Article

Argus: Traffic Behavior Based Prediction of Internet User Demographics through Hierarchical Neural Network

Mingda Wang , Zhining Liu, Hangyu Hu and Guangmin Hu

School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; liuzhininguestc@gmail.com (Z.L.); huhangyuuestc@gmail.com (H.H.); hgm@uestc.edu.cn (G.H.)

* Correspondence: wangmingdauestc@gmail.com

Received: 7 January 2020; Accepted: 2 February 2020; Published: 5 February 2020

Abstract: Predicting internet user demographics based on traffic behavior analysis can provide effective clues for the decision making of network administrators. Nonetheless, most of the existing researches overly rely on hand-crafted features, and they also suffer from the shallowness of information mining and the limitation in prediction targets. This paper proposes Argus, a hierarchical neural network solution to the prediction of Internet user demographics through traffic analysis. Argus is a hierarchical neural-network structure composed of an autoencoder for embedding and a fully-connected net for prediction. In the embedding layer, the high-level features of the input data are learned, with a customized regularization method to enforce their discriminative power. In the classification layer, the embeddings are converted into the label predictions of the sample. An integrated loss function is provided to Argus for end-to-end learning and architecture control. Argus has exhibited promising performances in experiments based on real-world dataset, where most of the metrics outperform those achieved by common machine learning techniques on multiple prediction targets. Further experiments reveal that the integrated loss function is capable of promoting Argus performance, and the contribution of a specific loss component during the training process is validated. Empirical settings for hyper parameters are given according to the experiments.

Keywords: user demographics prediction; traffic behavioral analysis; neural network; hierarchical structure; integrated loss function

1. Introduction

The Internet is unquestionably a colossal information bank nowadays, where a gargantuan amount of mankind's information is deposited with no aspect of human life spared. The omnipresent information of internet users (abbreviated as "user" for the rest of this paper) is so strikingly copious, that information miners such as Internet service providers (ISPs) are forced to leverage advanced mining techniques to obtain refined user information for better quality of service (QoS).

Being one kind of the very fundamental user information, demographics (such as gender, age, education and so forth) is capable of profiling a user with a high precision, which makes its mining very rewarding. The non-stop development of data mining techniques has already made such mining viable and effective. Provided with the dataset containing the trace of usage (such as website access logs, network traffic and so on), proper data mining techniques can infer the demographics of the user to a certain degree, which has motivated a wide range of research in search of better mining techniques for user behavioral information mining.

Among the information mining techniques, traffic behavioral based prediction of Internet user demographics (TPID) leverage the analysis of captured network traffic data to build up mapping

rules between traffic data representations and demographic label predictions. Flow connection based methods [1] are light-weighted, and their features are relatively easy to calculate. Statistical methods [2–6] outstand in scalability, and they are more versatile in the choice of features.

There are three major challenges existing in the research field of TPID. To begin with, the methods mentioned above are suggested to overly rely on hand-crafted features, whose discriminative power plays a significant role in their final performance. Besides, most of the manually-designed features are relatively light-weighted, where the higher-level information is not included. Moreover, the inspected prediction targets are quite limited to targets such as gender, education or a specifically defined anomaly.

Motivated by the challenges mentioned above, we devote our effort to building up a framework that is not only free of hand crafted features, but it is also able to mine the higher-level information. Meanwhile, this framework should perform robustly across different target prediction tasks, in order to address the existing challenges in TPID. Thus, in this paper we propose Argus, a hierarchical neural-network framework for the traffic-analysis-based prediction of Internet user demographics. To the best of our knowledge, Argus is the first neural-network-based framework proposed specifically to counter the aforementioned problems in TPID.

Argus features an end-to-end fashion for the network training process. On the level of information mining, Argus provides a hierarchical structure for learning to enforce a relatively comprehensive understanding of the latent distribution of the data. However, Argus has shown a competitive performance in the predictions of various kinds of target labels in comparison with some other prominent methods, which indicates the robustness in different-category prediction missions.

The hierarchical structure of Argus is composed of an autoencoder and a fully-connected net. The autoencoder learns the dense embedding of the input to reduce the high sparsity and the high dimensionality for classification. The fully-connected net converts the embeddings into the corresponding label predictions. To boot, an integrated loss function is designed to guarantee end-to-end learning of the overall structure, while at the same time it introduces controllability in the structure of Argus during training. The contributions of our work can be summarized as follows:

- This paper introduces a neural-network-based solution to meet the challenges posed by TPID specifically, which to the best of our knowledge is the first time such an attempt has been made.
- We propose Argus, an accordingly designed neural network structure to address the TPID problems. Argus has a hierarchical structure composed of an embedding layer and a fully-connected net. Argus does not gamble on hand-crafted features, and it guarantees a comprehensive mining of user information through a pipeline of embedding and classification processes.
- An integrated loss function is designed to guarantee an end-to-end training of Argus, which at the same time introduces controllability into the structure of Argus. The structure of Argus can be modified via emphasizing or depreciation of the components of the loss function, where Argus could transform into a vanilla autoencoder classifier, a pure fully-connected net or a hybrid network in between, controllably, according to demands.
- All the claims about the architecture and loss function of Argus are validated through experiments based on real-world datasets. Multiple prediction targets were experimented on, and Argus has achieved promising results in comparison with some prevailing prediction algorithms. After this, more experiments were carried out to validate the contribution of the integrated loss function and its components, and the hyper-parameter settings to obtain the best performance were given empirically.

The rest of this paper is organized as follows: Section 2 provides an overview of related works and the existing challenges. Section 3 gives a thorough explanation of the methodology. Section 4 introduces the experiment configurations, the experimental results and corresponding analysis. Section 5 summarizes this paper and gives a conclusion. At the end of this paper, grants are listed, and special thanks are given.

