Learning User Preferences and Understanding Calendar Contexts for Event Scheduling

Donghyeon Kim[†]
Korea University
Seoul, Republic of Korea
donghyeon@korea.ac.kr

Jinhyuk Lee[†]
Korea University
Seoul, Republic of Korea
jinhyuk_lee@korea.ac.kr

Donghee Choi
Korea University
Seoul, Republic of Korea
choidonghee@korea.ac.kr

Jaehoon Choi
Konolabs, Inc.
Seoul, Republic of Korea
jchoi@kono.ai

Jaewoo Kang[*]
Korea University
Seoul, Republic of Korea
kangj@korea.ac.kr

ABSTRACT

With online calendar services gaining popularity worldwide, calendar data has become one of the richest context sources for understanding human behavior. However, event scheduling is still time-consuming even with the development of online calendars. Although machine learning based event scheduling models have automated scheduling processes to some extent, they often fail to understand subtle user preferences and complex calendar contexts with event titles written in natural language. In this paper, we propose Neural Event Scheduling Assistant (NESA) which learns user preferences and understands calendar contexts, directly from raw online calendars for fully automated and highly effective event scheduling. We leverage over 593K calendar events for NESA to learn scheduling personal events, and we further utilize NESA for multi-attendee event scheduling. NESA successfully incorporates deep neural networks such as Bidirectional Long Short-Term Memory, Convolutional Neural Network, and Highway Network for learning the preferences of each user and understanding calendar context based on natural languages. The experimental results show that NESA significantly outperforms previous baseline models in terms of various evaluation metrics on both personal and multi-attendee event scheduling tasks. Our qualitative analysis demonstrates the effectiveness of each layer in NESA and learned user preferences.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Information systems** → *Personalization*;

KEYWORDS

Event scheduling; digital assistant; preference; multi-agent; recurrent neural network; convolutional neural network; highway network

---

[†]Both authors contributed equally to this work.
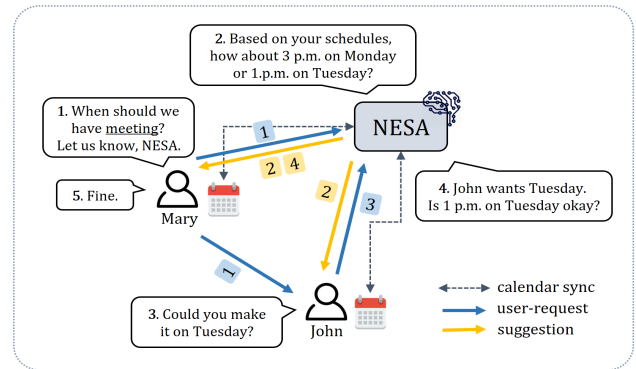[*]Corresponding author.

---

1 INTRODUCTION

Figure 1: Example of calendar event scheduling. Mary requests NESA to schedule a meeting with John. NESA considers each user's preference and calendar context, and the purpose of the event.

Calendar data has become an important context source of user information due to the popularity of online calendar services such as Google Calendar and Outlook Calendar. According to a research study conducted by Promotional Products Association International in 2011, about 40% of people referred to calendars on their computers, and about 22% of people used their mobile calendars every day [19]. As more people use online calendar services, more detailed user information is becoming available [24].

Event scheduling is one of the most common applications that uses calendar data [4, 6]. Similar to Gervasio et al. [13] and Berry et al. [2], we define event scheduling as suggesting suitable time slots for calendar events given user preferences and calendar contexts. However, even with the development of communication technology, event scheduling is still time-consuming. According to Konolabs,

Inc., the average number of emails sent between people to set the time for a meeting is 5.7.[1] At the same time, the market for digital assistants is growing fast. Gartner, Inc. stated that by 2019, at least 25% of households will use digital assistants on mobiles or other devices as primary interfaces of connected home services [32]. Thus, it is important for digital assistants to effectively schedule users' events [5].

An example of scheduling an event using NESA is illustrated in Figure 1. When a user (Mary) requests NESA to arrange an appointment with the other user (John), NESA suggests candidate time slots considering the purpose of the event (e.g., meeting), preferences of each user (e.g., Mary usually has meetings in the afternoon, and John likes to have meetings early in the week), and each user's calendar context. As a result, NESA reduces the communication cost between the users by assisting with event scheduling.

Despite its importance, automated event scheduling [4, 6, 23] has had limited success due to several reasons. First, previous studies heavily relied on hand-crafted event features such as predefined event types, fixed office/lunch hours, and so on. In addition to the cost of defining the hand-crafted event features, they could not accurately understand calendar contexts based on natural language. For instance, if a user requests to schedule a late lunch with other users, traditional scheduling systems do not suggest late lunch hours unless the keyword late is registered in the systems. Furthermore, raw online calendars frequently contain abbreviations (e.g., Mtg stands for meeting) and misspellings. To deal with natural language, recent studies have combined human labor with scheduling systems [8]. Second, most previous studies have developed their own scheduling systems to learn user preferences, which makes it difficult to apply their methodologies to other scheduling systems. Despite the wide use of the Internet standard format iCalendar [10], developing scheduling assistants based on iCalendar gained much less attention among researchers [34].

In this paper, we propose Neural Event Scheduling Assistant (NESA) which is a deep neural model that learns to schedule calendar events using raw user calendar data. NESA is a fully automated event scheduling assistant which learns user preferences and understands raw calendar contexts that include natural language. To understand various types of information in calendars, NESA leverages several deep neural networks such as Bidirectional Long Short-Term Memory (Bi-LSTM) [14, 29] and Convolutional Neural Network (CNN) [18]. The following four layers in NESA are jointly trained to schedule personal calendar events: 1) Title layer, 2) Intention layer, 3) Context layer, and 4) Output layer. After training, NESA is utilized for scheduling personal events (e.g., homework) and multi-attendee events (e.g., meetings). We compare NESA with previous preference learning models for event scheduling [3, 13, 23], and find that NESA achieves the best performance in terms of various evaluation metrics.

