-assigns all statistically significant insights per day on a given month that satisfy the corresponding preference criteria with an interestingness score of 1. Although we labelled all the insights as interesting or not interesting, we observed later (section IV.D) that in actual practice, only a fraction of these labelled data were be used for training. Additionally, to simulate conflicting feedback, we randomly toggled 10% of the interestingness scores from 1 to 0 and viceversa. Since neural networks understand only numbers, we encoded each comparison insights into a single dimension binary vector s containing 220 elements where each element corresponds to one parameter of comparison. For example, one element corresponds to each day of the week. Hence, if the comparison is related to Mondays and weekends, the elements corresponding to Mondays, Saturdays, and Sundays are assigned a binary one and the rest are assigned zero. We injected this vector into the model while transfer learning for interestingness recognition. In the following subsection, we explain how the model was transfer-learned and how the online learning pipeline was implemented and evaluated.

3. Transfer Learning

Transfer learning was performed to enable the model to learn insight interestingness in addition to significance. The self-learned model was frozen from the input layers up to and including the F5 layer. The vector s was passed as input to another fully connected layer F6 with 100 neurons. This layer was concatenated with the F5 layer as shown in Fig. 2. The concatenated layers are fed to another fully connected layer F7 having 100 neurons. While the layer F6 is linearly activated, the F7 layer is activated by the ReLu function. Finally, the

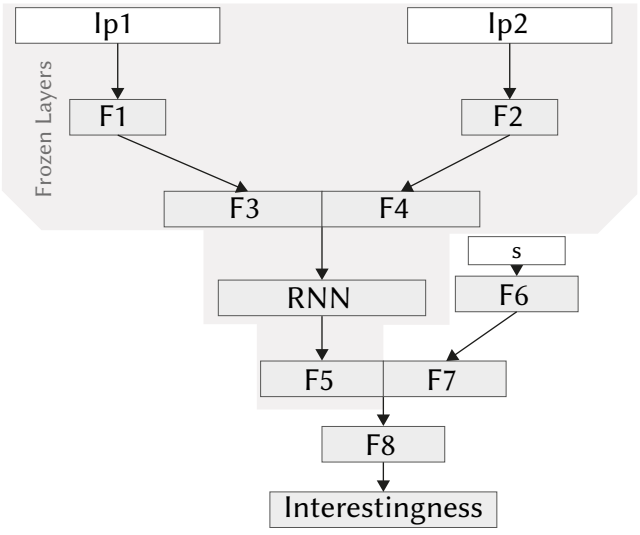

Fig. 2. Augmenting the base network for online learning.

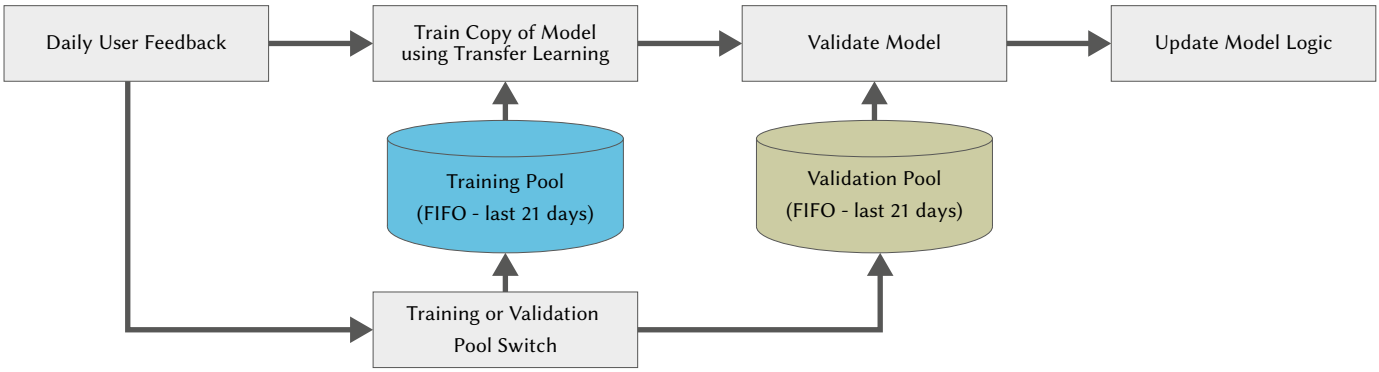

Fig. 3. Online learning through user feedback.

