Crawler Detection in Location-Based Services Using Attributed Action Net

Wei Xia[1], Fei Zhao[1], Haishuai Wang[2], Peng Zhang[3], Anhui Wang[1], Kang Li[1]
{weixia.xw,zf257601,anhui.wah,lk171040}@alibaba-inc.com,hwang@fairfield.edu,p.zhang@gzhu.edu.cn
[1]Alibaba Group, Shanghai, China
[2]Fairfield University, USA
[3]Guangzhou University, China

ABSTRACT

Malicious Web crawlers threaten information system due to heavily taking up bandwidth resources and stealing private user data. Ele.me, a prevalent on-demand food delivery platform in China, suffers from the negative impact of crawlers. The crawler detection systems face two major challenges: spatial patterns of the crawler behaviors and limited labeled data for training. In this paper, we present efficient solutions to tackle these challenges. Specifically, we propose a new Attributed Action Net (AANet for short) model to detect Location-Based Services (LBS) crawlers and a three-stage learning framework to train the model. AANet consists of three different embedding modules, including the action token sequence, temporal-spatial attributes of users, and the context information of the raw data. We have deployed the model at Ele.me, and both offline experiments and online A/B tests show that the proposed method is superior to the state-of-the-art models for sequence data classification on the food delivery platform.

CCS CONCEPTS

• **Information systems** → *Data mining*; • **Computing methodologies** → **Neural networks**.

KEYWORDS

crawler detection, anti-crawling, sequence classification, attributed sequence

ACM Reference Format:
Wei Xia[1], Fei Zhao[1], Haishuai Wang[2], Peng Zhang[3], Anhui Wang[1], Kang Li[1]. 2021. Crawler Detection in Location-Based Services Using Attributed Action Net. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3459637.3481907

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3481907

Figure 1: Ele.me platform (left): a prevalent location-based service platform that provides on-demand food delivery in China. The merchant information (right) collected by crawlers are selling in an illicit market. Those information include address, GPS, and monthly sales, etc.

1 INTRODUCTION

A crawler (a.k.a. Web spider or Web robot) is typically a script or a program that browses the targeted website in an automated manner [26]. Malicious crawlers may collect and sell data in illicit markets. In our case, Ele.me[1], a popular Location-Based Service (LBS) company that provides on-demand food delivery in China [30], is facing the problem that the information of millions of merchants are collected and sold in illicit markets, as illustrated in Figure 1. Many websites allow crawlers belonging to search engines to visit, however, business information on our platform can only be obtained through our app. Thus, in this study, all crawlers are harmful and need to be detected and eliminated to protect the business information of registered merchants at Ele.me.

Many companies prohibit web-crawlers accessing their web pages due to: 1) web-crawlers may degrade the availability of web servers; 2) contents in web servers are regarded as intellectual properties of the companies; 3) the pricing system of some specific products could help companies keep the leading place of the market (e.g., fluctuation of prices of flight tickets, and the lowest price of best-selling items). If competitors or other operators steal and use the data for their purposes, it will affect the company negatively and in the long run be a threat to their business model [19, 23].

Different from traditional online services, LBS has a special focus on providing location-centric services. Information is provided according to user's location. Thus, LBS crawler varies from web crawler that it must traverse all points-of-interest (POIs) over the map to collect data.

[1]https://www.ele.me

The main challenges of crawler detection in Location-Based Services are: i) **spatial pattern**: different from web crawlers, LBS crawlers must traverse the whole map to collect data. Figure 3 illustrates some patterns of LBS crawlers. Few studies have paid attention to learn the spatial patterns of LBS crawlers; ii) **limited labeled data**: CAPTCHA [2] can identify potential web robots, however, false detection leads to negative influence for user experience, and even brings customer defection to the company.

The most relevant work to address our problem is Attributed Sequence Embedding (ASE) [31]. They propose self-supervised sequence embedding learning to alleviate the limited number of labeled data challenge in fraud detection. The proposed method embeds the attribute of the sequence via a encoder-decoder network and the encoded attribute is used as the initial hidden state of long-short term memory, however, this work neglects the temporal-spatial information in LBS-based crawler detection problem. In this paper, we propose a three-stage learning framework that is able to improve the performance of crawler detection by introducing the auxiliary task learning. Specifically, we model the Uniform Resource Locator (URL) requests as an attributed sequence classification problem. Different from existing attributed sequence learning methods such as ASE [31], we incorporate the temporal-spatial information to detect the LBS-based crawlers. We present a crawler detection method called Attributed Action Net (AANet) that learns the action sequences, temporal-spatial patterns of LBS crawlers, and the context of the Uniform Resource Locators (URLs) simultaneously. We propose a three-stage learning framework to leverage the large-scale data in gateway systems where only a few labeled data are available.

To summarize, our main contributions in this work are as follows:

- To the best of our knowledge, this is the first effort to study crawler detection in Location-Based Services using deep learning;
- We present an Attributed Action Net (AANet) model that accepts URL request sequences as inputs and simultaneously learns action sequences, temporal-spatial patterns of LBS crawlers, and the context information to detect the LBS crawlers. Moreover, a three-stage learning framework is proposed to handle the large-scale data and alleviate the limited number of labeled data problem;
- We perform extensive offline experiments to demonstrate the effectiveness of the proposed model. Online A/B tests show that the proposed model can detect more crawlers than the existing systems with a higher precision.

The rest of the paper is organized as follows. Section 2 summarizes the related work. In Section 3, we elaborate the proposed AANet model and the three-stage training process. Offline and online experiments are presented in Section 4 followed by a case study. The work is summarized in Section 5.