The contributions of our paper are four-fold.

- We introduce NESA, a fully automated event scheduling model, which learns user preferences and understands calendar contexts, directly from their raw calendar data.
- NESA successfully incorporates deep neural networks for event scheduling tasks.

- We train NESA on 593,207 real online calendar events in Internet standard format which is applicable to any calendar systems.
- NESA achieves the best performance on both personal and multi-attendee event scheduling tasks compared with other preference learning models.

The rest of this paper is organized as follows. In Section 2, we introduce some related studies on event scheduling, and briefly discuss the recent rise of neural networks. In Section 3, we formulate personal and multi-attendee event scheduling tasks. In Section 4, we discuss our deep neural model NESA that consists of Bi-LSTM, CNN, and Highway Network. In Section 5, we introduce our dataset used for the event scheduling task and discuss our qualitative analysis along with the experimental results. We conclude the paper and discuss future work in Section 6. We make our source code and pretrained NESA[2] available so that researchers and machine learning practitioners can easily apply NESA to their scheduling systems.

2 RELATED WORK AND BACKGROUND

2.1 Preference Learning for Event Scheduling

Since the development of online calendars, researchers have focused on learning user preferences for scheduling calendar events. Mitchell et al. proposed Calendar Apprentice (CAP) which is a decision tree based calendar manager that can learn user scheduling preferences from experience [23]. Blum et al. introduced the Winnow and weighted-majority algorithms that outperformed CAP [6] on predicting various attributes of calendar events. Mynatt et al. also utilized the context of a user's calendar to infer the user's event attendance [25]. Berry et al. proposed an assistant called Personalized Calendar Assistant (PCalM), which is based on Naive Bayesian, for ranking candidate schedules [4]. Refanidis et al. have developed an intelligent calendar assistant which uses a hierarchical preference model [28].

However, most event scheduling models were based on specific calendar systems using hand-crafted event features such as predefined event types and system dependent features. Previous scheduling methodologies are rarely used for modern event scheduling systems due to the high cost of designing hand-crafted features. Also, it is difficult for existing models to understand user calendars that often include user written texts such as event titles. In this paper, we propose NESA which learns to schedule calendar events, directly using raw calendar data that contains natural language texts. As NESA is trained on the Internet standard format, it is generally applicable to other calendar systems.

2.2 Multi-Attendee Event Scheduling

Event scheduling has also been studied in the context of multi-attendee event scheduling. Researches on event scheduling focus on solving constraint satisfaction problems (CSPs), and such researches often assume that user preferences are already given. Garrido et al. used heuristics functions for finding the priority value of each available time interval [12]. Wainer et al. proposed a model to find optimal time intervals based on user preferences and dealt with

---

[1]Statistics obtained by Konolabs, Inc. (https://kono.ai) in 2017.

[2]https://github.com/donghyeonk/nesa

privacy issues of shared calendars [34]. Zunino et al. developed Chronos, a multi-attendee meeting scheduling system that employs a Bayesian network to learn user preferences [36].

However, most multi-attendee event scheduling models still depend on their own scheduling systems. Furthermore, due to the small amount of existing calendar event data (e.g., 2K events of 2 users [6, 23, 36]), some of the previous studies [6, 12] use complicated heuristic functions based on system dependent features to find proper time intervals, making their methodologies difficult to adopt. In contrast, NESA leverages 593K standard formatted events and learns event scheduling directly from raw calendar data. While the recent work of Cranshaw et al. relied on human labor for more effective event scheduling [8], our event scheduling assistant is fully automated. We also demonstrate the effectiveness of NESA on multi-attendee event scheduling.

2.3 Representation Learning using Deep Neural Networks

Many classification tasks such as image classification [18], sentiment analysis [11], and named-entity recognition [20] have benefited from the recent rise of neural networks. Deep neural networks learn how to represent raw inputs such as image pixels for any targeted task. Given a raw user calendar, NESA learns how to represent user preferences and calendar contexts for event scheduling. While the preliminary work of Mitchell et al. showed that decision tree based models with hand-crafted features are better than artificial neural network (ANN) based models with hand-crafted features [23], our work is the first to show that deep neural networks are effective for event scheduling tasks with raw calendar data.

Among various kinds of neural networks, Recurrent Neural Networks (RNNs) have achieved remarkable performance on natural-language processing (NLP) tasks such as language modeling [21], machine translation [1], and so on. Inspired by a character-level language model [16] and state-of-the-art question answering models [30], NESA handles various semantics coming from raw calendar events based on natural language. We use RNN and CNN to effectively represent user written event titles, and use Highway Network [31] to find nonlinear relationships among various calendar attributes.

3 PROBLEM FORMULATION

3.1 Attributes of Calendar Data

A user's calendar data consists of sequences of events which are sorted by their registered time. Each calendar event has at least five attributes: (1) *title* (what to do), (2) *start time*, (3) *duration*, (4) *registered time*, and (5) *user identifier* of an event. Although many other attributes (e.g., *location, description*) exist, we focus on the most common attributes of events. Note that the title of each event in iCalendar format does not have a label that indicates the event type, whereas previous scheduling systems rely on a predefined set of event types.

To simplify the problem, we group all the events of each user by the week in which their events start. For example, user A's events that start within the 15th week of 2018 will be grouped in A_2018_15. In each group, events are sorted by their registered time. For each user, all the K events in a specific week can be expressed

as follows: $E = e_1, \ldots, e_K$, and $e_i = (x_i, t_i, d_i, u_i)$ for $i = 1$ to K where x_i indicates the start time, t_i is the title, d_i is the duration, and u_i is the user identifier of e_i. We assume that u_i represents the *preference* of a user, t_i and d_i represent the *purpose* of i-th event, and $e_1, \ldots, e_{i-1}$ represent the *context* of i-th event. Note that the context can be extended to multiple weeks.