2. Related Works

Internet user behavior [2,6–12] has long been talked about and researched. Major aspects of this area include behavior modeling in the cloud [7,8], in online social networks (OSNs) [9,10], in network traffic [2,6] and so forth. Ning Xia [2] developed an integrated framework to quantify user behaviors, which leverages information from both OSN pages and traffic. Besides, in this paper the author has proven that information extracted from traffic is more detailed and dynamic than that drawn from static web pages.

Alongside the unstoppable development of the Internet, user classification has emerged and drawn a substantial amount of attention from both industry and academia. Existing research grabs all kinds of information sources to achieve accurate user classification. Some research uses the visit logs gathered from the website server to do classification [13,14], while others do analysis based on online social network (OSN) information from network crawlers or application programming interfaces (API) of the OSN itself [15–19]. Marco Pennacchiotti [15] has proposed a general framework for Twitter user classification. This method leverages four different kinds of features to fully depict a user.

Traffic-analysis-based user classification extracts user information from the internet traffic, and it builds detailed user profiles for classification [1–6]. Thomas Karagiannis [1] built a graphlet-based method to quantify user communication behaviors for user profiling and classification. Huaxin Li [6] captured the Wi-Fi access traffic in a campus network, then he applied several machine learning techniques to classify a user's demographics, such as gender and education. However, all the research mentioned in this paragraph above suffers from the problem of shallowness during mining, and the classification categories are relatively limited.

Early in 1999, Murray [11] proposed a demographics inference method based on latent semantic analysis (LSA). The data for use is constructed by the search term data and the webpage access data. The vector representation of the user is constructed by vectorization based on statistics and singular component analysis (PCA). After that, a 3-layered neural net based on a scaled conjugate gradient (SCG) is applied to generate the label prediction of the user. In 2019, Wu [12] proposed a user demographic inference method based on a hierarchical encoder network with attention (HURA). The information source is the search queries the user generates. HURA is constructed by three major parts: a word encoder, a query encoder and a predictor, where the word encoder and the query encoder jointly learn an integrated user representation for latter classification. An attention mechanism is applied to both the word-level learning and the query-level learning. As far as we are concerned, there are four major aspects that make Argus differ from the work of Murray [11] and that of Wu [12], which are stated as follows:

- The first aspect is the data for use. Argus concerns solely the information extracted from network traffic, while both the works of Murray and Wu rely on the query information the user generates. It is commonly known that traffic data is always more noisy than plain text query data because of the existence of encryption, sampling during capture and so on, which makes the demographic inference based on traffic analysis much more challenging.
- The second difference is in the network structure. Regarding the work of Murray [11], a simple, yet effective, structure with a 3-layered feed forward network is adopted. While for Wu, HURA is based on a more sophisticated network with multiple representation learning modules stacked together. In contrast, Argus harbors a hierarchical structure with an embedding layer and a classification layer, where the embedding is delivered through an autoencoder, and the prediction is fulfilled by a fully-connected network.
- The third difference is in the loss function for use. Argus has a specifically designed loss function for back propagation, where three different loss terms are combined to guarantee a thorough and interpretable learning both in the embedding and the prediction. Moreover, this integrated function is able to control the architecture of the whole network by rotating on the controller coefficients, which grants Argus much flexibility in coping with different tasks.

- The final difference concerns the predicted target. Argus is tested in multiple experiments concerning seven different targets, and the sub-categories in these prediction targets vary from binary to multi-class. While for Murray's work, all the prediction targets are binary. Regarding Wu's work, only age and gender are considered prediction targets.

Recent years have witnessed an explosive increase in researches on neural networks and their applications. While classical network structures such as convolution neural network (CNN) [20] and recurrent neural net (RNN) [21] have been widely implemented in all kinds of scenarios, the state-of-the-art of neural network research keeps evolving rapidly. Generative adversarial networks (GAN) [22–24], residual nets [25–27], graph embeddings [28] and so forth have already pushed the frontier to a much deeper level.

Autoencoder (AE) [29–31] has been prevailing in recent years, since it is competent for various kinds of tasks such as objective generating, signal de-noising, content embedding and so forth. As an evolved form of AE, variational autoencoder (VAE) [30] is still a favored technique for modeling, where the network tries to learn a distribution from the probabilistic models instead of learning just the transformation function from input to bottleneck layer. A number of exploratory studies have been carried out to discover the potential of VAE. Higgins [31] has proposed the β -VAE, where the network learns the disentangled factors by introducing a hyper parameter β to emphasize the latent bottleneck information.

Our research tries to address the specific traffic analysis based internet user classification problem by designing a layered neural network structure on-demand and customizing the loss function to guarantee an end-to-end training. To the best of our knowledge this is the first time an NN-based solution has been proposed for the traffic analysis based Internet user classification problem.

3. Methodology Explanation

3.1. Model Viability

Web service is one of the most-commonly consumed content services provided from various sources on the Internet, and it has evolved in such a proliferative way where the user experience it pinpoints is very refined. For example, years ago there were very few choices for online shopping such as "www.amazon.com", and the online commodities were relatively limited. Nowadays most of our online needs have their own specific websites to visit, for example we can purchase jewelry from "www.pandora.net", or we can buy liquor from "www.thewhiskyexchange.com". Provided with so many choices online, a user is prone to visit different websites to meet their own specific demands.

