Modeling User Satisfaction Dynamics in Dialogue via Hawkes Process

Fanghua Ye[†] and **Zhiyuan Hu**[‡] and **Emine Yilmaz**[†]
[†]University College London
[‡]National University of Singapore
{fanghua.ye.19, emine.yilmaz}@ucl.ac.uk, zhiyuan_hu@u.nus.edu

Abstract

Dialogue systems have received increasing attention while automatically evaluating their performance remains challenging. User satisfaction estimation (USE) has been proposed as an alternative. It assumes that the performance of a dialogue system can be measured by user satisfaction and uses an estimator to simulate users. The effectiveness of USE depends heavily on the estimator. Existing estimators independently predict user satisfaction at each turn and ignore satisfaction dynamics across turns within a dialogue. In order to fully simulate users, it is crucial to take satisfaction dynamics into account. To fill this gap, we propose a new estimator ASAP (s**A**tisfaction e**S**timation via H**A**wkes **P**rocess) that treats user satisfaction across turns as an event sequence and employs a Hawkes process to effectively model the dynamics in this sequence. Experimental results on four benchmark dialogue datasets demonstrate that ASAP can substantially outperform state-of-the-art baseline estimators.

1 Introduction

Dialogue systems are playing an increasingly important role in our daily lives. They can serve as intelligent assistants to help users accomplish tasks and answer questions or as social companion bots to converse with users for entertainment (Ni et al., 2022; Fu et al., 2022). In recent years, the research and development of dialogue systems has made remarkable progress. However, due to the complexity of human communication, the latest dialogue systems may still fail to understand users' intents and generate inappropriate responses (Liang et al., 2021; Deng and Lin, 2022; Pan et al., 2022). These deficiencies pose huge challenges to deploying dialogue systems to real-life applications, especially high-stakes ones such as finance and health. In light of this, it is crucial to evaluate the performance of dialogue systems adequately in their development phase (Sun et al., 2021; Deriu et al., 2021).

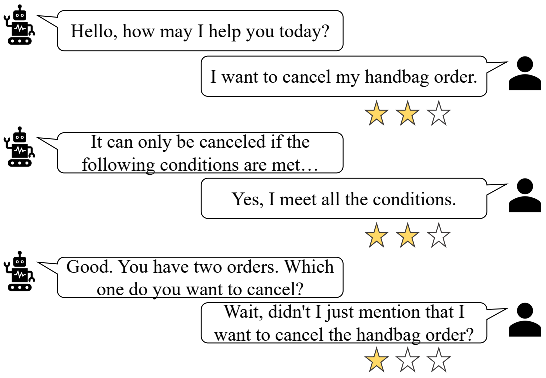

Hello, how may I help you today?
I want to cancel my handbag order.
★★☆
It can only be canceled if the following conditions are met…
Yes, I meet all the conditions.
★★☆
Good. You have two orders. Which one do you want to cancel?
Wait, didn't I just mention that I want to cancel the handbag order?
★☆☆

Figure 1: An example dialogue showing the dynamics of user satisfaction across different interaction turns.

Generally speaking, there are two types of evaluation methods, human evaluation and automatic evaluation (Deriu et al., 2021). Human evaluation is fairly effective, but costly and hard to scale up. By contrast, automatic evaluation is more scalable. However, due to the ambiguity of what constitutes a high-quality dialogue, there are currently no universally accepted evaluation metrics. Existing commonly used metrics such as BLEU (Papineni et al., 2002) usually do not agree with human judgment. Nonetheless, user satisfaction estimation (USE) has been proposed as an alternative (Bodigutla et al., 2019; Park et al., 2020; Kachuee et al., 2021; Sun et al., 2021). USE assumes that the performance of a dialogue system can be approximated by the satisfaction of its users and simulates users' satisfaction with an estimator. In this regard, USE performs automatic evaluation and is thus scalable.

Aside from helping developers find the defects of a dialogue system, USE also makes it possible to carry out timely human intervention for dissatisfied users and continuously optimize the system from human feedback (Hancock et al., 2019; Bodigutla et al., 2020; Deriu et al., 2021; Deng et al., 2022). In essence, USE is a multi-class classification problem and the goal is to predict user satisfaction at each turn. Take the dialogue shown in Figure 1 as

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 8875–8889
July 9-14, 2023 ©2023 Association for Computational Linguistics

an example, where user satisfaction is measured on a three-point scale. At the first two turns, the system responds appropriately. However, at the third turn, even though the response seems to be reasonable, the system asks for information that the user has already provided at the first turn, which may lead to dissatisfaction.

As a model-based metric, the evaluation quality of USE relies heavily on the satisfaction estimator used. In order to train a robust estimator, different approaches have been proposed (Sun et al., 2021; Liang et al., 2021; Kachuee et al., 2021; Pan et al., 2022; Deng et al., 2022). Despite the effectiveness of these approaches, they estimate user satisfaction at each turn independently and ignore the dynamics of user satisfaction across turns within a dialogue. Given that a user's satisfaction is not only related to the current dialogue context, but may also be related to the satisfaction states at previous turns, we argue that modeling user satisfaction dynamics is valuable for training a more powerful estimator.

To achieve this, we propose ASAP (s**A**tisfaction e**S**timation via H**A**wkes **P**rocess), a novel approach that leverages Hawkes process (Hawkes, 2018) to capture the dynamics of user satisfaction. Hawkes process is a self-exciting point process and it has been widely adopted to model sequential data such as financial transactions (Bacry et al., 2015) and healthcare records (Wang et al., 2018). In particular, we make the following contributions:

- We first propose a base estimator to predict user satisfaction based solely on the dialogue context. We then incorporate a Hawkes process module to model user satisfaction dynamics by treating the satisfaction scores across turns within a dialogue as an event sequence.
- We propose a discrete version of the continuous Hawkes process to adapt it to the USE task and implement this module with a Transformer architecture (Vaswani et al., 2017).
- We conduct extensive experiments on four dialogue datasets. The results show that ASAP substantially outperforms baseline methods.

2 Problem Statement

Suppose that we are provided with a dialogue session $\mathcal{X}$ containing T interaction turns, denoted as $\mathcal{X} = \{(R_1, U_1), (R_2, U_2), \ldots, (R_T, U_T)\}$. Each interaction turn t ($1 \leq t \leq T$) consists of a response R_t by the system and an utterance U_t by the user. The goal of USE is to predict the user satisfaction score s_t at each turn t based on the dialogue context $\mathcal{X}_t = \{(R_1, U_1), (R_2, U_2), \ldots, (R_t, U_t)\}$. Hence, our task is to learn an estimator $\mathcal{E} : \mathcal{X}_t \to s_t$ that can accurately estimate the user's satisfaction throughout the entire dialogue session.