Algorithm 1: Training and Validation Switch Logic

Result: Assign sample to Training pool, Validation pool or Both

```
feedback = pop(feedback_stack)
prediction_error = validate_model(feedback)
if mode == Accelerated Training then
    if prediction_error < 0.3 and coin_toss() == Heads and positive_fraction in range [0.42,0.6] then
        switch_state = Both
        update(positive_fraction)
    end
else
    if prediction_error < 0.1 and coin_toss() == Heads and positive_fraction in range [0.42,0.6] then
        switch_state = Validation
        update(positive_fraction)
    else
        if positive_fraction in range [0.42,0.6] then
            switch_state = Training
            update(positive_fraction)
        end
    end
end
```

output layer is a single neuron fully connected layer activated by a sigmoid activation function. Notice that the final layer is activated by a sigmoid function as this is a binary classification problem trained on user preferences instead of significance. By performing this transfer learning, the model retains the features that correspond to the significance and simultaneously recognise the interestingness of insights based on user preference.

4. Learning Modes

The architecture of the online learning scheme is presented in Fig. 3. The scheme is executed in two modes, namely, accelerated learning mode and normal learning mode. These modes determine how well the models are trained. The accelerated learning mode, by default, starts from the first day of usage of the insight generator till the tenth day. Then, the normal mode begins. During the accelerated learning mode, the model learns quickly from the data and during the normal mode, it learns at a normal pace. This is achieved by varying the learning rate. Thus, the accelerated training mode has a higher learning rate.

5. Training and Validation of Switch Logic

Insights are generated on a daily basis. The insights contain a textual description of the behaviour and the back-tracking information of the corresponding data. Using this, we can get the data distributions corresponding to insights. Every day, the insights are assigned an interestingness value based on the user's feedback. The learning mode, prediction_error and positive_fraction help to determine if the feedback will be used to train or validate the model. This logic is represented in Algorithm 1.

The system collects the feedback and stores them in a FIFO stack named *feedback_stack*. The algorithm starts with popping a feedback from the stack and calculating its prediction error using the *validate_model* function. This function runs the neural model on the feedback data to predict an interestingness score and calculates its absolute difference from the true label. Then, the system follows subsequent steps to assign the data to one of the pool based on the *prediction_error*, learning mode, positive_fraction and a coin toss as show in Algorithm 1. Here, the *positive_fraction* is mean of all the interestingness score in the training pool, the *coin_toss* function generates a *Head* or a *Tail* randomly.

If the user does not give any feedback, the model does not update since the system implicitly assumes that the user's preference is unchanged.

6. Pool Maintenance Logic

Both the pools are maintained to hold only a maximum limit of days of data. We fixed this to be 14 days because we assume a user's interestingness remains fairly unchanged for a period of two weeks. Every 20 days, the model forcefully pops out 7 days of the oldest data in a FIFO fashion. This helps to avoid overloading the training and validation pools and forgetting older preferences. Additionally, the validation pool is completely emptied at the beginning of the first day of the normal learning phase.

7. Update Logic and Metrics

At the end of every day, a copy of the model is trained on the training pool and validated on the validation pool. If the validation accuracy exceeds a set limit (here 70%), the old model is replaced by the recently trained model. However, as an exception in the accelerated learning mode, the model is updated every day irrespective of its performance. This purposefully over-fits the model to the insights during accelerating learning mode. The performance of online learning is monitored using statistical measures, namely, sensitivity, specificity, and accuracy in predicting the interestingness of insights. Additionally, we introduce the significance preservation score, which is calculated as shown in Equation 5.

$$P_s = N_a / N_p \tag{5}$$

where, N_a and N_p are the number of actual interesting insights in the validation pool and the number of predicted interesting insights during validation, respectively. The P_s is not defined when N_p is zero. This is a limitation of the metric.

IV. Experimental Results

In this section, we present the results that we obtained at each stage.

