Revisit Prediction by Deep Survival Analysis

Sundong Kim[1], Hwanjun Song[2], Sejin Kim[2], Beomyoung Kim[2], Jae-Gil Lee[2]

[1] Institute for Basic Science, sundong@ibs.re.kr
[2] KAIST, {songhwanjun,ksj614,dglidgli,jaegil}@kaist.ac.kr

Abstract. In this paper, we introduce *SurvRev*, a next-generation revisit prediction model that can be tested directly in business. The *SurvRev* model offers many advantages. First, *SurvRev* can use *partial observations* which were considered as missing data and removed from previous regression frameworks. Using deep survival analysis, we could estimate the next customer arrival from unknown distribution. Second, *SurvRev* is an event-rate prediction model. It generates the predicted event rate of the next k days rather than directly predicting revisit interval and revisit intention. We demonstrated the superiority of the *SurvRev* model by comparing it with diverse baselines, such as the feature engineering model and state-of-the-art deep survival models.

Keywords: Predictive Analytics · Survival Analysis · Deep Learning.

1 Introduction

Predicting customer revisit in offline stores has been feasible because of the advancement in sensor technology. In addition to well-known but difficult-to-obtain customer revisit attributes, such as purchase history, store atmosphere, customer satisfaction with products, large-scale customer motion patterns captured via in-store sensors are effective in predicting customer revisit [9]. Market leaders, such as Alibaba, Amazon, and JD.com, opened the new generation of retail stores to satisfy customers. In addition, small retail chains are beginning to apply third-party retail analytics solutions built upon Wi-Fi fingerprinting and video content analytics to learn more about their customer behavior. For small stores that have not yet obtained all the aspects of customer behavior, the appropriate use of sensor data becomes more important to ensure their long-term benefit.

By knowing the visitation pattern of customers, store managers can indirectly gauge the *expected revenue*. *Targeted marketing* can also be available by knowing the revisit intention of customers. By offering discount coupons, merchants can encourage customers to accidentally revisit their stores nearby. Moreover, they can offer a sister brand with finer products to provide new shopping experiences to customers. Consequently, they can simultaneously *increase* their revenue and *satisfy* their customers. A series of previously conducted works [9, 10] introduced a method of applying feature engineering to estimate important attributes for determining customer revisit. The proposed set of features was intuitive and easy to reproduce, and the model was powered by widely known machine learning models, such as XGBoost [2].

However, some gaps did exist between their evaluation protocol and real application settings. Although their approach could effectively perform customer-revisit prediction, in *downsampled* and *cross-validated* settings, it was not guaranteed to work satisfactorily in *imbalanced visitations* with *partial observations*. In the case of class imbalance, the predictive power of each feature might disappear because of the dominance of the majority label, and such small gaps might result in further adjustment in actual deployment. In addition, in a longitudinal prediction setup, the cross-validation policy results in implicit data leakage because the testing set is not guaranteed to be collected later than the training set.

By evaluating the frameworks using chronologically split imbalanced data, the gap between previously conducted works and real-world scenarios seemed to fill. However, an unconsidered challenge, i.e., *partial observations*, occurred after splitting the dataset by time. Partial observations occur for every customer, as the model should be trained up to certain observation time. In the case of typical offline check-in data, most customers are only one-time visitors for a certain point of interest [9]. Therefore, the amount of partial observations is considerably large for individual store level. However, previously conducted works [9, 10] ignored partial observations, as their models required labels for their regression model, resulting in not only significant information loss but also biased prediction, as a model is trained using only revisited cases. In this study, we adopt *survival analysis* [18] to counter the aforementioned instances.

A practical model must predict the behavior of both partially observed customers as well as new visitors who first appear during the testing period. Predicting the revisit of both censored customers and new visitors simultaneously is very challenging, as the characteristics, such as the remaining observation time and their visit histories, of both of them inherently differ from each other. In a usual classification task, it is assumed that the class distributions between training and testing sets are the same. However, the expected arrival rate of new visitors might be lower than that of the existing customers, as the former did not appear during the training period [16]. To understand the revisit pattern using visitation histories with irregular arrival rates, we use deep learning to be free from arrival rate λ and subsequently, predict quantized revisit rates.