Previous studies have shown that adding user action recognition (UAR) as an auxiliary task can facilitate the training of a stronger satisfaction estimator (Sun et al., 2021; Deng et al., 2022). When user action labels are available, our task shifts to learning an estimator $\mathcal{E}' : \mathcal{X}_t \to (s_t, a_t)$ that predicts user satisfaction and user action simultaneously. Here, a_t denotes the user action at turn t.

3 Method

In this section, we first describe how to build a base USE model leveraging only the dialogue context and without modeling the dynamics of user satisfaction. Then, we extend this model by integrating the Hawkes process to capture the dynamic changes of user satisfaction across dialogue turns. The overall model architecture is illustrated in Figure 2.

3.1 Base Satisfaction Estimator

Similar to Deng et al. (2022), we utilize a hierarchical transformer architecture to encode the dialogue context $\mathcal{X}_t$ into contextual semantic representations. A hierarchical architecture enables us to handle long dialogues. This architecture consists of a token-level encoder and a turn-level encoder.

3.1.1 Token-Level Encoder

The token-level encoder takes as input the concatenation of the system response R_t and user utterance U_t at each turn t and yields a single vector $\boldsymbol{h}_t$ as their semantic vector representation. To be specific, we adopt the pre-trained language model BERT (Devlin et al., 2019) to encode each (R_t, U_t) pair:

$$\boldsymbol{h}_t = \texttt{BERT}([CLS]R_t[SEP]U_t[SEP]). \tag{1}$$

3.1.2 Turn-Level Encoder

The token-level encoder can only capture the contextual information within each turn. In order to capture the global contextual information across turns, we develop a turn-level encoder that takes the semantic representations $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_t\}$ of all turns in the dialogue context $\mathcal{X}_t$ as input. We implement this encoder as a unidirectional Transformer encoder with L layers. Similar to the standard Transformer encoder layer (Vaswani et al.,

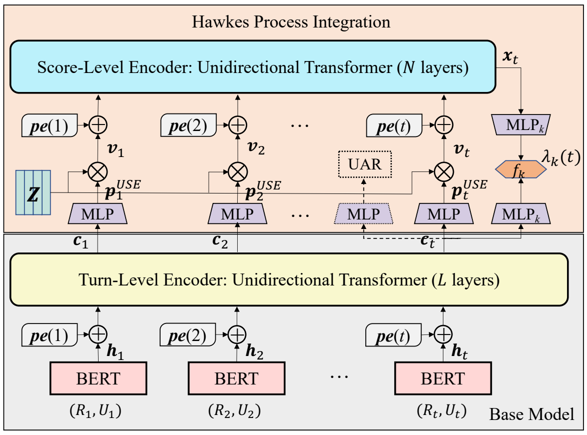


Figure 2: The architecture of our proposed model ASAP. It consists of a base estimator module and a Hawkes process integration module. Both modules leverage positional encodings to retain temporal information. Note that a single BERT model is shared by all turns and the (optional) UAR component is depicted in dashed lines.

2017), each layer includes two sub-layers. The first sub-layer is a masked multi-head attention module (`MultiHead`). The second sub-layer is a position-wise feed-forward network which is composed of two linear transformations with a ReLU activation in between (`FFN`).

Formally, each layer of the turn-level encoder operates as follows:

$$\boldsymbol{H}^{(0)} = [\boldsymbol{h}_1 + \boldsymbol{pe}(1), \dots, \boldsymbol{h}_t + \boldsymbol{pe}(t)], \quad (2)$$

$$\boldsymbol{H}^* = \texttt{MultiHead}(\boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}), \quad (3)$$

$$\boldsymbol{H}^{(l+1)} = \texttt{FFN}(\boldsymbol{H}^* + \boldsymbol{H}^{(l)}) + \boldsymbol{H}^* + \boldsymbol{H}^{(l)}, \quad (4)$$

where $\boldsymbol{H}^{(0)}$ is the input of the first layer, in which we add positional encodings $\boldsymbol{pe}(\cdot)$ to retain the turn order information. We calculate $\boldsymbol{pe}(\cdot)$ in the same way as Vaswani et al. (2017). $\boldsymbol{H}^{(L)} = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_t]$ is the output of the last layer with $\boldsymbol{c}_t$ denoting the final contextualized representation of the t-th turn. Notice that layer normalization (Ba et al., 2016) is omitted in the formulae above for simplicity.

3.1.3 Satisfaction Estimation

After acquiring the contextual representation $\boldsymbol{c}_t$, we can readily compute the probability distribution of user satisfaction at turn t by applying an MLP network (Rumelhart et al., 1986) with softmax normalization to $\boldsymbol{c}_t$, as shown below:

$$\boldsymbol{p}_t^{USE} = \texttt{softmax}(\texttt{MLP}(\boldsymbol{c}_t)), \quad (5)$$

where $\boldsymbol{p}_t^{USE} \in \mathbb{R}^K$, and K is the number of satisfaction classes. The class with the highest probability is selected as the prediction.

3.2 Hawkes Process Integration

3.2.1 Preliminaries on Hawkes Process

The Hawkes process is a self-exciting point process. It models the self-excitation of events having the same type and the mutual excitation of events with different types in an additive way. A Hawkes process is characterized by its conditional intensity function, which is defined as:

$$\lambda(t) = \mu(t) + \sum_{t_i : t_i < t} \psi(t - t_i). \quad (6)$$

Here, t_i denotes the occurrence time of a past event, $\mu(t) > 0$ is the background intensity or base intensity, and $\psi(\cdot) \geq 0$ is a pre-specified triggering kernel function. Typically, $\psi(\cdot)$ is chosen to be a time-decaying function (e.g., the exponential function $\exp(-t)$), indicating that the impacts of past events on the current event decrease through time.