We argue that the true catalyst that triggers the proliferation of a user's website visits is actually the uniqueness of the user as an individual human being. As a strong indicator of the user's identity, demographics can also have a discoverable correlation with the website-visiting behavior. Users with the same demographics may have similar website-visiting behavior. For instance, some common avocations for men such as cars and basketball games may lead to dominance in the visits of websites which provide such content, as is illustrated in Figure 1. On the contrary, users with different demographics generally tend to have somewhat different website-visiting behaviors, which can serve as informative clues to infer the identity of the user himself/herself. In the example shown in Figure 1, female users generally care about other things, thus their visits to websites will be reasonably different from those of male users.

Figure 1. A general example to explain that user demographics may exert influence on visits to websites.

There are many ways to model user website-visiting behavior. While simple statistics such as the frequency of a user's visits to a specific website can be quite informative, the sequential information concealed in a series of websites can also be crucial in user profiling. In this paper we take both the quantitative information and the sequential information into consideration, through modeling user behavior into a sequence of visited host names. The detailed process is narrated in the following subsection.

3.2. Preprocessing and Data Representation

3.2.1. Sequence Generation

As is discussed above, in this research we leverage website information to achieve user demographics prediction. The specific website information we leverage is the host name which the user requests during the establishment of any web session. It is worth noting that there is not a strict one-to-one mapping between host name and website, since one click to a webpage nowadays may trigger multiple requests for a series of host names due to the loading of different webpage content [32]. Still we argue that despite this many-to-one mapping from host name to website, the host names generated from different websites will have substantial differences between each other so that TPID could work. Furthermore, this assumption is validated later in the experiments.

In this paper, we consider host names extracted from three kinds of the most common protocols used for web services: the Hyper Text Transfer Protocol (HTTP), the Hyper Text Transfer Protocol over Secure Socket Layer (HTTPS) and the Domain Name System (DNS). Since HTTP harbors a plain text transferring fashion, the host name string loaded inside the packet header of an HTTP request is easy to extract. For the HTTPS case, the server name from the client hello packet of the handshake stage is extracted. While for DNS, we filter out the host name from the DNS queries.

After the host name extraction, all the host names are put sequentially into a vector according to the capture time of the packets. Thus, we can get a long sequence with hundreds of thousands of requested host names for each user in each monitoring period, and this process is shown in Figure 2. Nonetheless, this long sequence is not supposed to be directly fed into the network for classification. One reason is that the host names are actually categorical features, to which the numerical calculations like additions or multiplications can not directly apply. The second reason is that we expect the samples to be more temporally fragmented, since practically, the monitoring of certain users cannot always be long enough to make a temporally coarse-grained analysis work. The third reason is that with

such a long sequence being the input, the neural nets might be overwhelmed and thus generate a poor performance.

Figure 2. An illustration on the mapping from real-life user activity to host name sequence.

3.2.2. Sequence Reshaping

To cope with the network's input requirements, three more operations are added to the preprocessing of the traffic dataset. The first operation is to filter out the noise inside the sequence. In this research, we argue that those host names that appear more frequently are more likely to have useful information for classification, and those with lower appearance frequencies are relatively more noisy. Because the host names with a lower frequency are quite possibly introduced by rare behaviors of the inspected user, which may cause turbulence in the performance of the classification based on usual behavior analysis. Thus, those host names in the sequence with too few appearances are deleted, and this de-noised sequence after the filtering will participate in the further analysis.

After the filtering, the de-noised sequence is still too lengthy, and we expect to break it down to temporally finer grained small sequences, as is stated above. To do so, a threshold-based segmentation scheme is applied to the lengthy sequence. Giving a timeout threshold Δ , if the capture time interval between two consecutive host names exceeds the threshold, the sequence will be cut between these two host names, and thus two shorter sequences are generated, as is illustrated in Figure 3. The last operation in reshaping is to convert the input sequences into the same length. Sequences that are too long are chopped off from the tail and sequences too short are discarded, as is shown in Figure 3.

Figure 3. The reshaping process to generate eligible input sequence.

3.2.3. One-Hot Encoding

The pre-cut smaller sequences are still not qualified for the latter neural network analysis, since each term inside a sequence is still a categorical feature where numeric calculations cannot be directly applied. In this research, we leverage the one-hot encoding [33] technique to convert the categorical sequence into a numeric one. One-hot encoding is the operation that converts the categorical sequence into a sparse numerical sequence which contains only ones and zeros. Each term inside the categorical sequence is replaced with a new sequence, where the index of this host name in the appearance frequency sorting is also the index of the number "1", while all the other terms are zeros. For example, the second most frequent domain name is replaced with a binary sequence for which the second term is one and all the other terms are zeros. After one-hot encoding, the final data representation is reached and ready for the neural net to analyze. The complete process of the one-hot encoding of this research is shown in Figure 4.

Figure 4. The one-hot encoding process from the original host name sequence to the one-hot-encoded sparse sequence.

3.3. Argus Structure

As mentioned earlier, Argus has a hierarchical structure, which is shown in Figure 5. Argus is composed of two layers. One is the embedding layer, and the other is the classification layer.

3.3.1. Embedding Layer

The first layer is the embedding layer, which consists of an autoencoder. From Figure 5 we can see that the autoencoder has a cascading structure with multiple sub-layers. The Input Layer takes in a raw data vector and pass it to its following layers with descending neuron counts, where the dimensionality of the data vector is guaranteed to reduce. When the data vector reaches the bottleneck layer, the dimensionality of it also reaches its minimum (in the embedding layer), thus the most dense embedding of the original data vector is generated. After this, the data embedding will go through an inverse process to reconstruct the input data vector by passing through a series of layers with ascending neuron counts. A reconstruction error between the original data vector and the reconstructed input will be calculated and optimized to ensure that the embedding learned from the data input contains the majority of the original data information. The detailed calculation of this error will be stated later.

Figure 5. The hierarchical structure of Argus.