These abovementioned principles associated with a practical model might be crucial in applied data science research, and they offer considerable advantages compared with those offered by previously conducted works, which compromise difficulties. In the following section, we introduce our principled approach, i.e., *SurvRev*, to resolve customer-revisit prediction in more realistic settings.

Customer-Revisit Prediction [10]: Given a set of visits $V_{train} = \{v_1, \ldots, v_n\}$ with *known* revisit intentions $RV_{bin}(v_i)$ and revisit intervals $RV_{days}(v_i)$, where

$$v_i \in V_{train},\ RV_{bin}(v_i) \in \{0, 1\}, \text{ and } RV_{days}(v_i) = \begin{cases} \mathbb{R}_{>0}, & \text{if } RV_{bin}(v_i) = 1 \\ \infty, & \text{otherwise,} \end{cases}$$

build a classifier C that predicts $\hat{RV}_{bin}(v_j)$ and $\hat{RV}_{days}(v_j)$ for a new visit v_j.

2 Deep Survival Model (*SurvRev*)

In this section, we introduce our customer-revisit prediction approach. We named our model *SurvRev*, which is the condensed form of ***Surv***ival ***Rev***isit predictor.

2.1 Overall Architecture

Figure 1 depicts the overall architecture of our *SurvRev* model, which is designed as the combination of the following two modules: a *low-level visit encoder* (see § 2.2) and *high-level event-rate predictor* (see § 2.3). The low-level visit encoder learns hidden representations from each visit, and the high-level event-rate predictor estimates the event rates for the future by considering past information. The final output of the high-level module is a set of predicted revisit rates for the next k days. To calculate the loss function, we perform some calculations for converting event rates to revisit probability at time t and the expected revisit interval (see § 2.4). The entire model is trained using four different types of loss functions (see § 2.5), which are designed to optimize prediction results in terms of various metrics.

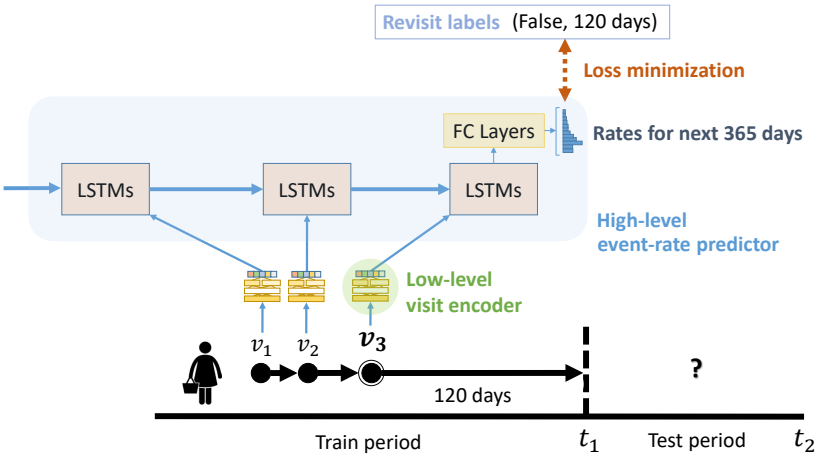


Fig. 1. Architecture of our *SurvRev* model. A training case is depicted for a censored customer who has not revisited for 120 days. The current visit data and histories of the customer are passed through low-level encoders. Subsequently, the learned representations pass through a high-level event-rate predictor that comprises long short-term memories (LSTMs) and fully connected (FC) layers. The output comprises the revisit rates for the next k days. Upon passing through several conversion steps, the model loss is minimized.

2.2 Low-Level Visit Encoder

Figure 2 depicts the architecture of the *low-level visit encoder*. In the encoder, the main area sequence inputs go through three consecutive layers and are,