While being able to model the influence of past events, the formulation in Eq. (6) is too simple to capture the complicated dynamics of many real-life event sequences. For example, it assumes that each of the past events has a positive effect on the occurrence of the current event, which can be unrealistic in numerous complex scenarios. To improve its capability, neural Hawkes process models have been devised (Mei and Eisner, 2017; Xiao et al., 2017). These models generalize the standard Hawkes process by parameterizing its intensity function with recurrent neural networks (RNNs) such as LSTM (Hochreiter and Schmidhuber, 1996). More concretely, the new intensity function is calculated in the following way:

$$\lambda(t) = \sum_{m=1}^{M} \lambda_m(t) = \sum_{m=1}^{M} f_m(\boldsymbol{w}_m^T \boldsymbol{x}_t), \quad (7)$$

where M is the total number of event types, $\boldsymbol{x}_t$ is the hidden state of the event sequence, and $\boldsymbol{w}_m$ is a parameter vector that converts $\boldsymbol{x}_t$ to a scalar. $f_m(\cdot)$ is the softplus function with a "softness" parameter β_m, i.e., $f_m(y) = \beta_m \log(1 + \exp(y/\beta_m))$. It guarantees that the intensity $\lambda(t)$ is always positive. In addition to the stronger expressiveness, this formulation of the intensity function has another advantage in that the probability of each event type m can be simply calculated as $\lambda_m(t)/\lambda(t)$.

The RNNs-based Hawkes process models inherit the intrinsic weaknesses of RNNs. Inspired by the superiority of Transformers over RNNs in dealing with sequential data, several Transformer Hawkes process models have been proposed recently (Zuo et al., 2020; Zhang et al., 2020; Zhou et al., 2022). For these models, one representative definition of the type-specific intensity function $\lambda_m(t)$ takes the form (Zuo et al., 2020):

$$\lambda_m(t) = f_m\left(\alpha_m \frac{t - t_i}{t_i} + \boldsymbol{w}_m^T \boldsymbol{x}_{t_i} + b_m\right). \quad (8)$$

In Eq. (8), b_m represents the base intensity and α_m is introduced to modulate the importance of time interpolation. This interpolation enables $\lambda_m(t)$ to be continuous over time. The overall intensity function $\lambda(t)$ is still defined as $\lambda(t) = \sum_{m=1}^{M} \lambda_m(t)$.

3.2.2 Adapting Hawkes Process for Satisfaction Estimation

Intuitively, the user satisfaction scores across turns within a dialogue can be regarded as an event sequence and each score corresponds to one type of event. Therefore, it is a natural fit to adopt Hawkes process to model the dynamics of user satisfaction. However, it is infeasible to apply the standard Hawkes process or its neural variants mentioned above directly. This is because these Hawkes processes are continuous in time, i.e., the domain of their intensity function $\lambda(t)$ is the interval $(0, T]$. A continuous Hawkes process models both *what* the next event type will be and *when* the next event will happen. By comparison, the satisfaction score sequence in our case is *discrete* in time. We only need to predict the next event type (i.e., the satisfaction score) and there is no need to predict when it will happen as we estimate user satisfaction at every turn. This difference inspires us to design a discrete version of the Hawkes process.

It is worth emphasizing that one satisfaction prediction is supposed to be made at every dialogue turn, meaning that one event regardless of its type will certainly happen at each turn. To achieve this, we constrain the intensity function $\lambda(t)$ to always take the value 1. Furthermore, following Eq. (7), $\lambda(t)$ is decomposed into:

$$\begin{aligned} &\lambda(t) = \sum_{k=1}^{K} \lambda_k(t) = 1,\ t \in \{1, 2, \ldots, T\}, \\ &\text{s.t. } \lambda_k(t) > 0,\ \ \forall k = 1, 2, \ldots, K. \end{aligned} \quad (9)$$

Recall that K represents the number of satisfaction classes. Due to $\lambda(t) = 1$, $\lambda_k(t)$ can be regarded as the probability that event type k happens (i.e., the satisfaction score is k). In Eq. (9), $\lambda(t)$ is defined on the discrete domain $\{1, 2, \ldots, T\}$ rather than the continuous interval $(0, T]$.

We propose to calculate each $\lambda_k(t)$ by the following formula:

$$\lambda_k(t) = \frac{\mathtt{exp}\big(f_k(\mathtt{MLP}_k(\boldsymbol{c}_t) + \mathtt{MLP}_k(\boldsymbol{x}_t))\big)}{\sum_{j=1}^{K} \mathtt{exp}\big(f_j(\mathtt{MLP}_j(\boldsymbol{c}_t) + \mathtt{MLP}_j(\boldsymbol{x}_t))\big)}, \quad (10)$$

where the term associated with $\boldsymbol{c}_t$ characterizes the contribution of the dialogue context $\mathcal{X}_t$ to the intensity (i.e., base intensity) and the term corresponding to $\boldsymbol{x}_t$ reveals the contribution of the satisfaction sequence. Different from Eqs. (7) and (8), we perform non-linear rather than linear transformations to convert both $\boldsymbol{c}_t$ and $\boldsymbol{x}_t$ into scalars using MLP networks. Note that $f_k(\cdot)$ is the softplus function.

Next, we describe how to compute $\boldsymbol{x}_t$, the hidden state of the satisfaction score sequence. Given the strong capability of Transformer Hawkes process models, we choose to employ a Transformer architecture (named **score-level encoder**) to compute $\boldsymbol{x}_t$. In particular, we adopt a unidirectional Transformer with N layers. Same as the turn-level encoder (refer to §3.1.2), each layer contains two sub-layers, the multi-head attention sub-layer and the position-wise feed-forward sub-layer.

The input to its first layer is the satisfaction score sequence. To convert this sequence into vector representations, we introduce an embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{d \times K}$ whose k-th column is a d-dimensional embedding for satisfaction class k. In principle, if we have the ground-truth score s_t for turn t, we can calculate the embedding vector of this turn as $\boldsymbol{Z}\boldsymbol{e}_{s_t}$, where $\boldsymbol{e}_{s_t}$ is the one-hot encoding of score s_t. In practice, however, we need to predict the satisfaction scores for all turns. Let $\hat{s}_t$ be the predicted score of turn t and $\boldsymbol{Z}\boldsymbol{e}_{\hat{s}_t}$ the corresponding embedding vector. Then, we can feed $[\boldsymbol{Z}\boldsymbol{e}_{\hat{s}_1}, \ldots, \boldsymbol{Z}\boldsymbol{e}_{\hat{s}_t}]$ to the score-level encoder to learn the dynamics of user satisfaction up to turn t and to obtain $\boldsymbol{x}_t$. This approach, albeit straightforward, has a severe limitation that there is no feedback from the score-level encoder to help train the base model because the gradients from the score-level encoder cannot be back-propagated to the base model. To overcome this limitation, we take the probability distribution of satisfaction classes $\boldsymbol{p}_t^{USE}$, as shown in Eq. (5), as the predicted "soft" score. Then, the embedding vector of turn t is computed by:

$$\boldsymbol{v}_t = \boldsymbol{Z}\boldsymbol{p}_t^{USE}. \quad (11)$$

It can be seen that $\boldsymbol{v}_t$ is a weighted sum of the embeddings of all satisfaction scores and the weights are the predicted probability by the base model.