A. Choosing The Base Model Architecture

An example of histograms of significant and insignificant pairs of normal distributions is shown in Fig. 4. It also demonstrates the variation of magnitude, range and cardinality (more samples have a smoother curve) of the synthetic data. Each of the base model architecture, namely, RNNA, RNNB, and SIAM were trained, validated and tested using the dataset containing only normal distributions. The performance of each model is presented in Table III. We observed that the RNNA model exhibits a test accuracy of 92% in predicting whether an insight is interesting or not. The performance of RNNA is thereby comparatively better than that of RNNB. This shows that more neurons do not always lead to improved performance. Also, RNNA exhibits slightly better performance than the SIAM network. This could be due to the sequential treatment of the data by the RNN which is part of the network. Additionally, since the SIAM network has fewer neurons, it also provides evidence that fewer neurons might not help either. In our view, the neural model should have an adequate number of neurons and parameters and an explainable architecture, which is, unfortunately, missing in recent works in this field. Hence, the RNNA architecture is chosen as the base model and considered for further analysis.

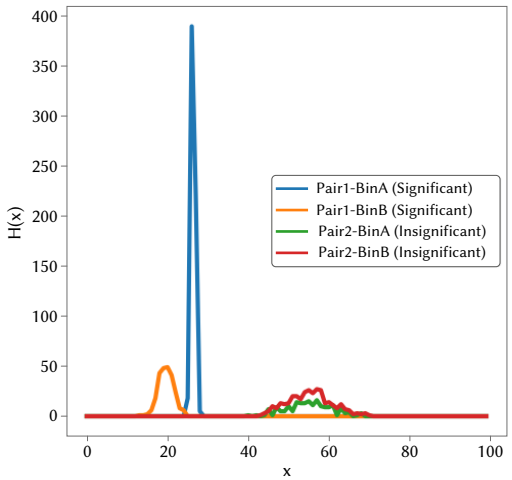

Fig. 4. Pair of normal distributions without significant difference.

TABLE III. Performance of Different Models while Training and Testing with Normal Distribution

Model	Description	Accuracy $\alpha = 0.05$
RNNA	Bidirectional RNN layer	0.92
RNNB	More neurons	0.86
SIAM	Siamese Network	0.87

B. Improving Base Model Training

We trained the base model using diverse pairs of distributions (histogram) such as Gamma, Gumbel, Laplace, Normal, Uniform and Wald. We observe that when we tested each distribution as shown in Fig. 5, we find out that the performance of the model to normal distribution remained at 0.92, but the uniform was even higher at 0.97. The worst performance was observed on Wald distribution. We have additional evidence that this is a limitation of the actual KS test that is being reflected in the neural model. It is also found that few distributions exhibit improved performances as alpha increases and few showed weaker performance as alpha increases.

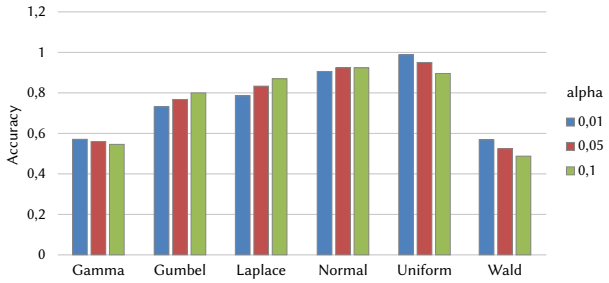

Fig. 5. Gaussian trained model on mixed distributions.

C. Generated Insights

We used the pipeline approach described in C to generate insights on the users sleep behaviour. A few examples are shown in Table IV.

TABLE IV. Representative Set of Generated User Behaviour Insights

S. No	Insights
1	In Q3, you slept shorter than in Q2
2	In October, it took longer for you to fall asleep than in September
3	In May, you spent less time in the bed than on other months
4	You slept longer, when the temperature during start of sleep was between 17°C and 30°C than when it was more than 30°C
5	It took longer for you to fall asleep, when the humidity during the start of sleep was high than when it was ideal
6	It took longer for you to fall asleep, when the illuminance during start of sleep was brighter than normal than when it was dim