2 RELATED WORK

2.1 Crawler Detection

Chen et al. [2] presented an overview of web crawler detection techniques and categorized detection methods into three classes: offline web-log mining, honeypot, and online web robot detection. Honeypot and online detection methods are beyond our discussion. Menshchikov el al. [19] classified offline web-log mining approaches into two categories: signature traffic analysis and machine learning methods.

Signature traffic analysis is based on certain characteristics detection, which are inherent in robotic systems contrasting to the human user, such as too high query rate, downloading HTML-only pages (without downloading scripts and CSS files) [19]. Ro et al. [23] proposed an anti-crawling method for distributed web crawlers named Long-tail Threshold Model (LTM). They concluded that the access frequency follows an exponentially decreasing curve and determined the access frequency threshold according to the Long-tail rule. Opposed to the previous methods that search for specific patterns in the logs, the aforementioned methods apply the deviation of the metric values based on typical user behaviors. These methods would benefit from the high coverage, though they require sensitivity setting for each metric.

Existing machine learning studies identify web crawlers by extracting features from sessions and train a classifier with various off-the-shelf machine learning models, such as Random Forests, GBDT [8], XGBoost [4], LightGBM [11]. Typical features extracted from sessions include *Total Requests*, *Session Duration*, *Average Time*, *Standard Deviation Time*, *Repeated Requestes*, *HTTP response codes* and *Specific Type Requests* [2]. Lagopoulos et al. [13] proposed semantic features that are extracted from a session, including *total topics*, *unique topics*, *page similarity*, *page variance* and *boolean page variance*. Companies with dozens of business units (BU) such as Alibaba may contain thousands of *topics* in gateway log system. Thus, semantic features like *topics* or *unique topics* should be designed carefully to achieve a good result, which is a labor intensive work.

Rajabnia and Jahan [22] proposed a hybrid fuzzy inference system based on NNGE (non-nested generalized exemplar) algorithm. In terms of the feature weights obtained by the NNGE algorithm, the main features are used to train the classification models, thus the dimension of the problem space is reduced. Bayesian network is another popular method in web robot detection. Suchacka and Sobkow [24] proposed Weak Bayesian Approach (WBA) and Strong Bayesian Approach (SBA) to robot detection based on characteristics of user sessions and compared in two variants: (a) using known bots' sessions form the testing samples and (b) using known bots' sessions from the verifying samples. In general, SBA gave better results in terms of practical accuracy. Haider and Elbassuoni [9] developed a two-class Boosted Decision Tree (BDT) for web robot detection based on website navigation behavior analysis.

2.2 Deep Learning Approaches for Sequence Classification

Deep learning can produce state-of-the-art results in many areas without extensive feature engineering [15, 28, 29]. Attention based deep learning methods, such as Transformers [27], BERT [6], ALBERT [14] and GPT [1, 20, 21], are widely used in Natural Language Processing (NLP) which is a typical sequence data learning problem.

Sequence data analysis by deep learning could also extend from NLP to other fields, e.g., recommender systems [3, 5, 17, 18, 25] and fraud detection [16, 31]. For fraud detection tasks, Liu et al. [16] proposed Local Intention Calibration tree-LSTM for fraud transaction detection. Zhuang [31] proposed Attributed Sequence Embedding (ASE) for fraud detection, which encodes the sequence-level

attribute with an encoder-decoder multi-layer perception (MLP) and then utilizes the encoded vector as the initialized hidden states of LSTM.

Applying deep learning for sequence classification directly to crawl detection will omit the semi-structured information of the URL requests, especially LBS crawlers. Figure 2(a) shows that the URL requests contain tremendous attributes that varies from different action tokens. The ASE [31] model aims to to embed the attribute of the sequence but fails to capture the spatial patterns of LBS crawlers. In this paper, we propose Attributed Action Net (AANet) to learn the action sequences, temporal-spatial patterns and the context attributes of LBS crawlers simultaneously.

3 PROPOSED METHOD

We present the proposed method, Attribute Action Net (AANet) and implementation details in this section. Problem formulation and feature processing are first discussed in Section 3.1. Then the architecture of the proposed model is introduced in Section 3.2, followed by the objective function in Section 3.3. Finally the three-stage learning framework is introduced in Section 3.4.

3.1 Problem Formulation

We aim to identify web crawlers at Ele.me platform, which can be formulated as a classification problem using raw gateway requests within d days, i.e., $[(url_1, time\_step_1), (url_2, time\_step_2), \cdots]$. Each url contains rich information of the user and has a corresponding $time\_step$ that contains the date and time information. Figure 2(a) illustrates the input data of the URL sequence.

Features. Gateway requests are raw URL (Uniform Resource Locator) requests and their corresponding timestamp, resulting in complex and semi-structured format. We model raw gateway requests data via three different kinds of features: Action Token Sequence (ATS), Temporal-Spatial Attributes (TSA), and Context Attributes (CA).

Action tokens represent user's actions in the platform. There are 2000 different tokens following the long tail distribution, such as MP, PP and CART etc. MP denotes main page browsing, PP denotes promotion page browsing, and CART represent to add something to the cart. Special tokens are also introduced in the model to help the model learn a proper representation. [CLS] that stands for the begin of a sequence is used as the embedding of the whole sequence. [SEP] means the end or begin of a day. [PAD] is used to pad each instance to the same length. Following [6], [MASK] is employed to be a placeholder of a masked token in the self-supervised learning process. Figure 2(a) shows how raw requests form the action token sequence. ATS_i represents action token sequence of the i^{th} user within d days. T is total time step. Behaviors of a normal user and a crawler in the app is completely different, thus, the ATS features are significant for identifying crawlers.