3.2 Personal Event Scheduling

Event scheduling involves considering users' preferences and calendar contexts to provide suitable time slots to users. We define personal event scheduling as scheduling events that have a single attendee (e.g., work, personal matters, and so on). We later describe how to extend personal event scheduling to multi-attendee event scheduling.

Personal event scheduling should consider the pre-registered events of the week (context) in which an event will be registered and the preferences of a user. Thus, an event scheduling model predicts the start time y_i of the i-th event e_i given the pre-registered events ($e_1, \ldots, e_{i-1}$) which constitute the context of the week, and given the title t_i, duration d_i, and user u_i attributes of the i-th event. Note that each pre-registered event also contains title, duration, user, and start time (x_i) attributes, making it difficult for any models to leverage all the given contexts.

Given the probability of target time slot y_i of event e_i, the optimal model parameters Θ^* are as follows:

$$\Theta^* = \underset{\Theta}{\arg\max}\, p(y_i | e_1, \ldots, e_{i-1}, t_i, d_i, u_i; \Theta) \qquad (1)$$

where Θ denotes the trainable parameters of a model. Note that there exist K event scheduling problems in a week including weeks with no pre-registered events. We treat each event scheduling problem as an independent problem to measure the ability of each model to understand calendar contexts and user preferences.

3.3 Multi-Attendee Event Scheduling

Multi-attendee event scheduling further considers the preferences and calendar contexts of multiple users attending an event. Given U users attending a specific event e_μ with the optimal model parameter Θ^*, the most suitable time slot y_μ^* among candidate time slots $\hat{y}_\mu$ is computed as follows:

$$y_\mu^* = \underset{\hat{y}_\mu}{\arg\max} \sum_{j=1}^{U} p(\hat{y}_\mu | E_{1:\mu-1}^j, t_\mu, d_\mu, u_j; \Theta^*) \qquad (2)$$

where $E_{1:\mu-1}^j$ denotes a group of j-th user's pre-registered events before the event e_μ (i.e., calendar context). In this way, we choose a time slot that maximizes the satisfaction of multiple users. Note that the number of pre-registered events may differ between users. Also, while we have assumed all users have the same influence in multi-attendee event scheduling, more sophisticated aggregation such as multiplying a weighting factor for each user is possible. However, we use the simplest form of aggregation to test the effectiveness of each model trained on personal event scheduling data.
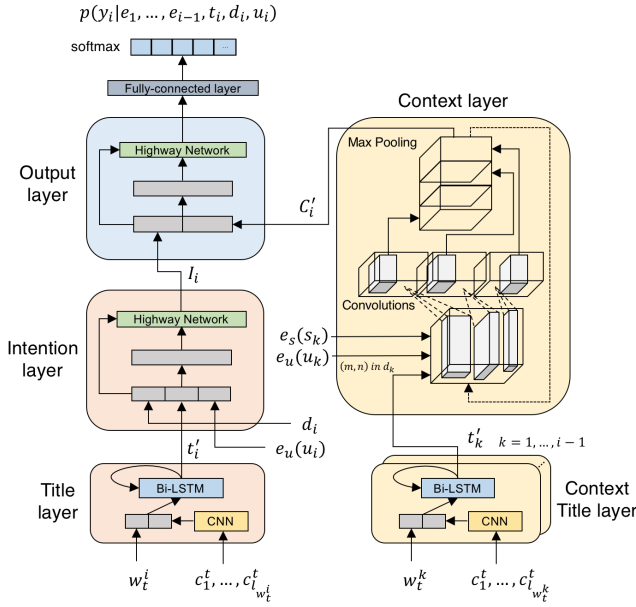
$$p(y_i | e_1, \ldots, e_{i-1}, t_i, d_i, u_i)$$

Figure 2: NESA overview. Given the title, duration, user attributes, and pre-registered events, NESA suggests suitable time slots for events.

4 METHODOLOGY

To deal with various types of raw calendar attributes, we propose NESA which consists of four different layers: 1) Title layer, 2) Intention layer, 3) Context layer, and 4) Output layer. The Title layer aims to represent the meaning of user written event titles using both the words and characters of the titles. In the Intention layer, our model utilizes title, duration, and user representations to learn user preferences and understand the purpose of events. The Context layer consists of multiple convolutional layers for understanding raw calendar contexts. Finally, the Output layer computes the probability of each time slot based on the representations from each layer. The architecture of NESA is illustrated in Figure 2.

4.1 Title Layer

RNNs have become one of the most common approaches for representing the meaning of written text [21]. Among the various RNN models, state-of-the-art NLP models for question answering [30] and named-entity recognition [20] often use not only word-level representations but also character-level representations as inputs. While word-level representations effectively convey semantic/syntactic relationships between words [22], character-level representations are widely used to represent unknown or infrequent words [16]. In event scheduling tasks, it is essential to use character-level representations for understanding personal calendars that have numerous pronouns or abbreviations.

Following previous works on question answering, we represent each title t_i using Bi-LSTM [14, 29] with pretrained word embeddings such as GloVe [27]. Given a title t_i comprised of T_i words, we map the words into a set of word embeddings $w_1^i, \ldots, w_{T_i}^i$. The

Title layer computes hidden state h_{T_i} of the LSTM as follows:

$$h_t = LSTM(w_t^i, h_{t-1}) \tag{3}$$

where h_t is the t-th hidden state of the LSTM which is calculated as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{i1} w_t + W_{i2} h_{t-1} + b_i) \\
f_t &= \sigma(W_{f1} w_t + W_{f2} h_{t-1} + b_f) \\
g_t &= \tanh(W_{g1} w_t + W_{g2} h_{t-1} + b_g) \\
o_t &= \sigma(W_{o1} w_t + W_{o2} h_{t-1} + b_o) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where we have omitted i from w_t^i for clarity and $\odot$ denotes element-wise multiplication. W_* and b_* are trainable parameters of the LSTM. LSTM is effective in representing long-term dependencies between distant inputs using input gate i and forget gate f.