Based on $\boldsymbol{v}_t$, the score-level encoder functions as follows to yield $\boldsymbol{x}_t$:

$$\boldsymbol{V}^{(0)} = [\boldsymbol{v}_1 + \boldsymbol{pe}(1), \ldots, \boldsymbol{v}_t + \boldsymbol{pe}(t)], \tag{12}$$

$$\boldsymbol{V}^{*} = \texttt{MultiHead}(\boldsymbol{V}^{(n)}, \boldsymbol{V}^{(n)}, \boldsymbol{V}^{(n)}), \tag{13}$$

$$\boldsymbol{V}^{(n+1)} = \texttt{FFN}(\boldsymbol{V}^{*} + \boldsymbol{V}^{(n)}) + \boldsymbol{V}^{*} + \boldsymbol{V}^{(n)}. \tag{14}$$

Similar to the turn-level encoder, we add positional encodings into the input of the first layer $\boldsymbol{V}^{(0)}$ to retain the temporal information. The output of the last layer is symbolized as $\boldsymbol{V}^{(N)} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t]$.

3.3 Training Objective

We employ the cross-entropy loss as our training objective. Recall that $\lambda_k(t)$ represents the probability of the satisfaction score being k at turn t. Thus, the training objective of USE is defined as:

$$\mathcal{L}_{USE} = -\log p(s_t|\mathcal{X}_t) = -\log \lambda_{s_t}(t), \tag{15}$$

where s_t is the ground-truth satisfaction label.

As stated in §2, adding UAR as an auxiliary task has the potential to help us train a more powerful satisfaction estimator. Even though the proposed Transformer Hawkes process model is expected to improve the performance of USE significantly, it is still meaningful to study if adding this auxiliary task can further improve the performance. To this end, we leverage an MLP network with softmax normalization on top of the turn-level encoder to calculate the probability distribution of user action when the ground-truth labels are provided:

$$\boldsymbol{p}_t^{UAR} = \texttt{softmax}(\texttt{MLP}(\boldsymbol{c}_t)). \tag{16}$$

Let $\boldsymbol{p}_{t,a_t}^{UAR}$ be the probability corresponding to the ground-truth action label a_t at turn t. The training objective of UAR is then defined as:

$$\mathcal{L}_{UAR} = -\log p(a_t|\mathcal{X}_t) = -\log \boldsymbol{p}_{t,a_t}^{UAR}. \tag{17}$$

We jointly optimize USE and UAR by minimizing the following loss:

$$\mathcal{L}_{joint} = \mathcal{L}_{USE} + \gamma\mathcal{L}_{UAR}. \tag{18}$$

Here, γ is a hyper-parameter that controls the contribution of the UAR task.

4 Experimental Setup

In what follows, we detail the experimental setup.

4.1 Datasets & Evaluation Metrics

We conduct our experiments on four publicly available dialogue datasets, including MultiWOZ 2.1 (MWOZ) (Eric et al., 2020), Schema Guided Dialogue (SGD) (Rastogi et al., 2020), JDDC (Chen et al., 2020), and Recommendation Dialogues (ReDial) (Li et al., 2018). In particular, we perform evaluations on the subsets of these datasets with user satisfaction annotations, which are provided on a five-point scale by Sun et al. (2021). Following existing works (Deng et al., 2022; Pan et al., 2022), the satisfaction annotations are mapped into three-class labels {*dissatisfied, neutral, satisfied*}. MWOZ, SGD, and ReDial are in English and all contain 1000 dialogues. While JDDC is a Chinese dataset and has 3300 dialogues. Except for ReDial, all the other three datasets have user action labels. The number of action types in MWOZ, SGD, and JDDC is 21, 12, and 236, respectively. For more details about these datasets, refer to Sun et al. (2021).

Following previous studies (Cai and Chen, 2020; Song et al., 2019; Choi et al., 2019; Deng et al., 2022), we use Accuracy (**Acc**) and Macro-averaged Precision (**P**), Recall (**R**), and F1 score (**F1**) as the evaluation metrics in our experiments.

4.2 Baseline Methods

We compare our proposed method ASAP with several state-of-the-art baseline methods in both single-task learning and multi-task learning settings.[1]

In the single-task learning setting, we only consider the USE task and the selected baselines are:

HiGRU (Jiao et al., 2019), which utilizes a hierarchical GRU structure (Cho et al., 2014) to encode the dialogue context.

HAN (Yang et al., 2016), which adds a two-level attention mechanism to HiGRU.

BERT (Devlin et al., 2019), which concatenates all the utterances in the dialogue context as a flat sequence. In addition, long sequences with more than 512 tokens are truncated automatically.

USDA (Deng et al., 2022), which leverages a hierarchical Transformer architecture to encode the dialogue context.

In the multi-task learning setting, we consider both the USE task and UAR task. And we compare ASAP to the following baseline methods:

[1] The implementation of ASAP is available at `https://github.com/smartyfh/ASAP`.

Models	IDPT	MWOZ				SGD				JDDC			
		Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
HiGRU	✗	44.6	43.7	44.3	43.7	50.0	47.3	48.4	47.5	59.7	57.3	50.4	52.0
HAN	✗	39.0	37.1	37.1	36.8	47.7	47.1	44.8	44.9	58.4	54.2	50.1	51.2
BERT	✗	46.1	45.5	47.4	45.9	56.2	55.0	53.7	53.7	60.4	59.8	58.8	59.5
USDA	✓	49.9	49.2	49.0	48.9	61.4	60.1	55.7	57.0	61.8	62.8	63.7	61.7
USDA†	✓	47.0	45.4	45.6	45.4	60.2	60.1	57.6	58.2	60.2	60.9	66.0	61.0
ASAP	✗	**56.3**‡	**55.1**‡	**55.4**‡	**55.0**‡	**64.5**‡	**62.4**‡	**61.9**‡	**62.1**‡	**65.4**‡	**64.2**‡	**68.5**‡	**65.3**‡

Table 1: Single-task performance comparison. † indicates our reproduced results. ‡ means significant performance improvements over USDA (measured by a paired t-test at $p < 0.05$). IDPT is short for in-domain pre-training.