As illustrated in Figure 3, spatial information is crucial for a LBS crawler. Therefore, we design a temporal-spatial module to learn such information. Temporal-spatial attributes are time series data. TSA_i represents the time and spatial information of i^{th} user, in which there are one temporal attribute $time\_gap$ and 4 spatial attributes log_poi, lat_poi, lon_user and lat_user. $time\_gap$ represents the time gap between two adjacent requests in time. log_poi and lat_poi indicate the longitude and latitude of the searching points-of-interest (POIs). lon_user and lat_user denote the longitude and latitude of the user's current place. All these information is extracted from raw URLs. Most of time, location of searching POI is the same as the user's current place. Large distance of these two place, the continuous change of searching POIs or small time gap between user's different locations usually indicates abnormal behavior of users. Temporal-spatial attribute embedding module is designed to learn such abnormal pattern.

Context attributes represent attributes such as geohash, shop id or business channel, which vary over time. There are over 1000 different context attributes, which contain contextual information and most of them are string-like values rather than numerical values. It is notable that some of them are strongly related to LBS crawlers, such as POI_name (crawlers usually travel a lot of POI_name). While some other attributes are meaningless to humans, such as RND (a.k.a. 'random', which is used for bucket splitting in A/B tests)

Table 1: Symbols and Notations.

Symbols	Description
D	the whole training data set
d	the number of days of a sequence
T	time step
N_{t_j}	the number of users of task j, where $j \in \{1, 2, 3\}$
ATS_i	action token sequence of i^{th} user, $i \in \{1, 2, 3, ..., N_{t_j}\}$
$TSA_i \in R^{T\times 5}$	temporal-spatial attribute of i^{th} user
CA_i	context attribute of i^{th} user
θ_1	parameters of ATS embedding module
θ_2	parameters of TSA embedding module
θ_3	parameters of CA embedding module
θ_4	parameters of tower module
$\Phi_{\theta_1}(\cdot)$	ATS embedding module
$\Phi_{\theta_2}(\cdot)$	TSA embedding module
$\Phi_{\theta_3}(\cdot)$	CA embedding module
$\Phi_{\theta_4}(\cdot)$	tower module
$\Phi_{in}(\cdot)$	inception network
S_{ATS}	{[CLS], [SEP], [PAD], [MASK], MP , PP, CART, ...}, and $\lvert S_{ATS}\rvert = 2000$
S_{CA}	{geohash, shopid, channel, user_id, ...}, and $\lvert S_{CA}\rvert = 1000$
$x_t^{ATS_i} \in S_{ATS}$	the t^{th} token of ATS_i
$x_{t_c}^{CA_i} \in S_{CA}$	the c^{th} feature of t^{th} feature group of CA_i
$X_i^{CA} \in R^{\lvert S_{CA}\rvert\times 3}$	the context attribute of i^{th} user generated by statistical method using CA_i
$Emb^{ATS} \in R^{n1}$	ATS embedding, where n1 is the length of embedding vector
$Emb^{TSA} \in R^{n2}$	TSA embedding, where n2 is the length of embedding vector
$Emb^{CA} \in R^{n3}$	CA embedding, where n3 is the length of embedding vector
$\lambda \in [0, 1]$	the threshold for determining whether the user is a crawler

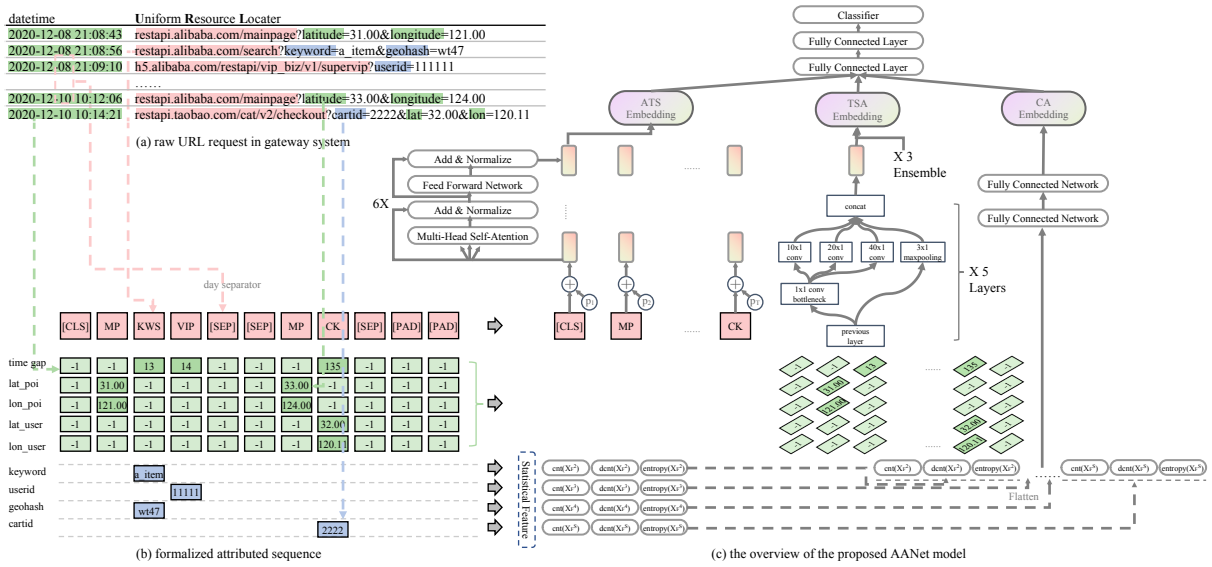


Figure 2: (a) A demonstration of raw URL requests in gateway system. (b) The formalized attributed sequence. (c) The overview of the proposed AANet model. We can see from the formalized sequence that the raw URLs (a) are formalized to three different parts (b). Tokens (in pink color) form a sequence, ⟨time gap, longitude, latitude⟩ (in green color) forms a time series, and parameters of URLs (in blue color) form a super sparse table with mostly string-like values in it. The overview of proposed AANet is presented in (c). We can find that different modules are used to process different inputs.

and SPM (Super Position Model, which is used for monitoring page view log or click log). The entropy of RND could be useful for identifying some crawlers who forget to change this parameter. As the gateway system is complex and the context varies among different business lines, the usefulness of each item will be determined by the model. We use three statistics of these attributes, which are the number of features, the number of features after deduplication and entropy. The entropy of context attributes is defined as $entropy_c = -\sum_t P(x_{t_c}) log_2 P(x_{t_c})$ where $c \in S_{CA}$.