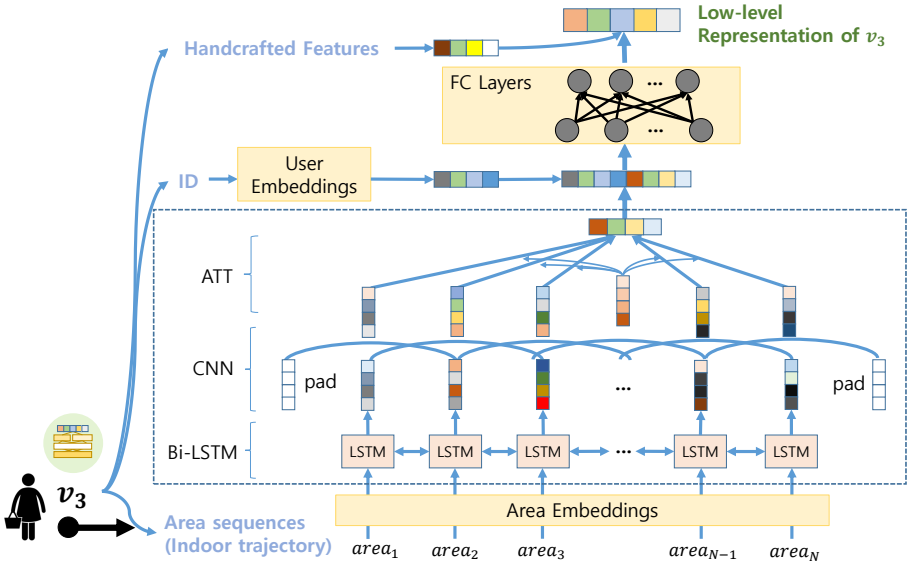


Fig. 2. Low-level visit encoder of the *SurvRev* model.

subsequently, combined with auxiliary visit-level inputs, i.e., user embeddings and handcrafted features. We first introduce three-tiered main layers for the area inputs, followed by introducing the process line of the auxiliary visit-level inputs.

Processing Area Sequences The first layer that is passed by an area sequence is a *pretrained area embedding* layer to obtain the dense representation for each sensor ID. We used the pretrained area and user embedding results via Doc2Vec [12] as initialization. The area embedding is concatenated with the dwell time, and, subsequently, it goes through a *bidirectional LSTM (Bi-LSTM) [17]*, which is expected to learn meaningful sequential patterns by looking back and forth. Each LSTM cell emits its learned representation, and the resulting sequences pass through a one-dimensional *convolutional neural networks (CNN)* to learn higher-level representations from wider semantics. We expect CNN layers to automate the manual process of grouping the areas for obtaining multilevel location semantics, such as category or gender-level sementics [10]. In business, the number of CNN layers can be determined depending on the number of meaningful semantic levels that the store manager wants to observe. The output of the CNN layer goes through *self-attention [1]* to examine all the information associated with visit. Using the abovementioned sequence of processes, *SurvRev* can learn the diverse levels of meaningful sequential patterns that determine customer revisits.

Adding Visit-Level Features Here, we concatenate a user representation with an area sequence representation and, subsequently, apply FC layers with ReLU

activation [4]. We can implicitly control the importance of both the representations by changing the dimensions for both the inputs. Finally, we concatenate selected *handcrafted features* with the combination of user and area representations. The handcrafted features contain the summary of each visit that may not be captured using the boxed component depicted in Figure 2. The selected handcrafted features are the *total dwell time*, *average dwell time*, *number of areas visited*, *number of unique areas visited*, *day of the week*, *hour of the day*, *number of prior visits*, and *previous interval*. We applied batch normalization [7] before passing the final result through the high-level module of *SurvRev*.

2.3 High-Level Event-Rate Predictor

The blue box in Figure 1 depicts the architecture of the *high-level event-rate predictor*. Its main functionality is to consider the *histories* of a customer by using dynamic LSTMs [6] and predict the revisit rate for the next k days. For each customer, the sequence of outputs from the low-level encoder becomes the input to the LSTM layers. We use dynamic LSTMs to allow sequences with variable lengths, which include a parameter to control the maximum number of events to consider. The output from the final LSTM cell goes through the FC layers with softmax activation. We set the dimension k of the final FC layer to be 365 to represent *quantized revisit rates* [8] for the next 365 days. For convenience, we refer to this 365-dim revisit rate vector as $\hat{\boldsymbol{\lambda}} = [\hat{\lambda_t}, 0 \leq t < k, t \in \mathbb{N}]$, where each element $\hat{\lambda_t}$ indicates a quantized revisit rate in a unit-time bin $[t, t+1)$.

2.4 Output Conversion

In this section, we explain the procedure to convert the 365-dim revisit rate $\hat{\boldsymbol{\lambda}}$ to other criteria, such as *probability density function*, *expected value*, and *complementary cumulative distribution function* (CCDF). The aforementioned criteria will be used to calculate the diverse loss function in § 2.5. Remember that $\hat{RV}_{days}(v)$ denotes the predicted revisit interval of visit v, meaning that *SurvRev* expects a revisit will occur after $\hat{RV}_{days}(v)$ from the time a customer made a visit v to a store.