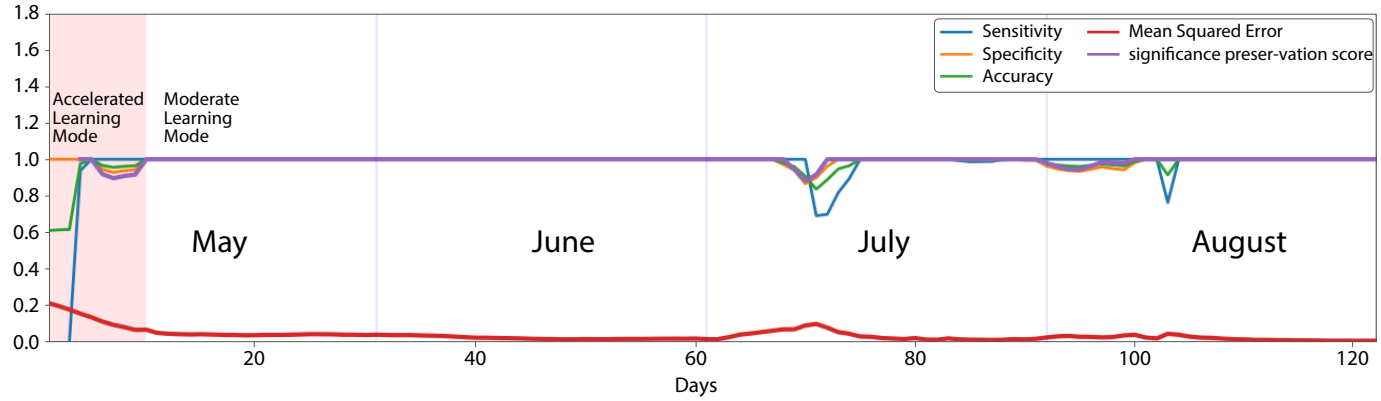

Fig. 6. Timeline of online learning with performance indicators.

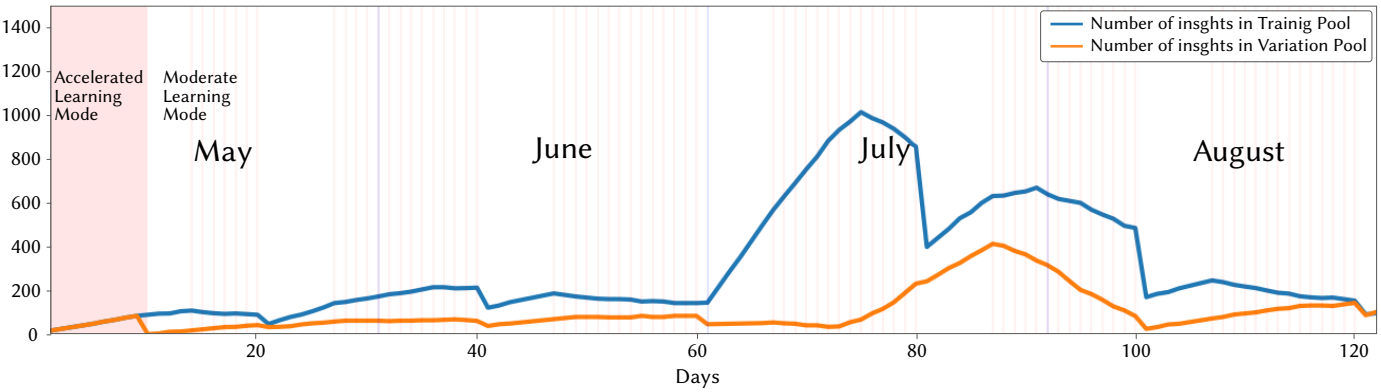

Fig. 7. Size of training and validation pool.

D. Online Learning on Single Preference

We simulated a user who has a preference for a single type of insight for a given month. We generated insights from another real user data and allowed the simulated user to provide feedback on each insight based on the implicitly defined preferences as follows

1. May: insights that talk about behaviour in weekdays
2. June: weekend insights
3. July: insights that describe how long the user sleeps
4. August: weekend insights