The Title layer uses Bi-LSTM for title representations. With the forward LSTM giving the final hidden state $h_{T_i}^f$, we build the backward LSTM which computes its hidden states with reversed inputs. The backward LSTM's last hidden state denoted as h_1^b is concatenated with $h_{T_i}^f$ to form the title representation. The title representation will be denoted as $t_i' = [h_{T_i}^f, h_1^b] \in \mathbb{R}^T$.

On the other hand, the characters of each word with length l_{w_t} can be represented as a set of character embeddings $c_1^t, \ldots, c_{l_{w_t}}^t$. A common way to combine character embeddings into a word character representation is to use convolutions as follows:

$$f_k^c = \tanh(< C_{k:k+m-1}^t, F > +b) \tag{4}$$

where f_k^c is k-th element of a feature map f^c, $C_{k:k+m-1}^t$ is a concatenation of character embeddings from c_k^t to c_{k+m-1}^t, m is a convolution width, F is a filter matrix, and $< \cdot, \cdot >$ denotes the Frobenius inner product. Using max-over-time pooling [7], the single scalar feature is extracted as $f^c = \max_k f_k^c$. Given N types of filters and each of them having a different number of filters, resulting word character representations are obtained as $w_t^{c,i} = [f^{c,1}, \ldots, f^{c,N}]$ where $[\cdot, \cdot]$ denotes a vector concatenation, and $f^{c,n}$ is a concatenation of the outputs of n-th filters. We concatenate word representation w_t^i with word character representation $w_t^{c,i}$, which is inputted into the LSTM in Equation 3.

4.2 Intention Layer

Users have different intentions when registering a specific event. For instance, event titles that contain only personal names connote *meetings* to someone, but could mean *appointments* to others. To capture the intention of each user, we incorporate the title t_i, duration d_i, and user u_i attributes in the Intention layer. In this way, the Intention layer takes into account user preferences and purposes of events. In particular, we use the Highway Network that has some skip-connections between layers [31].[3] Given a title representation t_i' from a Title layer, duration d_i, and user u_i, the output of the

---

[3]While we could use Multi-Layer Perceptron (MLP) instead, the Highway Network achieved better performance in our preliminary experiments.

Highway Network I_i is as follows:

$$x = [t'_i, d_i, e_u(u_i)] \qquad (5)$$

$$q = \sigma(W_q x + b_q) \qquad (6)$$

$$I_i = q \odot f(W_h x + b_h) + (1 - q) \odot x \qquad (7)$$

where $e_u(\cdot) \in \mathbb{R}^U$ is an embedding mapping for each user. W_q, $W_h \in \mathbb{R}^{(T+U+1)\times(T+U+1)}$ are trainable parameters and f is a nonlinearity. Due to the skip-connection from x to I_i in addition to the nonlinearity, the Intention layer easily learns both linear and nonlinear relationships between calendar attributes.

4.3 Context Layer

We define a calendar context as a set of events that are pre-registered before the event e_i. We denote each pre-registered event as e_k where k is from 1 to $i-1$. Note that each user's week has a varying number of events from 0 to more than 50. Also, each pre-registered event e_k is comprised of different kinds of attributes such as start time, title, and duration. In the Context layer, we represent the calendar context by reflecting the status of the current week and scheduling preferences of users. Then, we use CNN to capture the local and global features of the calendar context to understand the calendar context representation.

4.3.1 Context Title Representation. For each title t_k in a pre-registered event e_k, we build a Context Title layer that processes only the titles of pre-registered events. Using Bi-LSTM and character-level CNN, each context title representation is obtained as t'_k. Note that multiple context title representations are obtained simultaneously in a mini-batch manner.

4.3.2 Calendar Context Representation. Given the context title representations $t'_k \in \mathbb{R}^T$, we construct a calendar context $C_i \in \mathbb{R}^{(M\times N)\times(T+U+S)}$ where U and S are dimensions of user and slot embeddings, respectively. M represents the number of days in a week, and N represents the number of hours in a day. Each depth is denoted as $C_i^{m,n} \in \mathbb{R}^{T+U+S}$ which is from m-th row (day) and n-th column (hour) of C_i. Each $C_i^{m,n}$ is constructed as follows:

$$C_i^{m,n} = [t'_{(m,n)}, e_u(u_i), e_s(s_{(m,n)})] \qquad (8)$$

$$t'_{(m,n)} = \left\{ \begin{array}{ll} t'_k & \text{if } (m,n) \text{ lies in } e_k\text{'s duration } d_k \\ \mathbf{0} \in \mathbb{R}^T & \text{otherwise} \end{array} \right\} \qquad (9)$$

where $e_u(\cdot) \in \mathbb{R}^U$ and $e_s(\cdot) \in \mathbb{R}^S$ are user and slot embedding functions, respectively, and $s_{(m,n)}$ is a slot representation on m-th day at n-th hour.

Given the calendar context C_i, the first convolution layer convolves C_i with 100 (1×1), 200 (3×3), 300 (5×5) filters, followed by batch normalization [15] and element-wise rectifier nonlinearity. We pad the calendar context to obtain same size outputs for each filter, and concatenate each output depth-wise. The second convolution layer consists of 50 (1×1), 100 (3×3), 150 (5×5) filters, followed by batch normalization and max-over-time pooling. As a result, we obtain the final calendar context representation $C'_i \in \mathbb{R}^{300}$.