1. Substituting the quantized event rate $\hat{\boldsymbol{\lambda}}$ from 1 gives the survival rate, i.e., $1 - \hat{\boldsymbol{\lambda}}$, which denotes the rate at which a revisit will not occur during the next unit time provided that the revisit has not happened thus far. Therefore, the cumulative product of the survival rate with time gives the quantized probability density function as follows:

$$p(\hat{RV}_{days}(v) \in [t, t+1)) = \hat{\lambda_t} \cdot \prod_{r<t} (1 - \hat{\lambda_r}). \tag{1}$$

2. Subsequently, the predicted revisit interval can be represented as a form of expected value as follows:

$$\hat{RV}_{days}(v) = \sum_{t=0}^{k} (t + 0.5) \cdot p(\hat{RV}_{days}(v) \in [t, t+1)). \tag{2}$$

3. On the basis of the last time of the observation period, it can be predicted whether a revisit is made within a period, which is denoted by $\hat{RV}_{bin}(v)$. Here, we define a *suppress time* $t_{supp}(v) = t_{end} - t_v$, where t_v denotes the visit time of v and t_{end} the time when the observation ends. We used the term *suppress time* to convey that the customer suppresses his or her desire to revisit until the time the observation ends by not revisiting the store. Thus,

$$\hat{RV}_{bin}(v) = \begin{cases} 1 & \text{if } \hat{RV}_{days}(v) \leq t_{supp}(v) \\ 0 & \text{if } \hat{RV}_{days}(v) > t_{supp}(v). \end{cases} \tag{3}$$

4. Calculating the survival rate using suppress time gives CCDF and CDF, both of which will be used to compute the cross-entropy loss. When $t_{supp}(v)$ is a natural number, the following holds:

$$p(\hat{RV}_{days}(v) \geq t_{supp}(v)) = \prod_{r<t_{supp}(v)} (1 - \hat{\lambda}_r), \tag{4}$$

$$p(\hat{RV}_{days}(v) < t_{supp}(v)) = 1 - \prod_{r<t_{supp}(v)} (1 - \hat{\lambda}_r). \tag{5}$$

2.5 Loss Functions

We designed a custom loss function to learn the parameters of our *SurvRev* model. We defined four types of losses—negative log-likelihood loss, root-mean-squared error (RMSE) loss, cross-entropy loss, and pairwise ranking losses. The prefixes $\mathcal{L}_{uc}$, $\mathcal{L}_{c}$, and $\mathcal{L}_{uc-c}$ mean that each loss is calculated for uncensored, censored, and all samples, respectively.

Negative Log-likelihood Loss $\mathcal{L}_{uc-nll}$: For performing model fitting, we minimize the *negative log-likelihood* of the empirical data distribution. We compute $\mathcal{L}_{uc-nll}$ only for those uncensored samples v in the training set that have a valid value of the next revisit interval, i.e., $\forall v : RV_{days}(v) \in \mathbb{R}_{>0}$. For step-by-step optimization, we design five cases of $\mathcal{L}_{uc-nll}$ by changing the interval parameters: $\mathcal{L}_{uc-nll-season}$, $\mathcal{L}_{uc-nll-month}$, $\mathcal{L}_{uc-nll-week}$, $\mathcal{L}_{uc-nll-date}$, and $\mathcal{L}_{uc-nll-day}$, for season, month, week, date, and day, respectively. Among these five variants, we introduce $\mathcal{L}_{uc-nll-season}$ and $\mathcal{L}_{uc-nll-month}$ by considering the case wherein $RV_{days}(v) = 105$.

- $\mathcal{L}_{uc-nll-season}$: For some applications, e.g., clothing, it is essential to capture seasonal visitation patterns. Thus, if the customer revisited within 105 days, the model learns to increase the likelihood of the interval $RV_{days}(v) \in [90, 180)$.
- $\mathcal{L}_{uc-nll-month}$: Similarly, the model learns to increase the likelihood of monthly interval $RV_{days}(v) \in [90, 120)$.

Depending on the task domain, the losses to be considered will be slightly different. Therefore, the final $\mathcal{L}_{uc-nll}$ can be a weighted sum of five variants.