The advantages of the proposed feature processing method are as follows:

- Rich and different types of data are fully utilized, and it does not affect the end-to-end model training.
- The processing of temporal-spatial features is greatly suitable for LBS crawler detection.
- The parameter values in URL requests, which are mostly non-numerical and string-like, can be converted to features without heavy feature engineering process.

Problem. By using ATS_i, TSA_i and CA_i that are extracted from the raw data, we can produce an accurate prediction to output whether the i^{th} user is a crawler or not. We use y_i to denote the predicted label from our model. If y_i equals to 0, the i^{th} user is not a crawler, otherwise, the i^{th} user is a crawler, such that:

$$logit_i = \Phi_{\theta_4}(concat(\Phi_{\theta_1}(ATS_i), \Phi_{\theta_2}(TSA_i), \Phi_{\theta_3}(CA_i))) \quad (1)$$

$$y_i = \begin{cases} 0, if\ logit_i < \lambda \\ 1, otherwise \end{cases} \quad (2)$$

where $\Phi_{\theta_1}(\cdot)$ denotes action token sequence embedding module, $\Phi_{\theta_2}(\cdot)$ represents the temporal-spatial attribute embedding module, $\Phi_{\theta_3}(\cdot)$ denotes context attribute embedding module, and $\Phi_{\theta_4}(\cdot)$ denotes the tower module. Other parameters are defined in Table 1.

3.2 Model architecture

The proposed model consists of the following four parts: action token sequence embedding module, temporal-spatial attribute embedding module, context attribute embedding module, and tower module. The model first processes various features using appropriate modules, and then these vectors are concatenated and fed into the tower module. Finally, predictions are given by the tower module.

3.2.1 Action Token Sequence Embedding Module. Inspired by the excellent performance of Transformer [27] in Natural Language Processing and by considering the similarities between word sequences and action sequences, we design our action sequence embedding module using multi-layer bidirectional transformer encoder. The multi-head self attention mechanism is able to learn superb representations of the dependencies between different action tokens and can capture patterns from crawler actions. We denote the number of layers as L, the hidden size as H, and the number of self-attention heads as A. In our experimental settings, we set $L = 6$, $H = 192$, $A = 6$.

Action token sequence embedding is calculated as

$$Emb_i^{ATS} = \Phi_{\theta_1}(ATS_i) \quad (3)$$

where Emb_i^{ATS} represents the embedding of action token sequence of i^{th} user and Φ_{θ_1} denotes action sequence embedding module.

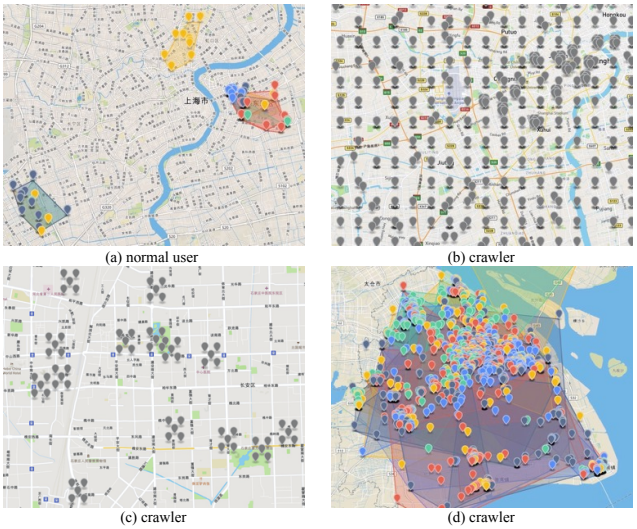


Figure 3: A demonstration of the LBS crawler. (a) illustrates a normal user who appears in a few POIs (home or workplace). (b)(c)(d) are different types of LBS crawler whose trajectories have special patterns. We can see (b) conducts grid search of the map, (c) randomly picks some POIs where every 5 points form a square shape and (d) travels almost all places of a city.

3.2.2 Temporal-spatial Attribute Embedding Module. The temporal-spatial attribute embedding module deals with time series data, which contains time and spatial information. Inspired by InceptionTime [7], we design our model architecture based on Inception network, which is one of the state-of-the-art backbones for time series classification. In order to enhance the ability of the Inception network to express temporal and spatial features, we use three inception networks at the same time and concatenate their results as the final temporal-spatial embedding vector. Each of these inception network is initialized randomly. Therefore, TSA_i is projected into a different representation subspace using different inception network.

$$\begin{aligned} Emb_i^{TSA} &= \Phi_{\theta_2}(TSA_i) \\ &= concat(\Phi_{in^1}(TSA_i), \Phi_{in^2}(TSA_i), \Phi_{in^3}(TSA_i)) \end{aligned} \tag{4}$$

where Emb_i^{TSA} is the TSA embedding vector of i^{th} user. $\Phi_{in^1}(\cdot)$, $\Phi_{in^2}(\cdot)$ and $\Phi_{in^3}(\cdot)$ are identical in model architecture but do not share the same weight with each other.

3.2.3 Context attribute embedding module. Parameter keys and values of an URL request contain rich information, which are important for us to identify which users are crawlers. For a Location-Based Services like Ele.me, users usually search and browse restaurant in a few fixed geohash locations. Therefore, whether a user is a crawler is likely to be reflected by the number of geohash. Considering that these features are sparse and non-numerical, we use their statistical values 'count', 'distinct count' and 'entropy' as features. The statistical features of each context feature are flattened and input into a fully connected network.