The purpose of the auto-encoder layer is to embed the sparse input vector into a dense space, where the redundant information hiding in the high dimensionality could be shred and the burden of the following processing is thus alleviated. Meanwhile, a customized regularization method is introduced to guarantee that the embeddings could contain as much discriminative information as possible, which further paves the road to a better performance in the latter classification process. This autoencoder is supervised since the training process takes the label information of the samples as part of the input.

3.3.2. Classification Layer

The second module of the structure is the classification layer. After obtaining the sample embeddings from the previous layer, a fully-connected net takes the embeddings as input, and it generates the final predictions of the sample as output. A softmax layer is appended to the hidden layers of the fully-connected net to fulfill the prediction. Although the neural net in the classification layer is simple in structure, it is necessary, since in our design the embedding layer is not supposed to generate the prediction labels directly. The major reason is that an autoencoder is a dimensionality reducer rather than a predictor, since its primary goal is to retain the most informative data for input reconstruction. Notwithstanding, the fact that whether this data is useful or not for prediction is not within the concern of an autoencoder. Thus, a classification layer is needed to convert the embedding to the predicted label, which results in the current architecture of Argus.

3.4. Forward Process

An autoencoder is a butterfly shaped neural network which is composed of two parts: the encoder network and the decoder network. The encoder has a descending neuron count as the layer goes deeper, which enforces the network to cut out useless information while retaining the core information of the input. The following decoder tries to reconstruct the input with a structure which is symmetrical to that of the encoder. Denoting the input host name sequence as $S \in \mathcal{R}^n$, where n refers to the length of the sequence. Suppose that the encoding part of the embedding layer is $\Psi: \mathcal{R}^n \rightarrow \mathcal{R}^m$, therein the generated embedding is denoted as $E \in \mathcal{R}^m$ with an embedding length m (also known as the neuron count of the bottle neck layer), thus we have the equation for the unsupervised encoding process as below:

$$E = \Psi(S). \quad (1)$$

The embedding E is then fed into the decoder for the reconstruction of S , where the decoder layer is denoted as $\Omega : \mathcal{R}^m \rightarrow \mathcal{R}^n$. So we can have the equation below:

$$\hat{S} = \Omega(E), \quad (2)$$

where $\hat{S}$ is the reconstructed input. Combining Equation (1) and Equation (2) we can reach:

$$\hat{S} = \Omega(\Psi(S)). \quad (3)$$

After obtaining E and $\hat{S}$, the forward pass of the embedding layer is finished. E will be fed into the fully-connected net to continue the forward process, while both E and $\hat{S}$ will be retained for back propagation after the forward process ends.

In our design, the autoencoder fulfills the task of embedding extraction, while the classification takes place with the fully-connected net inside the classification layer. This network takes the embeddings learned from the autoencoder as input, and it converts the embeddings into their corresponding label predictions as output. Denoting the one-hot representation of the ground-truth label as C , and the category count is denoted as c , the fully-connected net is equivalent to a function $\Lambda : \mathcal{R}^m \rightarrow \mathcal{R}^c$ which converts E into the label prediction $\tilde{C}$, as is shown in the equation below:

$$\tilde{C} = \Lambda(E). \quad (4)$$

Combining Equation (1) and Equation (4) we can get:

$$\tilde{C} = \Lambda(\Psi(S)). \quad (5)$$

The forward process is finished once $\tilde{C}$ is reached. For network training, the back propagation process will be launched henceforth, while for testing the whole process is closed and $\tilde{C}$ will be treated as the final prediction of the testing samples to participate in the evaluation. The back propagation process will be elaborated in the following subsection.

3.5. Back Propagation

Back propagation is the process to update the network parameters according to the gradient of the loss function, so that Equation (5) could approximate the real mapping function between S and C . One of the key challenges in any back propagation schematic is the designing of a proper loss function to provide the network with sufficient a priori information for better optimization. In our research we consider three kinds of losses to fully capture the a priori, while guaranteeing an end-to-end learning of Argus: the mean-square error (MSE), the triplet loss and the cross entropy.

3.5.1. Mean-Square Error

There are two losses derived for the embedding layer as the building blocks of the overall loss function. The first one is the mean-square error between S and $\hat{S}$, denoted as $\Gamma : \mathcal{R}^n \times \mathcal{R}^n \rightarrow \mathcal{R}$. The MSE is a loss term to enforce the autoencoder to keep the most important information when learning the embedding E and concurrently to ditch useless information as much as possible. The MSE is the arithmetic mean of the squared l_2 norm for the difference between $\hat{S}$ and S , which is formulated as Equation (6).

$$\Gamma(S, \hat{S}) = \frac{1}{n} \|(\hat{S} - S)\|_2^2. \quad (6)$$

3.5.2. Triplet Loss

The second loss is the term that introduces the label information into the training process. The label information is able to provide more clues for the back propagation process by compelling

the autoencoder to maximally keep the categorical information while shredding out all the irrelevant residuals during training. The conception behind the triplet loss is to measure the difference between the intra-most distance and the inter-most distance, where the intra-most distance refers to the largest distance between two points who belong to the same cluster, and the inter-most distance refers to the largest distance between two points who belongs to two different clusters. Thus we can infer that, the larger the triplet loss, the farther the clusters are, and thus the easier the classification is. On the contrary, a small triplet loss indicate that the clusters are close to each other, which requires more effort from the classifier to separate these clusters. The specific term leveraged in this paper is the batch-hard triplet loss.