Models	IDPT	MWOZ				SGD				JDDC			
		Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
JointDAS	✗	44.8	42.7	43.0	42.8	55.7	52.2	52.4	52.3	58.5	55.8	55.1	55.4
Co-GAT	✗	46.8	44.8	44.0	44.2	56.8	55.9	55.9	55.6	60.2	59.3	62.9	60.1
+BERT	✗	47.0	46.4	47.2	46.3	58.6	55.2	55.7	55.5	60.6	60.6	63.7	61.0
JointUSE	✗	47.6	44.6	44.9	44.7	57.4	55.0	54.8	54.7	58.3	56.6	58.7	57.2
+BERT	✗	48.9	47.2	48.0	47.3	59.0	57.4	57.1	57.3	63.8	60.8	58.6	59.2
USDA	✓	52.9	51.8	50.2	50.6	62.5	60.3	59.9	60.1	63.0	61.4	65.7	62.6
USDA†	✓	49.2	47.7	48.3	47.9	61.3	58.4	59.5	58.8	61.6	60.0	62.3	60.7
ASAP	✗	**58.1**‡	**58.1**‡	**54.7**‡	**55.6**‡	**64.8**‡	**63.0**‡	**62.3**‡	**62.6**‡	**64.1**‡	**62.6**‡	**67.3**‡	**63.9**‡

Table 2: Multi-task performance comparison. † indicates our reproduced results. ‡ means significant performance improvements over USDA (measured by a paired t-test at $p < 0.05$). IDPT is short for in-domain pre-training.

JointDAS (Cerisara et al., 2018), which jointly performs UAR and sentiment classification. We replace sentiment classification with the USE task.

Co-GAT (Qin et al., 2021), which leverages graph attention networks (Veličković et al., 2017) to perform UAR and sentiment classification. We also replace sentiment classification with the USE task.

JointUSE (Bodigutla et al., 2020), which adopts LSTM (Hochreiter and Schmidhuber, 1996) for learning temporal dependencies across turns.

USDA (Deng et al., 2022), which uses CRF (Lafferty et al., 2001) to model the sequential dynamics of user actions to facilitate USE.

Our method ASAP is closely related to USDA. The main difference is that USDA focuses on modeling user action dynamics while ASAP focuses on modeling user satisfaction dynamics. Given that user action labels may not be available in practice, our method is more applicable.

5 Experimental Results

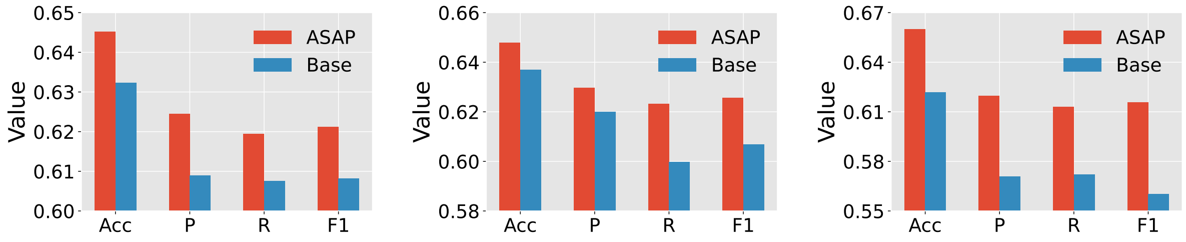
5.1 Baseline Comparison

Single-Task Learning. The results of single-task learning are summarized in Table 1 and Table 3. It can be observed that our proposed method ASAP consistently outperforms all baseline methods on all datasets. Notably, ASAP shows substantially higher performance than USDA over all four metrics even though USDA conducts in-domain pre-training to strengthen its capability of representation learning. For example, ASAP achieves 9.6%, 3.9%, 4.3%, and 6.9% F1 score improvements on MWOZ, SGD, JDDC, and ReDial, respectively.

Models	Acc	P	R	F1
HiGRU	46.1	44.4	44.0	43.5
HAN	46.3	40.0	40.3	40.0
BERT	53.6	50.5	51.3	50.0
USDA	57.3	54.3	52.9	53.4
USDA†	58.1	55.7	54.5	54.7
ASAP	**66.0**‡	**62.0**‡	**61.3**‡	**61.6**‡

Table 3: Performance comparison on ReDial.

Multi-Task Learning. The results of USE in the multi-task learning setting are reported in Table 2. For Co-GAT and JointUSE, we include results when the BERT model is leveraged. It can also be observed that the performance of ASAP is consistently higher than all baseline methods over all four metrics. For example, when compared to USDA, we observe that ASAP achieves 7.7%, 3.8%, and



(a) SGD (single-task)

(b) SGD (multi-task)

(c) ReDial (single-task)

Figure 3: Performance comparison between ASAP and the proposed base satisfaction estimator on SGD and ReDial.

3.2% absolute point improvements in terms of F1 score on MWOZ, SGD, and JDDC, respectively.

Single-Task Learning vs. Multi-Task Learning. From Tables 1 and 2, we can find that ASAP tends to perform better in the multi-task learning setting on MWOZ and SGD. This indicates that adding UAR as an auxiliary task is beneficial for improving performance. However, it is worth noting that the performance gain is relatively low. To be specific, the improvements of F1 score on MWOZ and SGD are merely 0.6% and 0.5%, respectively. Besides, on the JDDC dataset, ASAP even performs worse in the multi-task learning setting due to the large number (i.e., 236) of action types. The strong performance of ASAP in the single-task learning setting verifies the significance of modeling user satisfaction dynamics, especially considering that it is costly to collect user action labels.

In summary, our proposed method ASAP is able to outperform baseline methods in both the single-task learning setting and multi-task learning setting. Most importantly, it can achieve highly competitive performance in the single-task learning setting.

5.2 Effectiveness of Hawkes Process Integration

The above results have demonstrated the effectiveness of our method ASAP as a whole. However, it is unclear how much the Hawkes process integration module (i.e., the satisfaction dynamics modeling module) contributes to the overall performance. To better understand the effectiveness of this module, we conduct an ablation study where we compare the performance of ASAP with that of the base satisfaction estimator (refer to §3.1). Recall that the base estimator leverages only the dialogue context for USE. The results on SGD and ReDial are shown in Figure 3. For SGD, we report the results of both single-task learning and multi-task learning. From Figure 3, it can be observed that ASAP consistently outperforms the base estimator over all four metrics on both datasets. This observation validates the effectiveness of the Hawkes process integration module.