Table 1: Event Scheduling Dataset Statistics

Statistics	Personal	Multi-Attendee
# of users	859	260
# of unseen users[4]	–	217
# of events	593,207	1,354
# of weeks	109,843	1,045
Avg. # of pre-registered events	6.9	22.2
Avg. # of attendees	–	2.1

4.4 Output Layer

Given a calendar context representation C'_i and an intention representation I_i, the Output layer computes the probability of each time slot in $M \times N$. We again adopt a Highway Network to incorporate the calendar context representation and the intention representation. Similar to Equations 5-7, given the input $x_o = [C'_i, I_i]$, the probability distribution of time slots is as follows:

$$z = q_o \odot f(W_h x_o + b_h) + (1 - q_o) \odot x_o \qquad (10)$$

$$p_j = softmax(W_o z + b_o)_j \qquad (11)$$

$$softmax(\boldsymbol{\alpha})_j = \frac{exp(\alpha_j)}{\sum_{j'} exp(\alpha_{j'})} \qquad (12)$$

where q_o is obtained in the same way as Equation 6 and j is from 1 to $M \times N$. We have used a single fully-connected layer for predicting the start time slot y_i of the event e_i. Given the outputs, the cross-entropy loss $CE(\Theta)$ of NESA is computed as follows:

$$CE(\Theta) = -\frac{1}{K} \sum_{i=1}^{K} \log p(y_i | e_1, \ldots, e_{i-1}, t_i, d_i, u_i; \Theta) \qquad (13)$$

where K denotes the number of events in a week. The model is optimized on the weeks in the training set. We use the Adam optimizer [17] to minimize Equation 13.

5 EXPERIMENT

5.1 Dataset

5.1.1 Preprocessing. We used Google Calendar[5] data collected between April 2015 and March 2018 by Konolabs, Inc. The format of the data is based on iCalendar, which is the most commonly used online calendar format. We detected and removed noisy events from the raw calendar data to reflect only real online calendar events. Events that we considered as noise are as follows:

- Events automatically generated by other applications (e.g., phone call logs, weather information, and body weight).
- Having an event title that has no meaning (e.g., empty string).
- All-day events, i.e., the events that will be held all day long.

Although some of the all-day events are legitimate events such as vacations or long-term projects, most of them are regular events whose start times have been simply omitted by users. We represented time slots as integers ranging from 0 to 167 where each time slot was considered as one hour in a week (i.e., 7 days × 24 hours). Only one event was selected given the overlapping events. The

---

[4]The number of users not seen in the personal event scheduling dataset.
[5]https://www.google.com/calendar

Table 2: Hyperparameters of MLP and NESA

Model	Parameter	Value
MLP	Hidden layer size	500
	# of hidden layers	2
	Learning rate	0.0005
NESA	LSTM cell hidden size	100
	# of LSTM layers	2
	LSTM dropout	0.5
	Day M, hour N	7, 24
	T, C, S, U	200, 30, 30, 30
	Learning rate	0.001

duration of each event is scaled to a scalar value from 0 to 1.

In Table 1, the second column shows the statistics of the personal event scheduling dataset after filtering. Though we carefully filtered calendar events, the dataset still had a considerable number of unrecognizable events (e.g., personal abbreviations). However, to test the ability of our proposed model, we did not perform any further filtering. We split the dataset into training (80%), validation (10%), and test (10%) sets, respectively.

In Table 1, the third column shows the statistics of the multi-attendee event scheduling dataset. Each event in the multi-attendee event scheduling dataset has at least two attendees, and attendees in each event are in the same time zone.[6] Due to the small number of multi-attendee events, we use them only as a test set for multi-attendee event scheduling. Also, we ensure that no events in the multi-attendee event scheduling dataset appear in the personal event scheduling dataset. As the multi-attendee event scheduling dataset has multiple attending users per event, it has more pre-registered events (22.2) than the personal event scheduling dataset (6.9). Note that both the personal and multi-attendee event scheduling datasets have a much larger number of users than the CAP dataset[7] which has events of only 2 users [23, 36].

5.1.2 Evaluation Metrics. We used various metrics to evaluate the performance of each model in event scheduling. Recall@N is the metric that determines if the correct time slot is in the top n predictions. Recall@1 and Recall@5 were mainly used. We also used Mean Reciprocal Rank (MRR) which is the mean of the inverse of the correct answer's rank. Also, motivated by the fact that suggesting time slots close to the correct answers counts as proper event scheduling, we used Inverse Euclidean distance (IEuc) [33] which calculates the inverse distance between predicted slots $\hat{y}_i$ and answer slots y_i in two-dimensional space in terms of days (m) and hours (n) as follows:

$$Euc(\hat{y}_i, y_i) = \sqrt{(\hat{y}_{i_m} - y_{i_m})^2 + (\hat{y}_{i_n} - y_{i_n})^2} \quad (14)$$

$$IEuc(\hat{y}_i, y_i) = \frac{1}{Euc(\hat{y}_i, y_i) + 1}. \quad (15)$$

5.2 Experimental Settings

5.2.1 Baseline Models. While recent automatic scheduling systems have proven to be effective on small sized datasets [8, 34, 36], it is difficult to directly apply their methodologies to our tasks for the following reasons: 1) some of them assume that user preferences are already given [34], 2) some use learning mechanisms based on systematic interactions with users [36], or 3) require human labor [8]. As a result, we use baseline models that are easily reproducible but still effective in our tasks.

In our study, the baselines are as follows: 1) a variant of CAP [23] using Random Forest (RF), 2) Support Vector Machine (SVM) [3, 13], 3) Logistic Regression (LogReg), and 4) Multi-Layer Perceptron (MLP). While RF and SVM are representative of previously suggested scheduling models, we further evaluate LogReg and MLP which are frequently adopted as classification baseline models.

As previous studies have focused on building interactive scheduling software, their learning algorithms rely largely on system dependent features such as event types, position of attendees, names of college classes, and so on [23]. As the iCalendar format does not contain most of these system dependent features, we used the attributes in Section 3.1 as inputs to the four baseline models. Besides categorical or real-valued features, event titles are represented as the average of pretrained word embeddings, and calendar contexts are given as binary vectors in which filled time slots are indicated as 1. For user representations, we used the normalized event start time statistics of each user (i.e., 168 dimensional vector whose elements sum to 1.) to reflect the scheduling preferences of each user. The representation of an unseen user is obtained using the average start time statistics of all the users in the training set.[8] The biggest difference between the baseline models and NESA is that the baseline models use a fixed set of hand-crafted features, whereas NESA learns to represent user preferences and calendar contexts for effective event scheduling.