The triplet loss is computed based on E and C . Giving the total count of embeddings inside this batch as K , firstly we specify an anchor sample embedding $e_a \in E$ and the embedding set share the same label with e_a as E_a , where apparently $e_a \in E_a$. Denoting the function to compute the distance between two different embeddings as $D : \mathcal{R}^m \times \mathcal{R}^m \rightarrow \mathcal{R}$. Then, we find the most distant embedding which belongs to the same category (a.k.a. the hardest positive embedding) with e_a , where this sample is denoted as $e_+ \in E_a$, and the distance between e_a and e_+ can be denoted as d_{a+} (a.k.a. the hardest anchor-positive distance).

$$d_{a+} = \max_{e_x \in E_a} (D(e_a, e_x)). \quad (7)$$

Similarly, we find the closest embedding which belongs to a different category (the hardest negative sample) from e_a , where this sample is denoted as $e_- \in E$ and the hardest anchor-negative distance as d_{a-} :

$$d_{a-} = \min_{e_y \in E \setminus E_a} (\{D(e_a, e_y)\}). \quad (8)$$

The hardest triplet distance for e_a can be computed according to the equation below:

$$\theta_a = [d_{a+} - d_{a-} + \epsilon]_+, \quad (9)$$

where ϵ is the margin for preventing the network to output trivial solutions. Let us substitute Equation (7) and Equation (8) into Equation (9) and we can reach:

$$\theta_a = [\max_{e_x \in E_a} (D(e_a, e_x)) - \min_{e_y \in E \setminus E_a} (\{D(e_a, e_y)\}) + \epsilon]_+. \quad (10)$$

Let us compute the hardest triplet distances for all possible anchor embeddings, and the batch-hard triplet loss, denoted as $\Theta : \mathcal{R}^m \times \mathcal{R} \rightarrow \mathcal{R}$, can thus be reached as is shown in the equation below:

$$\Theta(E, C) = \sum_{e_a \in E} [\max_{e_x \in E_a} (D(e_a, e_x)) - \min_{e_y \in E \setminus E_a} (\{D(e_a, e_y)\}) + \epsilon]_+. \quad (11)$$

3.5.3. Cross Entropy

There is one loss term drawn from the fully-connected to build up the overall loss function. This term is the cross entropy between C and $\tilde{C}$. Cross entropy loss is a commonly used loss metric for classification and prediction tasks, which can effectively quantify the difference between the predicted labels and the ground truth labels. Let us denote the cross entropy as $H : \mathcal{R}^c \times \mathcal{R}^c \rightarrow \mathcal{R}$, and it is defined in Equation (12).

$$H(C, \tilde{C}) = - \sum_{i=1}^K p_i \log(\tilde{p}_i), \quad (12)$$

where p_i refers to the ground truth probability of the i -th sample s_i , and $\tilde{p}_i$ refers to the predicted probability of s_i .

3.5.4. Integrated Loss Function

As is stated above, it takes three parts to assemble the overall loss function of Argus: the MSE $\Gamma(S, \hat{S})$, the triplet loss $\Theta(E, C)$ and the cross entropy $H(\tilde{C}, C)$. The overall loss function is the weighted sum of these three loss terms, as is defined in the equation below:

$$L(S, \hat{S}, E, \tilde{C}, C) = \alpha\Gamma(S, \hat{S}) + \beta\Theta(E, C) + H(C, \tilde{C}), \quad (13)$$

where α and β are the weights for the MSE loss and the triplet loss respectively. Empirically the cross entropy loss is a baseline term for such a prediction problem, hence we set the coefficient of the cross entropy term to be the constant 1. Notably Argus will degrade into a pure fully-connected net at $\alpha = 0$, where the decoder part of the autoencoder was screened out for the training process. Similarly, when $\beta = 0$, the neural network degrades into an autoencoder classifier because the triplet loss is blacklisted. Scaling α and β will put corresponding emphasis on parts of Argus structure, which needs further hyper-parameter tuning to find out the best settings for a specific prediction task. A complete process of the Argus training algorithm is shown in Algorithm 1.

Algorithm 1 Argus Training Algorithm

```

1: procedure ARGUS TRAINING
2:    $i \leftarrow 0$ 
3:   while  $i \leq \text{max\_epoch}$  do ▷ Epoch rotation.
4:      $j \leftarrow 0$ 
5:     while  $j \leq \text{max\_iter}$  do ▷ Iteration rotation.
6:        $E_{i,j} \leftarrow \Psi(S_{i,j})$ 
7:        $\hat{S}_{i,j} \leftarrow \Omega(E_{i,j})$ 
8:        $\tilde{C}_{i,j} \leftarrow \Lambda(E_{i,j})$ 
9:        $L_{i,j} \leftarrow \alpha\Gamma(S_{i,j}, \hat{S}_{i,j}) + \beta\Theta(E_{i,j}, C_{i,j}) + H(\tilde{C}_{i,j}, C_{i,j})$ 
10:       $\Psi \leftarrow l_r \nabla L_{i,j}(\Psi)$  ▷ Update network parameters.
11:       $\Omega \leftarrow l_r \nabla L_{i,j}(\Omega)$  ▷  $l_r$  refers to the learning rate.
12:       $\Lambda \leftarrow l_r \nabla L_{i,j}(\Lambda)$ 
13:       $j \leftarrow j + 1$ 
14:     $i \leftarrow i + 1$ 
15:     $S \leftarrow \text{shuffle}(S)$ 
16:  return  $\Psi, \Omega, \Lambda$ 

```

4. Experiment and Analysis

4.1. Dataset and Configurations

The dataset we use for the experiment was captured from the gateway router located at the Laboratory of Complex and Multi-dimensional Signal Processing in University of Electronic Science and Technology of China. There were 60 users doing their daily jobs inside the network, and during the surveillance their devices were kept unchanged. The surveillance continued for over two months, and the overall captured traffic volume exceeded two terabytes. Figure 6 logically exemplifies the environment of the traffic capture.