5.3 Contribution of Satisfaction Sequence to Intensity Function

As shown in Eq. (10), the dialogue context and satisfaction sequence both contribute to the intensity function of the Hawkes process. Here, we explore how much contribution should be attributed to the satisfaction sequence. This study is a supplement to the analysis in the previous section and can provide more insights into the effectiveness of satisfaction dynamics modeling. Considering that the softplus function is monotonically increasing, we can measure the importance of the satisfaction sequence by the value $\exp(\texttt{MLP}_{s_t}(\boldsymbol{x}_t))/(\exp(\texttt{MLP}_{s_t}(\boldsymbol{x}_t)) + \exp(\texttt{MLP}_{s_t}(\boldsymbol{c}_t)))$. The larger this value is, the more the satisfaction sequence contributes. We calculate this value for all samples in the test set and employ a box plot to show the distribution of these values. The detailed results are provided in Figure 4, where the triangle marker indicates the mean value. We see that the importance of the satisfaction sequence depends on the dataset. For MWOZ and SGD, the dialogue context tends to contribute more than the satisfaction sequence. In contrast, for JDDC and ReDial, the satisfaction sequence tends to be more important. Despite the variance across datasets, we can conclude that the satisfaction sequence generally plays a critical role.

5.4 Performance over Dialogue Turn

Given that longer dialogues tend to be more challenging, we further investigate the relationship between the depth of dialogue and the performance of our method. Specifically, we study how the performance changes over dialogue turn. The results of ASAP on ReDial are illustrated in Figure 5, where we also report the results of the base estimator for comparison. We omit the results of the first three

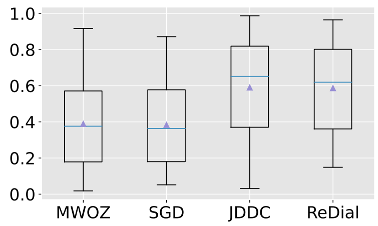


Figure 4: Contribution of satisfaction sequence to intensity function.

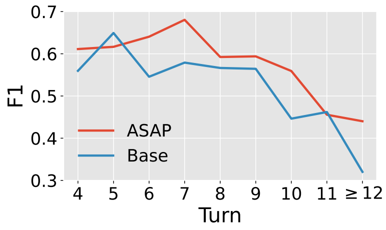


Figure 5: Performance over dialogue turn on ReDial.

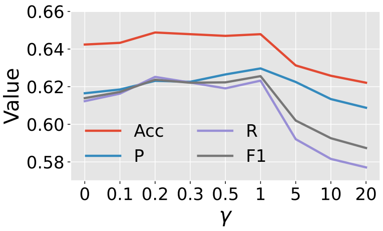


Figure 6: Effects of the parameter γ on SGD.

turns because of their short dialogue context. From Figure 5, it can be seen that ASAP outperforms the base estimator in most turns, which again verifies the effectiveness of the Hawkes process integration module. However, we observe that the performance of ASAP and the base estimator degrades when the dialogue is deep. Nonetheless, the performance of ASAP is more robust to the increase of dialogue depth, which should be attributed to the modeling of user satisfaction dynamics.

5.5 Effects of Parameter γ

Figure 6 shows the impacts of the parameter γ on the performance of our method in the multi-task learning setting. Note that γ is used to adjust the weight of the UAR task. From Figure 6, we observe that when γ takes small values, the performance is relatively stable. However, the performance drops drastically when γ becomes large. This is because when γ takes large values, the training objective is dominated by the UAR task. As a consequence, our method fails to optimize the satisfaction estimator.

6 Related Work

We briefly review related work on user satisfaction estimation and Hawkes process.

User Satisfaction Estimation. Evaluation is crucial for the development of dialogue systems (Sun et al., 2021). However, evaluating a dialogue system comprehensively can prove to be challenging due to the lack of a clear definition of what constitutes a high-quality dialogue (Deriu et al., 2021). Typically, a user study is carried out to collect feedback from end users. However, human evaluation is costly and time-intensive.

Another line of approaches is to perform evaluation from the language point of view. The main objective is to measure how natural and syntactically and semantically correct the system responses are (Kachuee et al., 2021). For example, several machine translation metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) can be used to measure if system responses are consistent with a set of provided answers. These approaches, albeit efficient, suffer from misalignment with human judgment (Novikova et al., 2017).

More recently, user satisfaction estimation has been proposed as an alternative (Liang et al., 2021; Bodigutla et al., 2019; Sun et al., 2021; Deng et al., 2022; Pan et al., 2022). It leverages human annotations regarding turn-level satisfaction to train an estimator. The estimator is then utilized to perform automatic evaluation by simulating users. Due to this, the evaluation quality depends heavily on the performance of the estimator. In the literature, different approaches have been proposed to train robust estimators (Jiang et al., 2015; Choi et al., 2019; Park et al., 2020; Deriu et al., 2021; Deng et al., 2022). However, none of them considered satisfaction dynamics, which we have shown is a severe deficiency in fully simulating users.

Hawkes Process. Hawkes process (Hawkes, 2018) is a self-exciting process and has been widely used to model sequential data (Salehi et al., 2019). To enhance the capacity of the standard Hawkes process, several RNNs-based and Transformer-based variants have been proposed (Xiao et al., 2017; Zhang et al., 2020; Zuo et al., 2020). All these Hawkes processes are continuous over time. There are also studies on discrete Hawkes processes (Seol, 2015; Browning et al., 2021). However, these discrete versions still predict when the next event happens.

7 Conclusion

In this paper, we proposed a new estimator ASAP that adopts the Hawkes process to efficiently capture user satisfaction dynamics across turns within a dialogue. Specifically, we devised a discrete version of the continuous Hawkes process to adapt it to the USE task and implemented this discrete version with a Transformer architecture. Extensive experiments on four benchmark datasets demonstrated the superiority of ASAP over baseline USE methods and the effectiveness of the Hawkes process module in modeling user satisfaction dynamics.