RMSE Loss $\mathcal{L}_{uc-rmse}$: The second loss is the RMSE loss which minimizes the error between the predicted revisit interval $\hat{RV}_{days}(v)$ and actual interval $RV_{days}(v)$. The term $\mathcal{L}_{uc-rmse}$ minimizes the error of the model for the case of

uncensored samples. One might consider the RMSE loss a continuous expansion of negative log-likelihood loss.

Cross-Entropy Loss $\mathcal{L}_{uc\text{–}c\text{–}ce}$: Using the cross-entropy loss, one can measure the performance of the classification model whose output is a probability value between 0 and 1. The cross-entropy loss decreases as the predicted probability converges to the actual label. We separate $\mathcal{L}_{uc\text{–}c\text{–}ce}$ into $\mathcal{L}_{uc\text{–}ce}$ and $\mathcal{L}_{c\text{–}ce}$ denoting the partial cross-entropy value of the uncensored and censored sets, respectively.

$$\mathcal{L}_{uc\text{–}c\text{–}ce} = \mathcal{L}_{uc\text{–}ce} + \mathcal{L}_{c\text{–}ce}, \tag{6}$$

$$\mathcal{L}_{uc\text{–}ce} = -\sum_{v \in V_{uncensored}} \log p(\hat{RV}_{days}(v) \leq t_{supp}(v)), \tag{7}$$

$$\mathcal{L}_{c\text{–}ce} = -\sum_{v \in V_{censored}} \log p(\hat{RV}_{days}(v) > t_{supp}(v)). \tag{8}$$

Pairwise Ranking Loss $\mathcal{L}_{uc\text{–}c\text{–}rank}$: Motivated by the ranking loss function [13] and c-index [14], we introduce the pairwise ranking loss to compare the orderings between the predicted revisit intervals. This loss function fine-tunes the model by making the tendency of the predicted and the actual intervals similar to each other. The loss function $\mathcal{L}_{uc\text{–}c\text{–}rank}$ is formally defined using the following steps.

1. First, we define two matrices P and Q as follows:

$$\begin{aligned} P_{ij} &= max(0, -sgn(\hat{RV}_{days}(v_j) - \hat{RV}_{days}(v_i))), \\ Q_{ij} &= max(0, sgn(RV_{days}(v_j) - RV_{days}(v_i))). \end{aligned} \tag{9}$$

   For a censored visit v, we use the suppress time $t_{supp}(v)$ instead of the actual revisit interval $RV_{days}(v)$ to draw a comparison between uncensored and censored cases.

2. The final pairwise ranking loss is defined as follows:

$$\mathcal{L}_{uc\text{–}c\text{–}rank} = \sum_{i,j:\, v_i \in V_{uncensored}} P_{ij} \cdot Q_{ij}. \tag{10}$$

   By minimizing $\mathcal{L}_{uc\text{–}c\text{–}rank}$, our model encourages the correct ordering of pairs while discouraging the incorrect one. Both the constraint $v_i \in V_{uncensored}$ and variable Q_{ij} remove the influence of incomparable pairs, such as v_i and v_j with $RV_{days}(v_i) = 3$ and $t_{supp}(v_j) = 2$, respectively, due to the censoring effect.

Final-Loss: Combining all the losses, we can design our final objective $\mathcal{L}$ to train our *SurvRev* model. Thus,

$$\arg\min_{\theta} \mathcal{L} = \arg\min_{\theta} \mathcal{L}_{uc\text{–}nll} \cdot \mathcal{L}_{uc\text{–}rmse} \cdot \mathcal{L}_{uc\text{–}c\text{–}ce} \cdot \mathcal{L}_{uc\text{–}c\text{–}rank}, \tag{11}$$

where θ denotes a model parameter of *SurvRev*. We used the product loss to benefit from all the losses and reduce the weight parameters among the losses.

3 Experiments

To prove the efficacy of our model, we performed various experiments using a real-world in-store mobility dataset collected by *Walkinsights*. After introducing the tuned parameter values of the *SurvRev* model, we summarized the evaluation metrics required for performing revisit prediction (see § 3.1). In addition, we demonstrate the superiority of our *SurvRev* model by comparison with *seven* different baseline event prediction models (see § 3.2).