The specific back propagation optimizer we used during our experiments was the adaptive moment estimation (Adam). Adam is considered an evolved version of the root mean square propagation (RMSprop), which applies the running average operation to not only the gradients themselves, but also the second moments of the gradients. This feature grants Adam better adaptivity over other prevailing optimizers such as the stochastic gradient descent (SGD), the adaptive gradient algorithm (AdaGrad) and RMSprop. Thus, we choose Adam as the optimizer for the back propagation of Argus.

The original dataset was split into three parts: the training set, the validation set and the testing set, where the volume ratio was 6 : 2 : 2, respectively. This ratio is an empirical setting widely adopted in various machine learning studies [6,7,9,10], where the training, validation and testing are believed to be most balanced empirically. For the label category, we considered a wide variety of labels whose ground truth was drawn from the laboratorial profiles of the users. The label categories and their corresponding explanations are shown in Table 1.

Figure 6. A logical illustration of the traffic capture environment.

Table 1. Prediction targets and their respective explanations. The abbreviation “P.o.B.” is for “Place of Birth”, “D.o.B.” for “Date of Birth” and “R.F.” for “Research Field”. The sharp symbol “#” in the header line means the number of different categories.

Target	Explanation	#	Example
Gender	The gender of the user.	2	male
Grade	The grade of the user as a student.	4	freshman
Education	The user’s latest academic degree.	2	master
P.o.B.	The user’s province of birth.	13	Szechuan
D.o.B.	The user’s month of birth.	24	06/1990
Advisor	The advisor of the user as a student.	6	Hu
R.F.	The research field of the user.	5	topology

To fully exhibit the performance of Argus, four prevailing machine learning algorithms were also tested, including the support vector machine (SVM), the random forest (RDF), the Gaussian naive Bayes (GNB) and the logistic regression (LGR). Four metrics were adopted for performance evaluation: accuracy, precision, recall and F1 score.

4.2. Experiments and Analysis

4.2.1. Weight Rotation Experiment

The first question to ask about Argus should be: what is the best achievable performance of Argus and its corresponding hyper-parameter setting? In the initial experiment we focused on finding the best pair of weights for MSE and triplet loss respectively, where a set of grid searches for the weights α and β were performed with all other hyper parameters fixed; the results are shown in Figure 7. Considering the layout for these heatmaps, the detailed information of gender prediction is

shown in the largest heatmap located at the far left of Figure 7, while the others are shown in smaller heatmaps and the detailed information is omitted. Notice that all four heatmaps in Figure 7 show lower accuracies where $\alpha = 0$ and $\beta = 0$ than those from most of the other areas, which indicates that MSE and triplet loss are capable of boosting the accuracy of Argus.

Figure 7. The accuracy heatmaps of all prediction targets. Noting the results for gender prediction as an example, and the heatmaps for the others are generated in the same way as with the genders, while the detailed information is omitted. The number in the title of a small heatmap represents the best accuracy. The abbreviation “P.o.B.” is short for “Place of Birth”, “D.o.B” for “Date of Birth” and “R.F.” for “Research Field”.

Argus might require different weight settings for best accuracies across different prediction targets, according to Figure 7. We can see that for the gender prediction (shown in the largest heatmap in Figure 7), the increase of weights introduced an ascending tendency in accuracy, where the larger the α and β , the higher the accuracy in most cases. However this proportionality does not hold for all the prediction targets. For the D.o.B. (as well as P.o.B.) prediction, the results may fluctuate across the search space. For the R.F. prediction, MSE and triplet loss require relatively small weights to achieve a high accuracy. In other cases, such as the prediction of education, the triplet loss shows no obvious impact on the accuracy as β continues to increase, while for higher α there is a trend for higher accuracy.

We argue that this instability of best weight settings across different prediction targets are introduced by the nature of the prediction task itself. Loss terms are basically regularization methods with prior information which are capable of narrowing down the solution space for optimization, so that the process could speed up and be less likely to be trapped inside the local minima. While different prediction targets have different solution spaces, and the requirements for the optimization to reach the minimal loss will differ naturally. Thus as the controllers of the loss terms, it is reasonable for α and β to have different best settings across different prediction targets.

To further exhibit the contribution of triplet loss, the scatter plot of the embeddings (for the gender prediction) in the space of their first and second principal components are illustrated in Figure 8. In this figure, what concerns us the most is the distinctiveness between samples from different categories, namely how clear the border between categories are. The plot on the left shows that the male user samples and female user samples mingle with each other and there is no obvious border between them. Adding triplet loss into the analysis, as shown in the middle plot, can significantly improve the resolution between genders. From this middle plot we can see that samples from the same category tend to form a more compact cluster within, and the border between categories is much clearer. Such resolution improvement continues as β increases, as is shown in the sub-figure located at the right side. This gives a more intuitive validation of the power of triplet loss, as well as the integrated loss function.

Figure 8. The scatter plot of the training samples on the space of principal component analysis (PCA) components. The horizontal axis refers to the largest principal component of the corresponding embeddings of the training samples, and the vertical axis refers to the second largest principal component. The yellow spots in the plot represent the male users, and the dark purple spots represent the female users. Three sub-figures share the same parameter settings except the values of β .

4.2.2. Sequence-Split-Rotation Experiment

In this experiment we tried to find the best setting for the variances which control the generation of samples. There are two major factors to control the sample generation: the time interval threshold Δ to split samples and the maximal length n of sequence. The larger the Δ , the fewer the time intervals which satisfy such a requirement, thus the fewer the cutting points. Consequently, the sample sequences before reshaping will grow longer, and the sample count will decrease. Since Argus is a neural network based method, we suggest that a relatively large amount of samples should be retained to guarantee the sufficient training of Argus, which also means a relatively small Δ should be provided. On the contrary, if Δ is overly small, it may dilute the integrity of the host name sequence by breaking it into trivial snapshot where the characteristics of the user cannot be captured.