Subsequently, we initiated the online learning scheme and the performance metrics are presented in Fig. 6. We put the system on accelerated learning mode for the first 10 days. It is observed that the accuracy, sensitivity, and specificity were unstable during the first 4 days of the accelerated learning phase. From the fifth day onwards, the three measures show improvement and are in the range of 0.9 to 1. The Ps measure is not defined when there are no significantly valid insights that are interesting. This is observed till day 3 and on day 4, 100% Ps is observed. This implies that the model exhibits significance preservation starting at least from day 4 onwards. The performance is rather stable all the while during the remaining days of May and the entire June. Even though there is a transition between weekday insights and weekend insights, the model seems to adapt very well. In the months of July and August, there are visible drops in the performance around the 10th day of the month even though the preference changed on the 1st of both months. This could be an instability caused by the sudden rise in the training pool and reduction of validation pool data as shown in Fig. 7. In general, the pool maintenance logic is able to control the number of training and test data points. Although the first half of July saw a huge influx of training data, the maintenance logic prevented the training pool from overloading. Otherwise, there would have been a huge chance of exposing the model to noise in the data. The mean squared error (MSE) curve shows that the error between predictions and ground truth is not very high. The MSE decreased more steeply during the accelerated learning mode compared to the normal mode. There are periodic valleys in the training pool count and validation pool count denoting the reach of the 20-day window for cleanup of the pool. Also, additional cleanups are done every day when the number of days of insights in the pool exceeds 14. All cleanups on the training and validation pool are indicated by faint red vertical lines in Fig. 7. Additionally, we could observe that number of labelled insights (training + validation) at given point in time is in the range [20, 1087] however, the number of newly fed insights ranges from 0 to 74 per day with an average of 15.6 insights per day. Thus, it doesn't require to label all the insights, but only as much as required by the model.

E. Online Learning on Multiple Preferences

Usually, the user preference is not as simple as described in the previous section. It is a combination of multiple preferences. Hence, we simulated the user to have multiple insight preferences at a time. We first investigated in detail, the effect of a dual preference user model and secondly, discuss its general impact by simulating multi-preference scenarios up to 10 simultaneous user preferences.

1. Dual Preference

We considered a dual preference scenario where the user has following pairs of preferences:

1. May: insights that talk about user behaviour during weekdays or weekends
2. June: insights that talk about weekend behaviour or the user's sleep latency

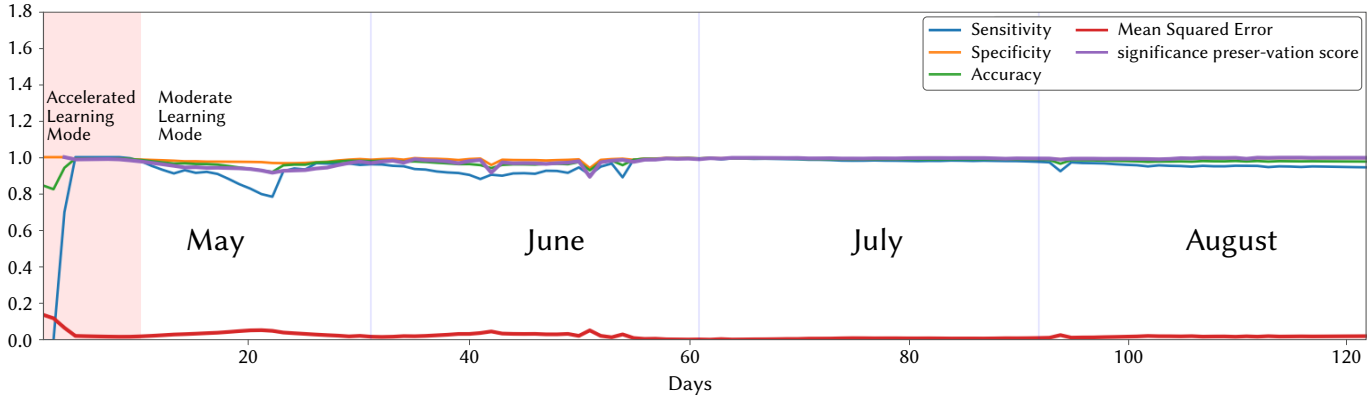

Fig. 8. Timeline of online learning on dual preference with performance indicators.

3. July: insights that describe how long the user sleeps and insights measured over the quarterly period
4. August: weekend-insights and insights talking about sleep latency

In the beginning, the model performs slightly better than on the single preference user model. There is a drop in performance around the twenty second day of the first month. However, the model is comparatively steady thereafter.

We had purposefully set the user preference in August to be the same as in June to see how well the model unlearns and relearns. From Fig. 8, it can be observed that the model learns in August, much more smoothly than in June. In overall, the dual preference scenario has slightly fewer and less intense performance drops than the single preference model.