We design our context attribute embedding module as:

$$\begin{aligned} V_0 &= \text{Flatten}(X_i^{CA}) \\ V_1^{CA} &= \rho(W_1^{CA} V_0 + b_1^{CA}) \\ Emb_i^{CA} &= \sigma(W_2^{CA} V_1^{CA} + b_2^{CA}) \end{aligned} \tag{5}$$

where ρ and σ are two activation functions. In the context attribute embedding module, we use *tanh* function (defined as $\rho(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$) and *sigmoid* function (defined as $\sigma(z) = \frac{1}{1+e^{-z}}$). W_k^{CA} is the k^{th} layer weight matrix of context attribute embedding module, b_k^{CA} is the k^{th} layer bias vector of the context attribute embedding module.

3.2.4 Tower module. After action sequence embedding, temporal-spatial attribute embedding and context attribute embedding are computed, and then be concatenated and fed into the tower module. The tower module is composed of two fully connected layers and two dropout layers.

We design our Tower module as:

$$\begin{aligned} V_1^{TM} &= \rho(\gamma(W_1^{TM} X_i^{TM} + b_1^{TM})) \\ logit_i &= \sigma(\gamma(W_2^{TM} V_1^{TM} + b_2^{TM})) \end{aligned} \tag{6}$$

where $X_i^{TM} = concat(Emb_i^{ATS}, Emb_i^{TSA}, Emb_i^{CA})$. ρ is *tanh* function, σ is *sigmoid* function, and γ is dropout layer. W_k^{TM} is the k^{th} layer weight matrix of Tower Module, b_k^{TM} is the k^{th} layer bias vector of the Tower module.

The overview of the proposed AANet model is illustrated in Figure 2.

3.3 Objective function

In the training phase, we minimize the classification error via the cross entropy loss:

$$Loss = \sum_{x_i \in D} \sum_{c} y_{i,c} log(p_c(x_i)) \tag{7}$$

where $x_i = (ATS_i, TSA_i, CA_i)$. D is the training data set. $c \in \{0, 1\}$, is the serial number of categories. $p_c(x_i)$ denotes the probability of x_i belonging to class c.

3.4 Three-stage learning framework

As shown in Table 2, the parameters of action sequence embedding module account for a large part of the total parameters. Conventional training methods such as training the whole model directly in the main task do not work well.

In this section, a three-stage training framework is proposed. We first use Masked Language Model (MLM) [6] to train the action sequence embedding module. Secondly, an auxiliary task that determines whether a user will have purchase behaviour in the next week is trained. Finally, we conduct the classification task to predict whether the user is a crawler.

3.4.1 Stage 1: Self-supervised pre-training. Given user behavior sequences are different from natural language sequences, existing pre-trained NLP models are inapplicable directly, therefore, we need to train from scratch. As the first step, similarly to [6], we mask 15% of the input tokens in each sequence at random, and then predict the masked tokens. Among the tokens that are masked, 80% is replaced with [MASK], 10% is replaced with a random token,

Table 2: Number of parameters of each module.

Module	Number of parameters
Action token sequence embedding module	3,014,208
Temporal-Spatial attribute embedding module	1,135,296
Context attribute embedding module	417,152
Tower module	642,051
total	5,208,707

and 10% remains unchanged. In this step, we use about 90 million training samples.

3.4.2 Stage 2: Auxiliary task training. In the second step, we use an auxiliary task to help the training process in the main task. We observe that determining whether a user is a crawler based on the past behaviors has a certain relationship with determining whether the user will have purchase behaviour next week. Therefore, a prediction of next-week-purchase is used as the auxiliary task for crawler detection. Some well-behaved users are filtered according to some business based rules, which could not be used as the auxiliary training instance. The number of training instances shrinks from 90 million to 10 million in the auxiliary task.

3.4.3 Stage 3: Main task training. In the final step, the main task for classifying whether a user is a crawler or not is trained. To train the main task, we need a well labeled data set. The action of checking whether the user is a crawler will reduce the experience of normal users, which limits the size of the main task data set. Therefore, the results of the previous deployed models and the random sampled data are checked by CAPTCHA, and the results could be used as the labels in the main task. In this step, we use about eight thousand training samples. The tower module of main task is different from that of auxiliary task, whose parameters are initialized randomly.

In conclusion, we summarize the three-stage learning framework as follows:

- Since there are a huge number of parameters in action sequence embedding module, the amount of labeled data is not enough to train a robust model. Therefore, we use 90 million samples to pre-train the modules in a self-supervised way to alleviate this issue.
- In order to further alleviate the problem of requiring large amount of labeled data, we use a similar task but the labeled data is easily to obtain to assist in training the model.
- Finally the main task is able to be trained using a small amount of labeled data.

4 EXPERIMENTS

In this section, we aim to verify the effectiveness of the proposed model using both collected historical data and the online data.

4.1 Dataset

Data is collected from historical URL request logs in the gateway system, called Baxia, at Alibaba. The format of URL is illustrated in Figure 2(a). Data collected from several days is used for training and another out-of-time data in a few days is used for testing.

Raw URL of 18 billion requests are processed into 90 million sequences. Each sub-sequence in a single day is truncated to 100 (including a [SEP]) and $d = 7$ days sub-sequences form a single sequence instance. Finally, a [CLS] will be added at the head and the sequence will be padded to make sure the length of it is $T = 701$. For self-supervised pre-train task, almost all data, including normal users and crawlers, could be used, since the object is to learn the masked tokens through the rest tokens of the sequence. In auxiliary task, some well-behaved users are filtered according to some business based rules, so the number of training samples drops from 90 million to 10 million. Finally, the main task of crawler detection is trained on the verified eight thousand instances.