5.2.2 Model Settings. While CAP uses a single decision tree for event scheduling, we constructed RF using thousand decision trees to build a more effective baseline model. The SVM model uses squared hinge losses and the one-vs-rest strategy for training. For LogReg, we used the SAGA optimizer [9]. Rectified linear unit (ReLU) [26] was used for MLP's activation function. Also for MLP, early stopping was applied based on the loss on the validation set, and we used the Adam optimizer for MLP. Both LogReg and MLP used L_2 regularizations to avoid overfitting.

The hyperparameters of MLP and NESA were chosen based on the MRR scores on the validation sets and the results are shown in Table 2. We used the same hyperparameters from [16] for character-level convolutions. A dropout of 0.5 was applied to the non-recurrent part of the RNNs of NESA to prevent overfitting [35]. We also clipped gradients when their norm exceeded 5 to avoid exploding gradients. Besides the character embedding, there are three additional embeddings in NESA: 1) word, 2) user, and 3) slot. We used pretrained GloVe[9] for word embeddings, and randomly initialized

---

[6]This can be easily extended to different time zone situations by shifting one of the time offsets.
[7]The CAP dataset contains system logs of Calendar Apprentice, which are difficult to convert to the iCalendar format.

[8]Each baseline feature representation was selected among various hand-crafted features based on our in-house experiments. For instance, statistics based user representation was better than one-hot user representation in terms of both event scheduling performance and generalization.
[9]For both NESA and baseline features, we used glove.840B.300d word embeddings.

Table 3: Personal Event Scheduling Results

Model	Recall@1	Recall@5	MRR	IEuc
RF [23]	0.0348	0.1483	0.0988	0.2520
SVM [3, 13]	0.0445	0.1762	0.1271	0.2619
LogReg	0.0442	0.1749	0.1279	0.2678
MLP	0.0442	0.1803	0.1277	0.2725
NESA	**0.0604**	**0.2156**	**0.1542**	**0.2881**

Table 4: Multiple Attendee Event Scheduling Results

Model	Recall@1	Recall@5	MRR	IEuc
RF [23]	0.0635	0.2585	0.0742	0.2389
SVM [3, 13]	0.0030	0.0340	0.0234	0.2530
LogReg	0.0037	0.0332	0.0260	0.2608
MLP	0.0406	0.1928	0.0773	0.2507
NESA	**0.0960**	**0.2740**	**0.1744**	**0.2950**

**Table 5: NESA Model Ablation
(Diff. %: average performance difference % of 4 metrics)**

Model	Recall@1	Recall@5	MRR	IEuc	Diff. %
NESA	0.0623	0.2289	0.1605	0.2910	–
- Context L.	<u>0.0419</u>	0.1789	<u>0.1083</u>	0.2668	<u>-23.9</u>
- Intention L.	0.0444	<u>0.1657</u>	0.1234	<u>0.2614</u>	-22.4
- Word E.	0.0561	0.2079	0.1476	0.2783	-7.9
- Character E.	0.0518	0.1974	0.1418	0.2836	-11.2
- Duration F.	0.0572	0.2049	0.1477	0.2820	-7.4
- User E.	0.0587	0.2125	0.1522	0.2889	-4.7

Table 6: Baseline Model Ablation

Model	Recall@1	Recall@5	MRR	IEuc	Diff. %
MLP	0.0445	0.1805	0.1283	0.2719	–
- Context F.	<u>0.0384</u>	<u>0.1624</u>	<u>0.1026</u>	<u>0.2582</u>	<u>-12.2</u>
- Word F.	0.0425	0.1710	0.1245	0.2661	-3.7
- Character F.	0.0433	0.1788	0.1271	0.2724	-1.1
- Duration F.	0.0433	0.1760	0.1256	0.2704	-2.0
- User F.	0.0440	0.1790	0.1269	0.2722	-0.7

embeddings for character, user, and slot embeddings. Word embeddings were fixed during optimization while other embeddings were optimized during training.

For training NESA, we used PyTorch with a CUDA enabled NVIDIA TITAN Xp GPU. The baseline models were trained using Scikit-learn. It took 8 hours of training for NESA to converge, which is quite short given the size of our training set and the complexity of NESA. NESA performs event scheduling as fast as baseline models by using mini-batches. We also experimented with increased number of layers and hidden dimensions in the MLP model so that it would have the same number of parameters as NESA (8.5M). However, the performance of the MLP model was lower than that of the MLP model trained on the best hyperparameters (7.0% in terms of MRR).

5.3 Quantitative Analysis

5.3.1 Personal Event Scheduling. The scores of personal event scheduling are presented in Table 3. The reported scores are average test set scores after ten separate trainings. The best scores are in bold. We first see that the performance ranking of the IEuc scores is similar to that of other metric scores such as the Recall@5 scores. This shows that the more a model accurately predicts an answer, the more it suggests nearby time slots around the correct answer. Among the baseline models, MLP performed the best on average, and RF achieved the lowest overall scores. However, despite MLP's deeper structure, performance improvements of MLP over LogReg were marginal, which shows the limitation of feature based models. NESA achieved higher scores than the baseline models in all metrics by learning to schedule directly using raw calendar data. NESA outperformed the baseline models by 29.6% on average in terms of MRR. More specifically, NESA outperformed MLP, which is the best baseline model, by 36.5%, 19.6%, 20.7%, and 5.7% in terms of Recall@1, Recall@5, MRR, and IEuc, respectively.