F. Higher Order Preference

We defined a list of possible categories of insights as shown Table V. Each insight can belong to one or more categories. We simulated a multi-preference user by randomly choosing a combination of N out of the 14 insight categories for each month. We ran our learning algorithm under these conditions and measured the performance in terms of accuracy and preservation of significance as shown in Fig. 9.

TABLE V. List of Preference Profiles

S. No	Insights
1	Insights that talk about behaviour on weekdays
2	Weekend insights
3	Insights on duration of sleep
4	Time to fall asleep
5	Time spent on bed
6	Average time of getting into the bed
7	Average time of getting out of bed
8	Insights consolidating behaviour over a quarter
9	Insights talking about monthly behaviour
10	Yearly insights
11	Impact of humidity at start of sleep on sleep measures
12	Impact of humidity at end of sleep on sleep measures
13	Impact of temperature during start of sleep
14	Temperature during end of sleep

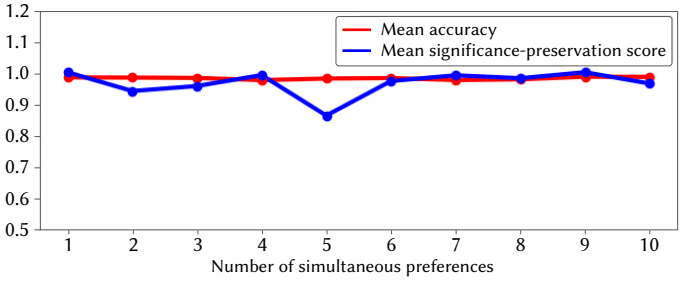

Fig. 9. Mean accuracy and mean significance preservation score (equation 5) by varying the number of simultaneous user preferences.

It is observed that the mean values of accuracy are consistently high. The difference between the highest and lowest mean accuracy score is as low as 0.010. The difference between the highest and lowest significance preservation score is 0.137. This is still good considering the fact that the means range from 0.86 to 1.

V. Discussion

A. Real User Feedback

In this work, we collected a real user's sleep signal and simulated their feedback to test the performance of the system. Our simulated-user was strictly compliant with the predefined or randomly chosen preference profiles. However real users might have conflicting preferences. For example, they might like an insight about sleep latency on weekdays, but at the same time might not be interested on their sleep latency on weekends. This means that the interestingness score of insights that talk about sleep latency is not 1, but a fraction of 1. There is no standard mechanism to simulating such conflicting scenarios. However, assuming multiple dimensions for the sleep profile avoids these confusions as the overlap between similar insights is reduced. Using the same example above, if we model the user preference as a two dimensional entity, we would define one interestingness profile to be a combination of sleep latency on weekdays and another to be sleep latency on weekends. Usually, random simulations have the risk of learning and unlearning within a short period of time thereby nullifying the notion of a strong user-preference. But, a few conflicting feedback can not bias the results as neural networks learn in small steps and are very robust to noisy labels [30].

B. Resource Consumption

The proposed neural models were trained and run on an NVIDIA V100 server. The final trained model has 3.1M param-eters. To update the model for one day of insights, the model takes on an average 7.9 seconds. However, this can be brought down with the help of pruning

techniques. Even otherwise, this is fast enough for a server based mobile application in which the training will be performed remotely. The proposed algorithm is light enough to be run on an edge device (mobile phones, smart watches and tablets) and is a once-a-day task.

VI. Conclusions and Future Scope

In this work, we proposed an artificial neural network model to score pre-validated insights of user behaviour from data. We consider comparative insights that talk about how a quantity differs in two different contexts. We score these insights considering the significance of the user behaviour depicted in it and the user's preference towards the insight. We used ANN to build an insight scoring model for its ability to learn, unlearn, and relearn tasks. For this, we used self-supervised training to train an ANN to perform a statistical significance test, namely the Kolmogorov-Smirnov test. Next, we augmented the architecture to learn user preference with an interactive-learning scheme. We evaluated three different model architectures of ANNs and chose the best to be our base model: a simple neural network with recurrent neural network (RNN) layers with fewer neurons. However, the other two networks: a similar RNN network with more neurons and a slightly different siamese network also exhibited satisfactory performance. Subsequently, we improved our RNN model with more and more variety of input distributions following a self-supervised learning approach. We proceeded to relearn this model to also consider user preferences in an interactive setting with the help of transfer learning.