3.1 Settings

Data Preparation We used a Wi-Fi fingerprinted dataset introduced in [9], which represents customer mobility captured using dozens of in-store sensors in flagship offline retail stores located in Seoul. We selected four stores that had collected data for more than 300 days from Jan 2017. We consider each store independently, only a few customer overlaps occurred among the stores. We randomly selected 50,000 customers that had visits longer than 1 min, which is a sufficiently large number of customers to guarantee satisfactory model performance according to [10]. If a customer reappears within 10 min, we do not consider that particular subsequent visit as a new visit. We also designed several versions of training and testing sets by varying the training length to 180 and 240 days. Table 1 lists several statistics of the datasets used, where V_{tr}, V_{tef}, and V_{tep} denote the uncensored training set, testing set with first-time visitors, and testing set with partial observations who appeared in the training periods but censored, respectively. Observe the considerable difference of both average revisit rate $E[RV_{bin}(v)]$ and average revisit interval $E[RV_{days}(v)]$ among the three subsets.

Hyperparameter Settings The embedding dimension was set to be 64 for both *area embeddings* and *user embeddings*. A set of new IDs and that of new

Table 1. Statistics of the datasets.

Store ID	A		B		C		D	
Number of sensors	14		11		22		40	
Data length (days)	365		365		312		300	
Training length (days)	180	240	180	240	180	240	180	240
$\lvert V_{tr}\rvert$	39,473	49,987	45,051	57,961	50,898	67,745	35,259	48,550
$\lvert V_{tef}\rvert$	24,166	11,403	25,664	12,963	22,991	7,494	20,907	8,260
$\lvert V_{tep}\rvert$	31,409	38,179	29,511	36,193	32,208	40,474	28,069	37,562
$E[RV_{bin}(v_{tr})]$	0.204	0.236	0.345	0.376	0.367	0.403	0.204	0.226
$E[RV_{bin}(v_{tef})]$	0.204	0.140	0.285	0.183	0.233	0.112	0.116	0.053
$E[RV_{bin}(v_{tep})]$	0.191	0.146	0.203	0.152	0.223	0.126	0.115	0.058
$E[RV_{days}(v_{tr})]$	38.7	52.5	26.4	34.2	31.0	37.4	33.9	40.9
$E[RV_{days}(v_{tef})]$	45.7	30.0	30.2	15.2	30.6	17.2	28.3	13.4
$E[RV_{days}(v_{tep})]$	165.2	159.8	137.1	129.6	109.6	103.2	107.0	104.2

areas in the testing set were mapped to `[unk]` and, subsequently, embedded to default values. For the low-level module, the 64-dim Bi-LSTM unit was used. The kernel size of CNN was 3 with 16-dim filters, and the number of neurons in the FC layer was 128. We used only one dense layer. For a visit with a long sequence, we considered m areas that could cover up to 95% of all the cases, where m depends on each dataset. In the high-level module, the dynamic LSTM had 256-dim units and processed up to 5 events. We used two layers of LSTM with tanh activation. For the rate predictor, we used two FC layers with 365 neurons and ReLU activation. For training the model, we used Adam [11] optimizer with the learning rate of 0.001. We set the mini-batch size to be 32 and ran 10 epochs. The NLL loss $\mathcal{L}_{uc-nll}$ was set as the average of $\mathcal{L}_{uc-nll-season}$ and $\mathcal{L}_{uc-nll-month}$. Some of these hyperparameters were selected empirically via grid search.

Input Settings We made a switch to control the number of user histories to be used while training the *SurvRev* model. For predicting partially-observed instances (v_{tep}), all the histories up to the observation time were used to train the model. For instance, if an input visit v_5 is a partial observation, then $\{v_1, \cdots, v_5\}$ and $t_{supp}(v_5)$ are fed in the high-level event-rate predictor. For predicting *first-time visitors*, only the first appearances ($v_1 \in V_{train}$) were used to train the model. In the latter case, the LSTM length in a high-level event-rate predictor is always one because each training instance has no prior log.

Evaluation Metrics We used *two* metrics, namely, *F-score* and *c-index*, to evaluate the prediction performance.

- *F-score*: F-score measures the binary revisit classification performance.
- *C-index* [14]: C-index measures the global pairwise ordering performance, and it is the most generally used evaluation metric in survival analysis [13, 15].

3.2 Results

Comparison with Baselines We verify the effectiveness of our *SurvRev* model on the large-scale in-store mobility data. To compare our method with various baseline methods, we implemented *eight* different event-prediction models.