5.3.2 Multi-Attendee Event Scheduling. The performance results of the models on multi-attendee event scheduling are presented in

Table 4. The scores of each model are obtained by Equation 2. Compared to the performances on personal event scheduling, Recall@1 and Recall@5 of RF have been greatly improved, but MRR and IEuc of RF have been degraded. This verifies the limited effectiveness of decision tree based models as reported in the work of Mitchell et al. [23]. RF fails to provide precise probability distribution over time intervals, that reflects user preferences and calendar contexts, as MRR and IEuc are more sensitive to suggestion quality over the whole week. Other baseline models such as SVM, LogReg, and MLP have failed to produce meaningful results on multi-attendee event scheduling. We found that the huge performance degradation of these models comes from generalization failure on unseen users as most users (217 out of 260) in the multi-attendee event scheduling dataset are unseen during training on the personal event scheduling dataset. The performance of SVM, LogReg, and MLP on multi-attendee event scheduling was higher (but still insufficient compared to RF and NESA) when all the attendees were comprised of seen users during training.

NESA does not suffer from the unseen user problem by understanding raw online calendars to infer user preferences and understand calendar contexts. While preferences of known users can be encoded in user embeddings in NESA, preferences of unseen users can be inferred from their raw calendars. As with the personal event scheduling task, NESA outperforms the other baseline models by large margins on the multi-attendee event scheduling task. Specifically, NESA outperforms the best baseline model RF by 51.2%, 6.0%, 135.0%, and 23.5% in terms of Recall@1, Recall@5, MRR, and IEuc, respectively. This shows that using raw calendar data for understanding user preferences and calendar contexts is very important in event scheduling tasks.

5.3.3 NESA Model Ablation and Analysis. To analyze the architecture of NESA, we removed each layer or component of NESA. The results are shown in Table 5. When the Context layer is removed,
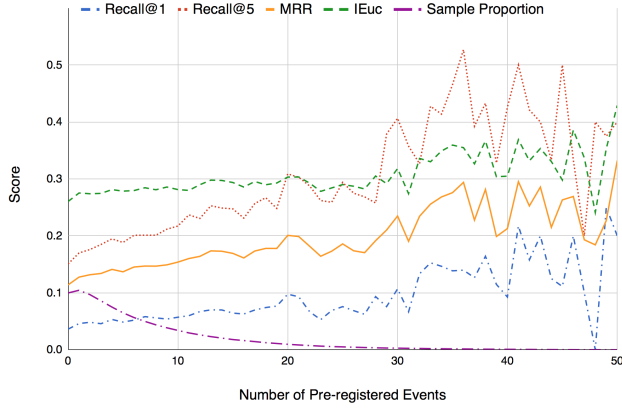
Figure 3: Performance changes with different numbers of pre-registered events in NESA.
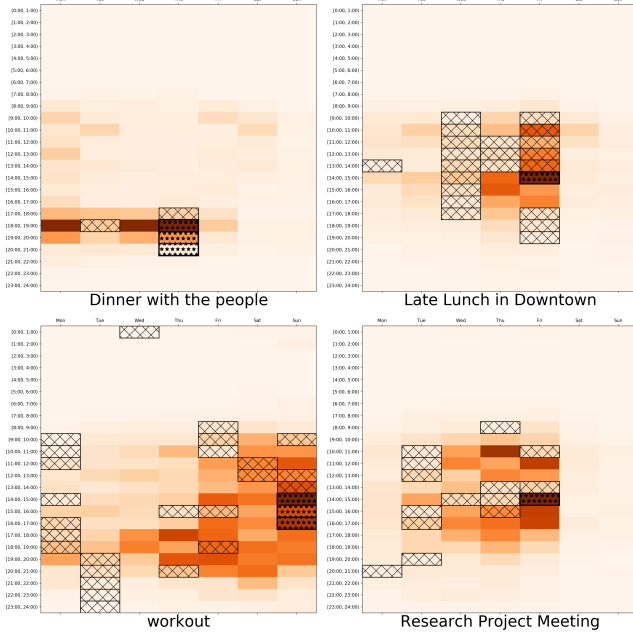

Figure 4: Output probabilities of NESA given different titles.

the Output layer receives only the intention representation. We feed the title representation instead of the intention representation to the Output layer when the Intention layer is removed. The Context layer has the most significant impact on the overall performance. The Intention layer also shows that incorporating user and duration attributes with title attributes is crucial for event scheduling. The character embedding has substantial effects on the performance.

To demonstrate the Context layer's impact, we illustrate the changes in performance of NESA based on different numbers of pre-registered events in Figure 3. As the number of pre-registered events grows, overall performance improves. Note that the sampling proportion decreases as the number of pre-registered events increases, which causes a high variance in performance.

Table 7: Nearest Neighbors (NNs) of Title Representations Given the Title `Family lunch`

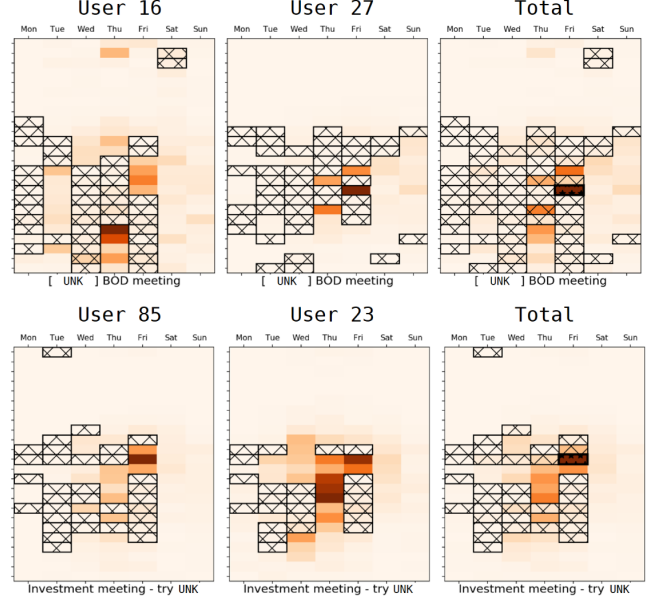| MLP NNs	NESA NNs
Family Dinner out	Birthday <u>lunch</u>
Family Dinner	Themed <u>Lunch</u>
<u>Lunch</u> with Family Friends	UNK / BDP <u>lunch</u>
family dinner	Hope <u>lunch</u>

Figure 5: Output probabilities of NESA in multi-attendee meeting scheduling.