Limitations

Although our proposed method ASAP is able to outperform baseline estimators, an important factor it ignores is the subjectivity of user satisfaction. In practice, different users may have different degrees of satisfaction with the same dialogue. This implies that ASAP may be effective for some users, but it may also fail to predict true satisfaction for others. In order to adequately simulate a user, it is essential to take the issue of subjectivity into account. Given this, we would like to extend ASAP for personalized satisfaction estimation by incorporating user profile information in the future.

Acknowledgements

This work was funded by the EPSRC Fellowship titled "Task Based Information Retrieval" (grant reference number EP/P024289/1) and the Alan Turing Institute.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Emmanuel Bacry, Iacopo Mastromatteo, and Jean-François Muzy. 2015. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005.

Praveen Kumar Bodigutla, Lazaros Polymenakos, and Spyros Matsoukas. 2019. Multi-domain conversation quality evaluation via user satisfaction estimation. *arXiv preprint arXiv:1911.08567*.

Praveen Kumar Bodigutla, Aditya Tiwari, Spyros Matsoukas, Josep Valls-Vargas, and Lazaros Polymenakos. 2020. Joint turn and dialogue level user satisfaction estimation on multi-domain conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3897–3909, Online. Association for Computational Linguistics.

Raiha Browning, Deborah Sulem, Kerrie Mengersen, Vincent Rivoirard, and Judith Rousseau. 2021. Simple discrete-time self-exciting models can describe complex dynamic processes: A case study of covid-19. *PloS one*, 16(4):e0250015.

Wanling Cai and Li Chen. 2020. Predicting user intents and satisfaction with dialogue-based conversational recommendations. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, pages 33–42.

Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun, and Hoa T. Le. 2018. Multi-task dialog act and sentiment recognition on mastodon. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 745–754, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Meng Chen, Ruixue Liu, Lei Shen, Shaozu Yuan, Jingyan Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. The JDDC corpus: A large-scale multi-turn Chinese dialogue dataset for E-commerce customer service. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 459–466, Marseille, France. European Language Resources Association.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jason Ingyu Choi, Ali Ahmadvand, and Eugene Agichtein. 2019. Offline and online satisfaction prediction in open-domain conversational systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1281–1290.

Jianyang Deng and Yijia Lin. 2022. The benefits and challenges of chatgpt: An overview. *Frontiers in Computing and Intelligent Systems*, 2(2):81–83.

Yang Deng, Wenxuan Zhang, Wai Lam, Hong Cheng, and Helen Meng. 2022. User satisfaction estimation with sequential dialogue act modeling in goal-oriented conversational systems. In *Proceedings of the ACM Web Conference 2022*, pages 2998–3008.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54(1):755–810.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Tingchen Fu, Shen Gao, Xueliang Zhao, Ji-rong Wen, and Rui Yan. 2022. Learning towards conversational ai: A survey. *AI Open*, 3:14–28.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Alan G Hawkes. 2018. Hawkes processes and their applications to finance: a review. *Quantitative Finance*, 18(2):193–198.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 9.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, Umut Ozertem, Imed Zitouni, Ranjitha Gurunath Kulkarni, and Omar Zia Khan. 2015. Automatic online evaluation of intelligent assistants. In *Proceedings of the 24th International Conference on World Wide Web*, pages 506–516.

Wenxiang Jiao, Haiqin Yang, Irwin King, and Michael R. Lyu. 2019. HiGRU: Hierarchical gated recurrent units for utterance-level emotion recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 397–406, Minneapolis, Minnesota. Association for Computational Linguistics.

Mohammad Kachuee, Hao Yuan, Young-Bum Kim, and Sungjin Lee. 2021. Self-supervised contrastive learning for efficient user satisfaction prediction in conversational agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4053–4064, Online. Association for Computational Linguistics.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31.

Runze Liang, Ryuichi Takanobu, Feng-Lin Li, Ji Zhang, Haiqing Chen, and Minlie Huang. 2021. Turn-level user satisfaction estimation in E-commerce customer service. In *Proceedings of the 4th Workshop on e-Commerce and NLP*, pages 26–32, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Hongyuan Mei and Jason M Eisner. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, and Erik Cambria. 2022. Recent advances in deep learning based dialogue systems: A systematic survey. *Artificial intelligence review*, pages 1–101.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Yan Pan, Mingyang Ma, Bernhard Pflugfelder, and Georg Groh. 2022. User satisfaction modeling with domain adaptation in task-oriented dialogue systems. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 630–636, Edinburgh, UK. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Dookun Park, Hao Yuan, Dongmin Kim, Yinglei Zhang, Matsoukas Spyros, Young-Bum Kim, Ruhi Sarikaya, Edward Guo, Yuan Ling, Kevin Quinn, et al. 2020. Large-scale hybrid approach for predicting user satisfaction with conversational agents. *arXiv preprint arXiv:2006.07113*.

Libo Qin, Zhouyang Li, Wanxiang Che, Minheng Ni, and Ting Liu. 2021. Co-gat: A co-interactive graph attention network for joint dialog act recognition and sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13709–13717.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Farnood Salehi, William Trouleau, Matthias Grossglauser, and Patrick Thiran. 2019. Learning hawkes processes from a handful of events. *Advances in Neural Information Processing Systems*, 32.

Youngsoo Seol. 2015. Limit theorems for discrete hawkes processes. *Statistics & Probability Letters*, 99:223–229.

Kaisong Song, Lidong Bing, Wei Gao, Jun Lin, Lujun Zhao, Jiancheng Wang, Changlong Sun, Xiaozhong Liu, and Qiong Zhang. 2019. Using customer service dialogues for satisfaction analysis with context-assisted multiple instance learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 198–207, Hong Kong, China. Association for Computational Linguistics.

Weiwei Sun, Shuo Zhang, Krisztian Balog, Zhaochun Ren, Pengjie Ren, Zhumin Chen, and Maarten de Rijke. 2021. Simulating user satisfaction for the evaluation of task-oriented dialogue systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2499–2506.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2447–2456.

Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen Chu. 2017. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. 2020. Self-attentive hawkes process. In *International conference on machine learning*, pages 11183–11193. PMLR.

Zihao Zhou, Xingyi Yang, Ryan Rossi, Handong Zhao, and Rose Yu. 2022. Neural point process for learning spatiotemporal event dynamics. In *Learning for Dynamics and Control Conference*, pages 777–789. PMLR.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. 2020. Transformer hawkes process. In *International conference on machine learning*, pages 11692–11702. PMLR.