We subsequently learn user preference on the same model using their feedback. For this, our model requires three inputs, namely, the distribution of the quantity in one context, its distribution in the other context, and a binary encoding of the insight. We froze a part of the base model and augmented it with an additional input layer that reads the binary encoding vector. We trained it on a real dataset while simulating user preferences. We came up with single and multiple user preference scenarios. The model performs well with consistently good accuracy and preserves its knowledge about statistical significance while learning interestingness. This made the network unique in an intelligent way. Also, this is the first attempt in which a single neuron is shown to play two simultaneous roles. Our evaluations suggest that the model can learn complex and dynamic user preferences. In future, we would like to perform a field testing of the proposed technique and also device ways to obtain feedback from users with the least disturbance.

Acknowledgment

This paper is an extension of our previous work presented at the IntelLang workshop at the ECAI 2020 conference [31]. In that, we considered user preferences that are simple and concerns at-most one type of insights at a time. In this paper, we extend this to two or more types of similar or different insights being preferred at the same time.

This work was supported by the Horizon H2020 Marie Skłodowska-Curie Actions Initial Training Network European Industrial Doctorates project under grant agreement No. 812882 (PhilHumans).

References

[1] M. Hingle, H. Patrick, "There are thousands of apps for that: navigating mobile technology for nutrition education and behavior," *Journal of nutrition education and behavior*, vol. 48, no. 3, pp. 213–218, 2016.

[2] J. P. Higgins, "Smartphone applications for patients' health and fitness," *The American journal of medicine*, vol. 129, no. 1, pp. 11–19, 2016.

[3] A. Härmä, R. Helaoui, "Probabilistic scoring of validated in-sights for personal health services," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–6, IEEE.

[4] S. J. O'Malley, R. T. Smith, B. H. Thomas, "Data mining office behavioural information from simple sensors.," in *AUIC*, 2012, pp. 97–98.

[5] D. Braun, E. Reiter, A. Siddharthan, "Saferdrive: An nlg-based behaviour change support system for drivers," *Natural Language Engineering*, vol. 24, no. 4, pp. 551–588, 2018.

[6] S. G. Sripada, F. Gao, "Linguistic interpretations of scuba dive computer data," in *2007 11th International Conference Information Visualization (IV'07)*, 2007, pp. 436–441, IEEE.

[7] R. Agrawal, J. C. Shafer, "Parallel mining of association rules," *IEEE Transactions on knowledge and Data Engineering*, vol. 8, no. 6, pp. 962–969, 1996.

[8] A. A. Freitas, "On rule interestingness measures," in *Research and Development in Expert Systems XV*, Springer, 1999, pp. 147–158.

[9] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, pp. 37–37, 1996.

[10] N. Sudarsanam, N. Kumar, A. Sharma, B. Ravindran, "Rate of change analysis for interestingness measures," *Knowledge and Information Systems*, pp. 1–20, 2019.

[11] H. op den Akker, M. Cabrita, R. op den Akker, V. M. Jones, H. J. Hermens, "Tailored motivational message generation: A model and practical framework for real-time physical activity coaching," *Journal of biomedical informatics*, vol. 55, pp. 104–115, 2015.

[12] E. Reiter, R. Robertson, L. M. Osman, "Lessons from a failure: Generating tailored smoking cessation letters," *Artificial Intelligence*, vol. 144, no. 1-2, pp. 41–58, 2003.

[13] A. Susaiyah, A. Härmä, E. Reiter, R. Helaoui, M. Petković, et al., "Towards a generalised framework for behaviour insight mining," in *SmartPHIL: 1st Workshop on Smart Personal Health Interfaces*, 2020, ACM.

[14] "Connected sleep and wake up light hf3670/60," Mar 2020. [Online]. Available: https://www.usa.philips.com/shop/US_Air-FryerOnly/personal-care/light-therapy/smartsleep-connected-sleep-and-wake-up-light/p/HF3670_60.

[15] K. Antczak, "Deep recurrent neural networks for ecg signal denoising," *arXiv preprint arXiv:1807.11551*, 2018.