4.2 Evaluation metrics

The evaluation metrics are P90R, AUC, F1 score and KS. The detection result will be used for crawler verification, include voice calling, upstream texting, etc. These methods are relatively expensive, but basically guarantee the credibility of results. User experience may be hurt if the precision of the detection result is not reliable. So the evaluation of recall at a high precision threshold is used for such purpose. P90R is the recall when the precision is 90%. AUC is the area under receiver operating characteristic curve. AUC presents the overall performance at every threshold for each algorithm. F1 score is the harmonic mean of precision of recall. KS refers to the difference (maximum value) of the cumulative distribution between good and bad samples, and is used to evaluate the risk discrimination ability of the model. These four metrics can comprehensively represent the performance of the algorithms.

4.3 Offline Experiments

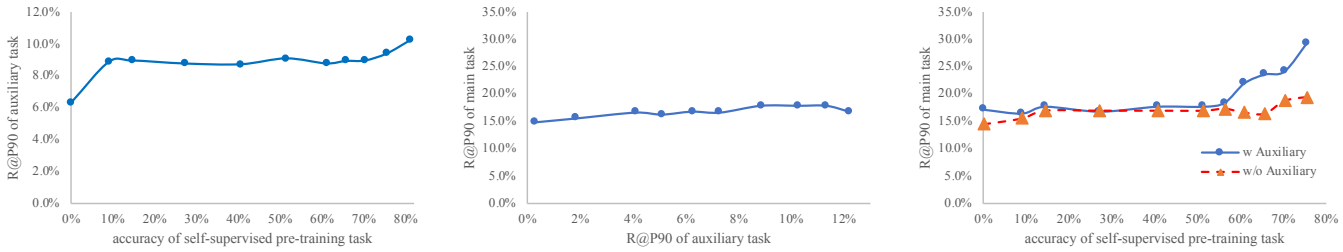
4.3.1 Ablation study of different attribute modules. In this section we evaluate the individual contributions of every module and the effects of different combinations among them. The basic modules we compared are as follows:

- **ATS**: **A**ction **T**oken **S**equence embedding module introduced in section 3.2.1, which learns the action pattern of user.
- **TSA**: **T**emporal-**S**patial **A**ttribute embedding module which learns the time gap and its corresponding latitude and longitude as described in section 3.2.2.
- **CA**: **C**ontext **A**ttribute embedding module which reflects the detail of whole request session as described in section 3.2.3.

Table 3 shows the ablation study of different modules on auxiliary task, and Table 4 shows the result on main task, which is trained after auxiliary task. Experiments in both tables use self-supervised pre-training. The best result is shown in bold.

Both tables show that the performance of two modules combined is better than single module, and all three modules together perform best in the 4 metrics. It is clear that, in Table 4, Action Token Sequence plays the most important role for crawler detection. But this conclusion does not fit for the auxiliary task.

4.3.2 Ablation study of training methods. For the three-stage training framework proposed, we conduct ablation experiments on both our proposed AANet and ASE [31]. And the results are shown in Table 5. It can be seen that each stage of the three-stage training



(a) the relationship between self-supervised pre-training task and auxiliary

(b) the relationship between auxiliary task and main task in performance (no self-

(c) the relationship between self-supervised pre-training and main task in performance

Figure 4: Sensitivity study of three training stages. It can be concluded that stage 1 (self-supervised pre-training) and stage 2 (the auxiliary task) help to improve the performance of stage 3 (the main task) altogether. The performance of main task is more sensitive to the accuracy of self-supervised pre-training. A significant improvement of main task relies on a high accuracy (above 60% accuracy) of self-supervised pre-training and the training of auxiliary task.

Table 3: Ablation study of modules on auxiliary task.

model	P90R	AUC	KS	F1 Score
ATS	12.70%	0.6960	0.2641	0.5001
TSA	2.16%	0.6668	0.2609	0.4790
CA	12.70%	0.7073	0.2734	0.5032
ATS+TSA	12.66%	0.6919	0.2640	0.4896
ATS+CA	12.86%	0.7136	0.2822	0.5118
TSA+CA	12.50%	0.7125	0.2855	0.5067
ATS+TSA+CA (AANet)	**13.22%**	**0.7274**	**0.3278**	**0.5294**

Table 4: Ablation study of modules on main task.

model	P90R	AUC	KS	F1 Score
ATS	28.75%	0.7644	0.3769	0.4570
TSA	3.39%	0.4978	0.0904	0.2906
CA	5.36%	0.6630	0.2728	0.3489
ATS+TSA	28.57%	0.7858	0.4534	0.4600
ATS+CA	28.21%	0.7855	0.4211	0.4656
TSA+CA	3.39%	0.7200	0.3528	0.3864
ATS+TSA+CA (AANet)	**29.46%**	**0.8245**	**0.5086**	**0.5115**

method we proposed is useful. If any one or two stages are removed, the training effect will be dampened.

4.3.3 Sensitivity Study. In this paper, we propose a three-stage training framework. The ablation study is shown in the previous subsection. In this subsection, we present how sensitive the self-supervised task and next-week-purchase auxiliary task affect the crawler detection (main task). The results are shown in Figure 4. Experiment is setup as follows:

- (a) Only self-supervised training is employed before main task training. After the self-supervised task is trained to reach the target accuracy, fixed learning step $Iteration = 5000$ with $batch\_size = 10$ is used to train the main task.

Table 5: Ablation study of training methods. P stands for self-supervised Pre-training, A stands for Auxiliary task and M stands for Main task.