Baselines Not Considering Covariates: The first three baselines focus on the distribution of revisit labels and consider them an arrival process. They do not consider the attributes, i.e., covariates, obtained from each visit.

- *Majority Voting (Majority)*: Prediction results are dictated by the majority class for classification, which depends on the average values of regression; this baseline is a naive but powerful method for an imbalanced dataset.
- *Personalized Poisson Process (Poisson)* [16]: We assume that the inter-arrival time of customers follows the exponential distribution with a constant λ. To make it personalized, we control λ for each customer by considering his or her visit frequency and observation time.
- *Personalized Hawkes Process (Hawkes)* [5]: It is an extended version of the Poisson process, and it includes self-stimulation and time-decaying rate λ.

Baselines Considering Covariates: The following two models considered the covariates derived from each visit. For ensuring fairness, we used the same set of handcrafted features for the latter baseline.

- *Cox Proportional Hazard model (Cox-PH)* [3]: It is a semi-parametric survival analysis model with proportional hazards assumption.
- *Gradient Boosting Tree with Handcrafted Features (XGBoost)* [9]: It uses carefully designed handcrafted features with XGBoost classifier [2].

Baselines Using Deep Survival Analysis: The last two models are state-of-the-art survival analysis models that applied deep learning.

- *Neural Survival Recommender (NSR)* [8]: It is a deep multi-task learning model with LSTM and three-way factor unit used for music subscription data with sequential events. However, the disadvantage of this model is that the input for each cell is simple, and the input does not consider lower-level interactions.
- *Deep Recurrent Survival Analysis (DRSA)* [15]: It is an auto-regressive model with LSTM. Each cell emits a hazard rate for each timestamp. However, the drawback of this model is that each LSTM considers only a single event.

Table 2. Superiority of *SurvRev* compared to baselines, evaluated on partial observations. We highlighted in **bold** the cases where *SurvRev* shows the best performance.

Model	Store A	Store B	Store C
Majority	0.500	0.500	0.500
Poisson	0.528	0.591	0.582
Hawkes	0.530	0.593	0.580
XGBoost	0.420	0.597	0.549
NSR	0.497	0.497	0.523
DRSA	0.500	0.500	0.500
SurvRev	**0.561**	**0.672**	**0.647**

(a) C-index (180 days).

Model	Store A	Store B	Store C
Majority	0.500	0.500	0.500
Poisson	0.552	0.622	0.617
Hawkes	0.549	0.624	0.613
XGBoost	0.667	0.568	0.830
NSR	0.509	0.513	0.504
DRSA	0.500	0.500	0.501
SurvRev	0.606	**0.726**	0.702

(b) C-index (240 days).

Table 3. Superiority of *SurvRev* compared to baselines, evaluated on first-time visitors.

Model	Store A	Store B	Store C
Majority	0.000	0.000	0.000
Poisson	0.244	0.302	0.244
Hawkes	0.242	0.304	0.241
Cox-ph	0.286	0.353	0.000
XGBoost	0.236	0.317	0.097
NSR	0.000	0.000	0.000
DRSA	0.298	0.360	0.277
SurvRev	**0.315**	**0.373**	**0.295**

(a) F-score (180 days).

Model	Store A	Store B	Store C
Majority	0.000	0.000	0.000
Poisson	0.214	0.275	0.204
Hawkes	0.212	0.276	0.209
Cox-ph	0.000	0.000	0.000
XGBoost	0.025	0.194	0.000
NSR	0.000	0.000	0.000
DRSA	0.245	0.300	0.223
SurvRev	**0.272**	**0.307**	**0.263**

(b) F-score (240 days).

Comparison Results: Tables 2 and 3 summarize the performance of each model on partially observed customers (V_{tep}) and first-time visitors (V_{tef}), respectively. The prediction results on the partially observed set shows that *SurvRev* outperforms other baselines in terms of the c-index, in most cases. In addition, regarding first-time visitors, *SurvRev* outperforms other baselines in terms of the f-score. As a preliminary result, it is fairly satisfying to observe that our model showed its effectiveness on two different settings. However, we might need to further tune our model parameters to achieve the best results for every evaluation metric.