[16] S. Kaymak, A. Helwan, D. Uzun, "Breast cancer image classification using artificial neural networks," *Procedia computer science*, vol. 120, pp. 126–131, 2017.

[17] S. Chopra, D. Yadav, A. Chopra, "Artificial neural networks based indian stock market price prediction: before and after demonetization," *J Swarm Intel Evol Comput*, vol. 8, no. 174, p. 2, 2019.

[18] K. T. Islam, G. Mujtaba, R. G. Raj, H. F. Nweke, "Handwritten digits recognition with artificial neural network," in *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*, 2017, pp. 1–4, IEEE.

[19] J. Tao, X. Fang, "Toward multi-label sentiment analysis: a transfer learning based approach," *Journal of Big Data*, vol. 7, no. 1, pp. 1–26, 2020.

[20] M. Long, Y. Cao, J. Wang, M. I. Jordan, "Learning transfer-able features with deep adaptation networks," *arXiv preprint arXiv:1502.02791*, 2015.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[22] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, 2009.

[23] B. Settles, "From theories to queries: Active learning in practice," in *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, 2011, pp. 1-18.

[24] X. Zhai, A. Oliver, A. Kolesnikov, L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE international conference on computer vision, 2019*, pp. 1476-1485.

[25] J. Devlin, M. W. Chang, K. Lee, K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[26] S. Pascual, M. Ravanelli, J. Serrà , A. Bonafonte, Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," 2019.

[27] P. Neculoiu, M. Versteegh, M. Rotaru, "Learning text similarity with siamese recurrent networks," in *Proceedings of the 1st Workshop on*

Representation Learning for NLP, 2016, pp. 148-157.

[28] S. Berlemont, G. Lefebvre, S. Duffner, C. Garcia, "Siamese neural network based similarity metric for inertial gesture classification and rejection," in *International Conference on Automatic Face and Gesture Recognition*, 2015, pp. 1–6.

[29] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, X. Chen, "A sparse auto-encoder-based deep neural network approach for induction motor faults classification," *Measurement*, vol. 89, pp. 171-178, 2016, doi: https://doi.org/10.1016/j.measurement.2016.04.007.

[30] K. Choi, G. Fazekas, K. Cho, M. Sandler, "The effects of noisy labels on deep convolutional neural networks for music tagging," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 139-149, 2018, doi: 10.1109/TETCI.2017.2771298.

[31] A. Susaiyah, A. Härmä, E. Reiter, M. Petković, "Iterative neural scoring of validated insight candidates," in *ECAI workshop on Intelligent Information Processing and Natural Language Generation*, Santiago de Compostela, Spain, Sep 2020.

Allmin Susaiyah

llmin completed his masters (by research) from the Indian Institute of Technology, Madras, and started his career as a Researcher at Philips, India in 2017. Currently, he is pursuing his doctoral studies at TU/e, Netherlands, and Philips Research, Eindhoven. His areas of interest are, artificial intelligence, insight mining, natural language generation, deep learning, and medical imaging. He is also a Marie Curie Fellow being a part of the PhilHumans consortium.

Aki Härmä

Dr. Aki Härmä did his PhD in audio and speech signal processing at HUT, Finland, in 2001. After positions at Lucent Bell Labs and HUT he joined Philips Research in 2004. He has published more than 100 conference and journal papers, numerous patents, and contributed to various product lines of Philips. He is currently at the position of Principal Scientist in the Data Science Department of Philips Research working on big data technologies, predictive analytics, and computational intelligence for automated personal and home health services. He has supervised several students at different levels.

Ehud Reiter

Ehud Reiter received the A.B. degree in mathematics and the Ph.D. degree in computer science from Harvard University, in 1982 and 1990, respectively., He is currently a Professor of computing science with the University of Aberdeen, and also a Chief Scientist with Arria NLG. He specializes in natural language generation, and has a Google Scholar h-index of 40. He also holds eight patents. Prof. Reiter is currently the Chair of ACL SIGGEN, the special interest group in (natural language) generation of the Association for Computational Linguistics.

Milan Petković

Milan Petković is a Professor at the Department of Mathematics and Computer Science, Eindhoven University of Technology. He is also the Head of the Department of Data Science, Philips Research, Eindhoven, the Netherlands. Among his research interests are information security, secure content management, privacy protection, multimedia information retrieval, and database systems.