	P	A	M	P90R	AUC	KS	F1 Score
AANet (ours)			✓	28.57%	0.2404	0.4313	0.2734
		✓	✓	16.61%	0.7988	0.4811	0.4956
	✓		✓	29.11%	0.8107	0.5064	0.4989
	✓	✓	✓	**29.46%**	**0.8245**	**0.5086**	**0.5115**
ASE [31]			✓	0.18%	0.5220	0.0414	0.2812
		✓	✓	5.71%	0.5989	0.2152	0.3340
	✓		✓	-	0.7813	0.4213	0.4537
	✓	✓	✓	14.82%	0.8160	0.4975	0.4942

- (b) Only auxiliary task is trained before main task, no self-supervised training is involved. Learning step is carefully increased to let main task reach the P90R, then fixed learning step $Iteration = 200$ with $batch\_size = 10$ is used to train main task. And the P90R of main task is reported on Y axis.
- (c) Self-supervised training is first performed same as (a). Then P90R of main task trained with auxiliary task (solid line) and without auxiliary task involved (dash line) is plotted. Auxiliary task and main task are trained using the same learning step and batch size as used in (a), (b).

From Figure 4(a), we can see that as stage 1 has a relative high accuracy (above 70%), the self-supervised pre-train can help improve the crawler detection task. In Figure 4(b), we find that the main task is not sensitive to the next-week-purchase auxiliary task if no self-supervised pre-train is involved. From Figure 4(c), two interesting conclusions could be drawn: 1) similar to (a), self-supervised training can affect the main task only when it reaches a high masked-token-prediction accuracy (above 60%); 2) the auxiliary task could help improve the training of the main task, but the significance of improvement happens only when self-supervised pre-train reaches a high accuracy (above 60%).

We can conclude from Table 5 and Figure 4 that the self-supervised training and auxiliary task help the main task training altogether. And the three-stage training framework is more sensitive to the

accuracy of self-supervised stage. A significant improvement of main task relies on a high accuracy of self-supervised pre-train stage.

4.3.4 Comparisons between different models. In this section, we evaluate the performance of the proposed model compared with the following baseline methods:

- **GBDT**. Traditional crawler detection methods first extract statistics features which are described in Section 2, including *Total Requests*, *Session Duration*, *Average Time* and semantic features such as *total topics*, *unique topics* detailed in [2, 13]. There are 268 extracted features. Then Gradient Boosting Decision Tree (GBDT) [8] is employed to classify crawlers.
- **LSTM**. Long Short term memory (LSTM) is a widely used deep learning sequence classification algorithm [10]. Action tokens are first converted as embedding then used as input of LSTM. Auxiliary task and main task are adopted to output the final result.
- **InceptionTime**. Fawaz et. al. propose InceptionTime for time series classification in [7] and show it is one of the state-of-art time series classification methods. Action token embeddings and temporal-spatial features are concatenated and used as inputs to InceptionTime. The auxiliary task is trained before main task.
- **ASE** [31]. Zhuang et al. [31] proposed Attributed Sequence Embedding (ASE), which extends LSTM to attributed sequence embedding. The encoded context feature is used as the initial hidden states of LSTM. The original unsupervised embedding learning phase of ASE is used as the first pre-training step. And action tokens are first converted as embedding then used as input of ASE. Auxiliary task is also trained before main task.

For the proposed AANet, we apply Adam optimizer [12] with learning rate of 1e-4. We also use the same learning rate for LSTM, InceptionTime, ASE and AANet. Table 6 shows that our proposed method, AANet, outperforms other algorithms in all 4 evaluation metrics.

Table 6: Comparisons between different models.

model	P90R	AUC	KS	F1 Score
GBDT	2.50%	0.7327	0.3688	0.4418
LSTM	3.21%	0.5292	0.1314	0.2827
InceptionTime [7]	8.04%	0.7426	0.4044	0.4229
ASE [31]	14.82%	0.8160	0.4975	0.4942
AANet (ours)	**29.46%**	**0.8245**	**0.5086**	**0.5115**

In conclusion, we obtain the following observations from offline experiments:

- No matter which evaluation metric is used, the best performance using two modules is better than that using one module, and the best performance using three modules is better than the one using two modules. This conclusion holds for both the auxiliary task and the main task.
- The three-stage training framework is able to improve the performance for crawler detection. This is verified by the ablation study on both AANet and ASE.
- The proposed method is better than other baseline methods such as GBDT, LSTM, InceptionTime and ASE.

4.4 Online A/B test

A/B test is conducted to verify the online performance of the model proposed in this paper. Our verification methods include voice calling, upstream texting, etc. These methods are relatively expensive, but basically guarantee the credibility of results. The existing crawler detection system is constructed by expert rules and GBDT. A/B test is performed by splitting grids in 50/50 ratio. Considering data privacy, we use relative numbers to illustrate crawlers identified by the model. Compared with the results provided by the existing system, our new model has a 15.6% significant increase in the precision of prediction. Meanwhile, our new model detects 25.1% more crawlers.

Table 7: Online A/B test results of the proposed model compared to the existing detection system.

Metric	Change
The number of detected crawlers	+25.1%
Precision	+15.6%

4.5 Case study

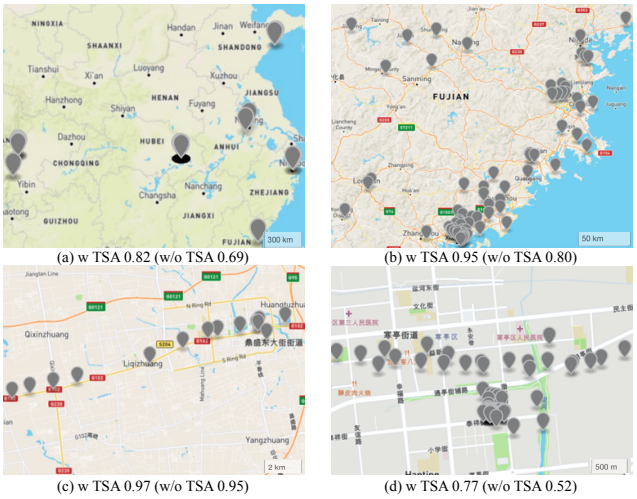
Different from traditional web crawlers, LBS crawlers need to travel as much as POIs on the map to fetch information. As described in section 3.2.2, we propose a temporal-spatial attribute embedding module to learn the patterns of LBS crawlers.