As for the reshaping of the split sequence, the maximal sequence length n is also not supposed to be too big or too small. If n is overly small the sequence may not be able to capture enough user behavioral information since there are many host names discarded. While if n is excessively big, many samples will be discarded because they do not have that many host names in sequence. To find the best setting of Δ and n , we rotate on these factors and plotted the curves for accuracy and sample count in Figure 9. Both the sub-figures consider the prediction of the user's gender.

Figure 9. The rotation experiment for time interval threshold (left) and maximal sequence length (right).

From Figure 9 we can see that, as the rotation parameters increase, both sample count curves decrease as expected. Nonetheless, the accuracy curves exhibit different patterns in two plots: the accuracy for the time interval threshold decreases at first yet starts to increase after a certain value is exceeded, while for the maximal sequence length the accuracy curve has an ascending-to-almost-descending pattern. However we argue that the cause for such patterns may be introduced by the change of input data conditions across each rotation, where the input data changes because the split sample is different from the previous, making a comparison between different rotations.

Some empirical conclusions can still be drawn here. From the left of the sub-figure of Figure 9 we can see that Argus can achieve high accuracies with both small Δ and large Δ , yet the smaller one is preferred since we do not want the samples to be too few, as stated previously. The right sub-figure tells us that there does exist a maximum in our grid search; however, the accuracy differences between consecutive points are not substantial (roughly less than 0.03). Thus for n we also suggest a relatively small value to choose for better traffic utilization.

4.2.3. Comparison Experiments between Methods

In this subsection we compare Argus between different methods, which include SVM, RDF, GNB and LGR. The metrics for use are accuracy, precision, recall and F1 score, and the prediction targets are listed in Table 1. The results are shown in Figure 10.

Figure 10. Bar plots for four matrices versus fourteen different prediction targets. All four subplots share the same legend.

From this figure we can see that Argus has an overall better performance than those of other methods, where most of the metrics are higher than those achieved by other methods. However, two defeats of Argus were spotted during the precision evaluation, where Argus takes the second place in the prediction of place of birth, and the lowest in the prediction of date of birth. Yet the precision gaps between Argus and other methods are quite trivial. Moreover, we notice that both the prediction of date of birth and that of place of birth have relatively low metric scores across all methods and all metrics. These results are within expectation because these two kinds of demographics have a relatively lower impact on website visiting behavior.

4.2.4. Latent Connection Discovery

Although Argus has achieved promising results in various experiments, we argue that the results do not only serve as strong evidence for the capability of Argus in TPID tasks, but they also shed light upon the possibility for Argus to sniff out latent connections between users.

For the experiment in the classification of P.o.B., we find the classification results surprising and counter intuitive. It is our prior conjecture that P.o.B. should not have much influence on website visiting behaviors among the students, since most of the students enrolled in our lab do not usually exhibit regionally specific behaviors during work. Nevertheless, the classification results shows otherwise: an accuracy above 0.4 was achieved, which is far better than random guessing. This

phenomenon gives a hint of a latent connection between the students, so we conducted an additional investigation on this matter by personally interviewing each relevant student.

Interestingly, the results of the investigation show that students from regionally close areas do have more similar website visiting behaviors than those not. The students from regionally close areas tend to share the same dormitory, and they would have somehow more interactions internally than externally in daily life. For instance, all three students from Hunan live in the same dormitory. They have developed a common habit, which is watching NBA games, and they would sometimes do so during the break hours in the lab. While some students from other provinces also watch NBA games, they do it less frequently and have shorter durations watching. Besides, these three students would tend to search for similar materials for study online, which brings their webpage visiting behaviors closer. Further inspection into the detailed classification results show that these students are indeed classified as being from Hunan. Thus, the investigation results have alleviated our confusion.

4.3. Further Discussion on Experiments

All the experiments aim to prove the capability and superiority of Argus in TPID tasks. However, it is not our ultimate aim to show how accurate Argus is when classifying user demographics. Argus is a framework designed to predict user demographics in order to provide information building blocks for accurate user profiling, and thus we can pave the road to a faster and more pinpoint operation for network management.

As the cornerstone for such a management operation, Argus has furthermore shown capability in the discovery of latent connections between users. This feature bestows Argus with viability in the mining of covert user attributes and behaviors such as finding user needs for better advertisements, or even digging out criminal groups and their actions hiding under calm web visits. Further research covering such features of Argus is outside the scope of this paper, and they will surely be done in the future.

5. Conclusions

Argus is a hierarchical neural network structure designed to address various existing problems in current TPID. Argus takes the host name sequences as input, with no further hand crafted feature extraction attached. In the embedding layer the input samples are first converted into embeddings which entrap the discriminative information for prediction, while maximally cutting out the noise. The classification layer then generates label predictions from the samples. An integrated loss function is designed to guarantee the end-to-end training of Argus, while at the same time it can introduce controllability into the structure of Argus and promote the final performance.

In this paper, Argus is tested in a series of experiments based on a real-world dataset. In the predictions of 16 totally different targets of demographics, hobbies and coding habits, Argus was tested under four general evaluation metrics where the results were compared with those of SVM, RDE, GNB and LGR. The results show that Argus is capable of achieving a better performance under a large portion of the prediction targets, with an average accuracy of over 80%. The best settings for the hyper parameters vary across prediction targets, and the empirically best settings are given. The specifically designed loss function is proven by experiment to be useful in speeding up the convergence, promoting the accuracy and augmenting the controllability of the structure. Besides, emphasizing the triplet loss will effectively boost the categorical resolution of the learned embeddings, which further validates the contribution of triplet loss in the prediction of user demographics.