Ablation Studies Throughout *ablation studies*, we expect to observe the effectiveness of the components of both *the low-level encoder* and *high-level event-rate predictor*. The variations in both low-level encoders (L1–L6) and high-level event-rate predictors (H1–H2) are as follows:

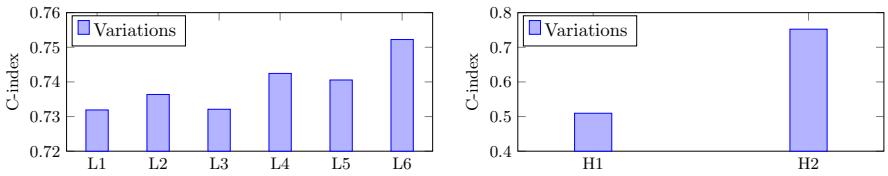
Ablation by simplifying the low-level module:

- L1 (*Bi-LSTM+ATT*): Use only two layers to represent the visit.
- L2 (*CNN+ATT*): Use only CNN and attention layers to represent the visit.
- L3 (*Bi-LSTM+CNN+AvgPool*): Substitute an attention layer to pooling.
- L4 (*Bi-LSTM+CNN+ATT*): Use only area sequence information.
- L5 (*Bi-LSTM+CNN+ATT+UserID*): Add user embedding results to L4.
- L6 (*Bi-LSTM+CNN+ATT+UserID+FE*): Add handcrafted features to L5. This one is equivalent to our original *low-level encoder* described in § 2.2.

Ablation by simplifying the high-level module:

- H1 (*FC+FC*): Concatenate the outputs of the low-level encoder and, subsequently, apply an FC layer instead of LSTMs.
- H2 (*LSTM+FC*): Stack the outputs of the low-level encoder and, subsequently, apply two-level LSTM layers. This one is equivalent to our original *high-level event-rate predictor* described in § 2.3.

Figure 3 depicts the results of the ablation study. The representative c-index results are evaluated on a partially-observed set of store D with 240-day training interval. The results show that the subcomponents of both the *low-level visit encoder* and *the high-level event-rate predictor* are critical to designing the *SurvRev* architecture.



(a) Low-level variation results. (b) High-level variation results.

Fig. 3. Ablation studies of the *SurvRev* model.

4 Conclusion

In this study, we proposed the *SurvRev* model for customer-revisit prediction. In summary, our *SurvRev* model successfully predicted customer revisit rates for the next time horizon by encoding each visit and managing the personalized history of each customer. Upon applying survival analysis with deep learning, we could easily analyze both first-time visitors and partially-observed customers with inconsistent arrival behaviors. In addition, *SurvRev* did not involve any parametric assumption. Through comparison with various event-prediction approaches, *SurvRev* proved effective by realizing several prediction objectives. For future work, we would like to extend *SurvRev* to other prediction tasks that suffer from partial observations and sessions with multilevel sequences.

Acknowledgement: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (No. 2017R1E1A1A01075927).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. ICLR (2015)
2. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. KDD (2016)
3. Cox, D.R.: Regression models and life-tables. Journal of the Royal Statistical Society. Series B (Methodological) **34**(2), 187–220 (1972)
4. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. AISTATS (2011)
5. Hawkes, A.G.: Spectra of some self-exciting and mutually exciting point processes. Biometrika **58**(1), 83–90 (1971)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML (2015)
8. Jing, H., Smola, A.J.: Neural survival recommender. WSDM (2017)
9. Kim, S., Lee, J.: Utilizing in-store sensors for revisit prediction. ICDM (2018)
10. Kim, S., Lee, J.: A systemic framework of predicting customer revisit with in-store sensors. Knowledge and Information Systems (2019)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2015)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. ICLR (2014)
13. Lee, C., Zame, W.R., Yoon, J., van der Schaar, M.: DeepHit: A deep learning approach to survival analysis with competing risks. AAAI (2018)
14. Raykar, V.C., Steck, H., Krishnapuram, B., Dehing-Oberije, C., Lambin, P.: On ranking in survival analysis: Bounds on the concordance index. NeurIPS (2007)
15. Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., Qiu, L., Yu, Y.: Deep recurrent survival analysis. AAAI (2019)
16. Ross, S.M.: Stochastic processes (Second edition). Wiley (1996)
17. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing **9**(8), 2673–2681 (1997)
18. Wang, P., Li, Y., Reddy, C.K.: Machine learning for survival analysis: A survey. ACM Computing Surveys **1**(1) (2018)