We have shown the effectiveness of Temporal-Spatial Attribute (TSA) module in section 4.3.1 and Table 4. In this section, we demonstrate some detected crawlers which have a higher prediction score when TSA module is involved. From Figure 5, we can see that some simple crawlers may search POIs across provinces (Figure 5(a)) or cites (Figure 5(b)). Professional LBS crawlers [19] will simulate a real user's travel path (Figure 5(c)). Note that sometimes professional LBS crawlers may fail to fit the simulated path to a real road on the map, as illustrated in Figure 5(d).

5 CONCLUSION

In this work, an important but under-explored problem of LBS crawler detection is studied. A new model has been proposed and is currently deployed at Alibaba, which consists of three different embedding modules for different kinds of features and a tower module for classification. In order to make full utilization of unlabeled data and auxiliary task labels, we design a three-stage training framework. Extensive offline and online experiments have demonstrated the effectiveness of the model.

There are several directions that we intend to extend this work. Firstly, the main task has limited labeled data, which is not ideal for deep learning model training. However, there is a large amount of unlabeled data. In order to make full utilization of the unlabeled data, we will investigate semi-supervised methods to improve model training process. Secondly, the method of using auxiliary tasks is a valuable and interesting idea to improve the model performance. We can explore a variety of more effective auxiliary tasks. Thirdly,



(a) w TSA 0.82 (w/o TSA 0.69)

(b) w TSA 0.95 (w/o TSA 0.80)

(c) w TSA 0.97 (w/o TSA 0.95)

(d) w TSA 0.77 (w/o TSA 0.52)

Figure 5: Case study of the detected crawlers. This figure demonstrates the predicted score via AANet with and without TSA module. We can see amateur LBS crawlers travels across provinces (a) or cites (b). Professional crawlers will simulate a real user's travel path as shown in (c). In some cases, the simulation fails to fit local trajectory of POIs into a real road as show in (d), and the model with TSA module will assign it a higher score.

our method can be applied to user comprehension task since the input data of crawler detection and user comprehension are similar.

REFERENCES

[1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[2] Hanlin Chen, Hongmei He, and Andrew Starr. 2020. An Overview of Web Robots Detection Techniques. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 1–6.

[3] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[5] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 108–116.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[8] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[9] Rabih Haidar and Shady Elbassuoni. 2017. Website navigation behavior analysis for bot detection. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 60–68.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Athanasios Lagopoulos, Grigorios Tsoumakas, and Georgios Papadopoulos. 2018. Web Robot detection: A semantic approach. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 968–974.

[14] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[15] Yann Lecun, Yoshua Bengio, and Geoffrey E Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[16] Can Liu, Qiwei Zhong, Xiang Ao, Li Sun, Wangli Lin, Jinghua Feng, Qing He, and Jiayu Tang. 2020. Fraud Transactions Detection via Behavior Tree with Local Intention Calibration. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3035–3043.

[17] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[18] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 483–491.

[19] Alexander Menshchikov, Antonina Komarova, Yurij Gatchin, Anatoly Korobeynikov, and Nina Tishukova. 2017. A study of different web-crawler behaviour. In *2017 20th Conference of Open Innovations Association (FRUCT)*. IEEE, 268–274.

[20] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI blog* (2018).

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[22] Javad Rajabnia and Majid Vafaei Jahan. 2020. Web robot detection with fuzzy inference system based on NNGE (non-nested generalized exemplar). https://www.academia.edu/8753500/Web_Robot_Detection_With_Fuzzy_Inference_System_Based_On_NNGE.

[23] Inwoo Ro, Joong Soo Han, and Eul Gyu Im. 2018. Detection Method for Distributed Web-Crawlers: A Long-Tail Threshold Model. *Security and Communication Networks* 2018 (2018).

[24] Grażyna Suchacka and Mariusz Sobkow. 2015. Detection of internet robots using a Bayesian approach. In *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*. IEEE, 365–370.

[25] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[26] Trupti V Udapure, Ravindra D Kale, and Rajesh C Dharmik. 2014. Study of web crawler and its different types. *IOSR Journal of Computer Engineering (IOSR-JCE)* 16, 1 (2014), 01–05.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[28] Haishuai Wang, Paul Avillach, et al. 2021. Diagnostic Classification and Prognostic Prediction Using Common Genetic Variants in Autism Spectrum Disorder: Genotype-Based Deep Learning. *JMIR medical informatics* 9, 4 (2021), e24754.

[29] Haishuai Wang, Zhicheng Cui, Yixin Chen, Michael Avidan, Arbi Ben Abdallah, and Alexander Kronzer. 2018. Predicting hospital readmission via cost-sensitive deep learning. *IEEE/ACM transactions on computational biology and bioinformatics* 15, 6 (2018), 1968–1978.

[30] Lin Zhu, Wei Yu, Kairong Zhou, Xing Wang, Wenxing Feng, Pengyu Wang, Ning Chen, and Pei Lee. 2020. Order Fulfillment Cycle Time Estimation for On-Demand Food Delivery. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2571–2580.

[31] Zhongfang Zhuang, Xiangnan Kong, Rundensteiner Elke, Jihane Zouaoui, and Aditya Arora. 2019. Attributed Sequence Embedding. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 1723–1728.