A Implementation & Training Details

In our experiments, we follow the same procedure as Deng et al. (2022) to pre-process all datasets. For the token-level BERT encoder, we employ the pre-trained BERT-base-uncased model to initialize its weights for MWOZ, SGD, and ReDial. For the JDDC dataset, we use the pre-trained BERT-base-Chinese model for initialization. Both pre-trained models are available from HuggingFace[2]. For the turn-level encoder, we fix the number of attention heads at 12 and set the number of layers (i.e., L) to 2. For the score-level encoder (i.e., the Transformer Hawkes process module), we also fix the number of attention heads at 12. But we treat the number of its layers (i.e., N) as a hyper-parameter and choose the value from $\{2, 4, 6, 8, 10, 12\}$. The dimension d of the embedding of each satisfaction class is fixed at 768. For both the turn-level encoder and score-level encoder, the hidden size of the Transformer `FFN` inner representation layer is set to 3072. All the other involved MLP networks contain only one hidden layer with the hidden size set to 192. The size of their output layers is either the number of satisfaction classes or the number of action types. The "softness" parameter β of the softplus function is fixed at 1.

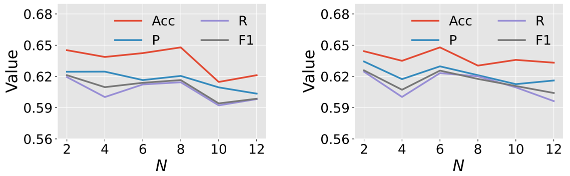
AdamW (Loshchilov and Hutter, 2017) is exploited as the optimizer, and a linear schedule with warmup is created to adjust the learning rate dynamically. The peak learning rate is chosen from {1e-5, 2e-5}. The warmup proportion is set to 0.1. The dropout ratio is also set to 0.1. For all datasets, we train the model for up to 5 epochs. For MWOZ and SGD, we adopt a batch size of 16. While we set the batch size to 24 for ReDial and JDDC. In the multi-task learning setting, we set the parameter γ for MWOZ, SGD, and JDDC to 0.5, 1.0, and 0.1, respectively. The best model checkpoints are selected based on the F1 score on the validation set. For all experiments, we use a fixed random seed 42. And it took us around 300 GPU hours to finish the experiments.

To justify that the performance improvements of our proposed method are significant, we apply the SciPy package's stats.ttest_rel function[3] to perform a paired t-test against the most competitive baseline USDA and calculate the p-value.

B Performance of User Action Recognition

Recall that in the multi-task learning setting, our method ASAP is trained to predict user satisfaction and user action simultaneously. We have presented the results on USE. In this part, we further investigate the performance on the UAR task. The results on MWOZ, SGD, and JDDC are summarized in Table 4, from which we can see that while ASAP slightly underperforms USDA on MWOZ and SGD according to the official USDA results, its performance is on par with that of USDA based on our reproduced results. Compared to other baselines, ASAP consistently achieves better results on both MWOZ and SGD. However, on the JDDC dataset, we find that the performance of ASAP is relatively low. This is because we have used a small value of 0.1 for γ on this dataset. Because of this, during the training phase, ASAP is mainly optimized for the USE task rather than the UAR task. It is worth emphasizing that our focus is on improving the performance of USE instead of UAR in this work. Thus, the reported UAR results are based on the checkpoints which achieve the best USE performance. These checkpoints may not fully demonstrate the capabilities of ASAP on the UAR task. In fact, we empirically found that by setting γ to larger values, ASAP can achieve much higher performance on action recognition. But this sacrifices the performance on satisfaction estimation.

C Effects of Number of Layers N in the Score-Level Encoder



(a) SGD (single-task)

(b) SGD (multi-task)

Figure 7: Effects of the number of layers (i.e., N) in the score-level encoder on SGD.

Given that the score-level encoder (i.e., the Transformer Hawkes process module) consists of N layers, it is worth studying the impacts of N on performance by varying its value. For this purpose, we conduct another experiment on the SGD dataset and choose the value of N from $\{2, 4, 6, 8, 10, 12\}$. We carry out this experiment in both the single-task

[2] https://huggingface.co/docs/transformers/model_doc/bert

[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

Models	**MWOZ**				**SGD**				**JDDC**			
	Acc	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**
JointDAS	75.1	64.5	64.7	62.8	79.5	72.1	72.7	70.9	63.4	41.8	43.6	41.1
Co-GAT	75.6	68.5	68.4	66.6	87.5	80.9	81.5	80.2	64.2	42.5	43.6	41.5
+BERT	86.2	79.8	80.1	78.8	92.5	88.2	88.3	87.6	66.7	49.4	48.9	47.5
JointUSE	76.5	68.7	67.7	66.9	85.0	78.0	78.9	77.3	61.8	39.0	41.8	38.8
+BERT	84.4	77.4	78.0	76.3	92.4	88.3	88.5	87.7	66.8	49.2	48.7	47.3
USDA	**87.7**	**82.8**	**82.4**	**81.4**	**95.8**	**93.6**	**93.4**	**93.1**	**69.7**	**53.1**	**53.0**	**51.3**
USDA†	86.3	81.3	82.2	80.3	94.5	91.4	91.2	90.8	69.4	52.3	52.1	50.6
ASAP	87.0	81.5	81.7	80.4	94.5	91.2	91.4	90.8	47.0	19.1	26.6	20.9

Table 4: Comparison of performance on user action recognition. † indicates our reproduced results. The best results are shown in bold and the second-best results are underlined.

learning setting and the multi-task learning setting. The results are shown in Figure 7. It can be observed that although different values of N lead to different results, the performance is relatively stable. Even so, the performance tends to be higher when N takes smaller values. When N is larger, it is harder to optimize the model because there are more parameters. Additionally, the model is also more prone to overfitting the data.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☒ A2. Did you discuss any potential risks of your work?
Except for the limitations, we do not yet find any other risks the proposed model would have.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Sections 3, 4, and Appendix A

☑ B1. Did you cite the creators of artifacts you used?
Section 4 and Appendix A

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The datasets we used are publicly available. As for our implementation, we will be happy to share the code upon acceptance and will create a license accordingly.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
The datasets we used are publicly available and do not contain a license. For our contribution, we will add a license upon acceptance.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The data we used do not have any sensitive information.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4.1

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.1

C ☑ Did you run computational experiments?

Section 5 and Appendices B and C

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix A

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We use a fixed random seed in our experiments.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4.1 and Appendix A

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.