Author Contributions: Conceptualization, M.W.; methodology, M.W. and Z.L.; software, M.W.; validation, M.W. and Z.L.; formal analysis, M.W.; investigation, M.W., Z.L. and H.H.; resources, M.W. and H.H.; data curation, M.W. and H.H.; writing—original draft preparation, M.W.; writing—review and editing, M.W. and Z.L.; visualization, M.W.; supervision, G.H.; project administration, G.H.; funding acquisition, G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported under grant by the National Natural Science Foundation of China (No. 61471101 and No. 61571094).

Acknowledgments: Special thanks should be given to all the participants who have grant consent to this research and offered their network traffic as data for analysis.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Karagiannis, T. Profiling the end host. In Proceedings of the International Conference on Passive and Active Network Measurement, Louvain-la-Neuve, Belgium, 5–6 April 2007.
2. Xia, N.; Song, H.H.; Liao, Y.; Iliofotou, M.; Nucci, A.; Zhang, Z. Mosaic: Quantifying privacy leakage in mobile networks. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 279–290. [[CrossRef](#)]
3. Jaber, M.; Cascella, R.G.; Barakat, C. Using host profiling to refine statistical application identification. In Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, 25–30 March 2012.
4. Qin, T. Behavior spectrum: An effective method for user's web access behavior monitoring and measurement. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Anaheim, CA, USA, 3–7 December 2012.
5. Gonzalez, R.; Soriente, C.; Laoutaris, N. User profiling in the time of https. In Proceedings of the 2016 Internet Measurement Conference, Santa Monica, CA, USA, 14–16 November 2016.
6. Li, H.; Xu, Z.; Zhu, H. Demographics inference through Wi-Fi network traffic analysis. In Proceedings of the 35th Annual IEEE International Conference on Computer Communications Computer Communications, IEEE INFOCOM 2016, San Francisco, CA, USA, 10–15 April 2016; pp. 1–9.
7. Subashini, S.; Kavitha, V. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.* **2011**, *34*, 1–11. [[CrossRef](#)]
8. Ristenpart, T.; Tromer, E.; Shacham, H. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; pp. 199–212.
9. Krishnamurthy, B.; Wills, C.E. Characterizing privacy in online social networks. In Proceedings of the First Workshop on Online Social Networks, Seattle, WA, USA, 17–22 August 2008; pp. 37–42.
10. Yang, Z.; Yang, M.; Zhang, Y. Appintend: Analyzing sensitive data transmission in android for privacy leakage detection. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 1043–1054.
11. Murray, D.; Durrell, K. Inferring demographic attributes of anonymous internet users. In Proceedings of the International Workshop on Web Usage Analysis and User Profiling, San Diego, CA, USA, 15 August 1999; pp. 7–20.
12. Wu, C.; Wu, F.; Liu, J.; He, S.; Huang, Y.; Xie, X. Neural demographic prediction using search query. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11 February 2019.
13. Culotta, A.; Kumar, N.R.; Cutler, J. Predicting the Demographics of Twitter Users from Website Traffic Data. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 72–78.
14. Stevanovic, D.; Vljajic, N.; An, A. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Appl. Soft Comput.* **2013**, *13*, 698–708. [[CrossRef](#)]
15. Pennacchiotti, M.; Popescu, A.-M. A Machine Learning Approach to Twitter User Classification. In Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media, Catalonia, Spain, 17–21 July 2011; pp. 281–288.
16. Alowibdi, J.S.; Buy, U.A.; Yu, P. Language independent gender classification on Twitter. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Niagara Falls, ON, Canada, 25–28 August 2013.
17. Bhat, S.Y.; Abulaish, M. Community-based features for identifying spammers in online social networks. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), San Francisco, CA, USA, 18–21 August 2013.

18. Cesare, N.; Grant, C.; Nsoesie, E.O. Detection of user demographics on social media: A review of methods and recommendations for best practices. *arXiv* **2017**, arXiv:1702.01807.
19. Gilani, Z.; Kochmar, E.; Crowcroft, J. Classification of Twitter accounts into automated agents and human users. In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Sydney, Australia, 31 July–3 August 2017.
20. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NA, USA, 3–6 December 2012; pp. 1097–1105.
21. Mikolov, T.; Karafiát, M.; Burget, L. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, 26–30 September 2010.
22. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M. Generative adversarial nets. In Proceedings of the Advances in neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2672–2680.
23. Zhao, J.; Mathieu, M.; LeCun, Y. Energy-based generative adversarial network. *arXiv* **2016**, arXiv:1609.03126.
24. Chen, X.; Duan, Y.; Houthoofd, R. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 4–9 December 2016; pp. 2172–2180.
25. He, K.; Zhang, X.; Ren, S. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
26. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146.
27. Veit, A.; Wilber, M.J.; Belongie, S. Residual networks behave like ensembles of relatively shallow networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 4–9 December 2016; pp. 550–558.
28. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
29. Ballard, D.H. Modular Learning in Neural Networks. In Proceedings of the Sixth National Conference on Artificial Intelligence, Seattle, WA, USA, 13–17 July 1987; pp. 279–284.
30. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
31. Higgins, I. *Beta-Vae: Learning Basic Visual Concepts with a Constrained Variational Framework*; ICLR: Toulon, France, 2016.
32. Li, B.; Yang, Q.; Xue, X. *Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction*; IJCAI: Pasadena, CA, USA, 2009.
33. Guo, H. Deepfm: A factorization-machine based neural network for ctr prediction. *arXiv* **2017**, arXiv:1703.04247.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).