5.3.4 Baseline Model Ablation. Although the performance of the baseline models is lower than that of NESA, models such as MLP still achieve reasonable performance. We present the ablated MLP model in Table 6 and compare all its features to determine which feature contributes the most to the performance. We removed each feature one by one, and retrained the MLP model. We found that the MLP model, like NESA, largely depends on the context feature. It seems that MLP tends to choose empty slots based on the context features.

5.4 Qualitative Analysis

5.4.1 Effect of the Title Layer. Given different titles, NESA assigns different probabilities to each slot. In Figure 4, we visualized the output probabilities of NESA given four different input titles. The rows of each heatmap denote the hours in a day, and the columns denote the days in a week. The filled time slots are marked with lattices and the answers are marked with stars. For the title "Dinner with the people," NESA suggests mostly night times. Also, for the title "Late Lunch in Downtown," NESA not only suggests late lunch hours, but it also chooses days that the user may be in downtown. *Workout* and *Meeting* are more ambiguous keywords than *Lunch* or

Table 8: Nearest Neighbors of Title/Intention Representations Given the Title `App project work` (duration 120 min.)

| Title layer	Intention layer		
	User A (Duration 120 min.)	**User B** (Duration 120 min.)	User A (**Duration 240 min.**)
App project work (120)	Make V1 of app (120)	Create paperwork for meetings (60)	Meet Databases Team (240)
App work (540)	Do Databases project (120)	Try Fontana again (60)	App work (540)
App Description to Richard (60)	Databases (120)	Try Peter @ UNK again (60)	Watch databases, do algorithmics (240)
App w Goodman (60)	UNK and spot market (120)	Try pepper Jaden Mark (60)	Databases Final Meeting (180)
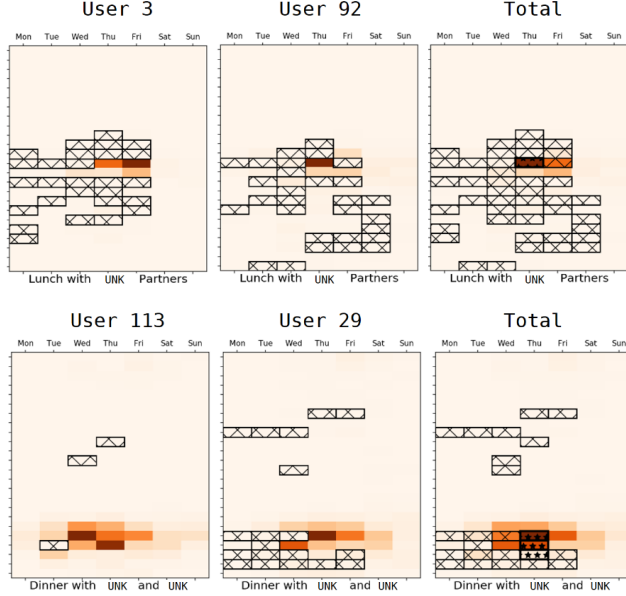
Figure 6: Output probabilities of NESA in multi-attendee event scheduling given lunch and dinner events.

Figure 7: Output probabilities of NESA in multi-attendee event scheduling given misspelled and non-English events.

Dinner, but NESA suggests again suitable time slots based on each title. Figure 4 shows *workout* is done on weekends or at evening-time while *Meetings* are held during office hours.

In Table 7, we show the 4 nearest neighbors of title representations of MLP and NESA. The distances between each representation were calculated using the cosine similarity. MLP's title representation is the element-wise average of word embeddings, and NESA uses the Title layer for title representations. With the title "Family lunch," we observe that MLP's title representations do not differentiate each keyword in event scheduling. Although the keyword *lunch* should have more effect on event scheduling, most nearest neighbors of MLP's title representation are biased towards the keyword *Family*, while nearest neighbors of NESA's title representation are mostly related to *lunch*.

5.4.2 Effect of the Intention Layer. The Intention layer in NESA combines different types of attributes from calendar data. In Table 8, we present the 4 nearest neighbors of the title and intention representations based on the cosine similarities. Given the title "App project work," the Title layer simply captures semantic representations of the title. Titles with similar meanings such as "App
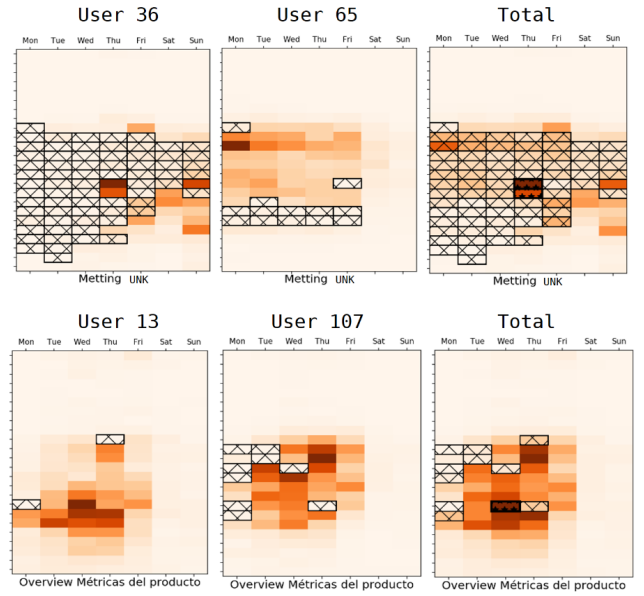
work" are its nearest neighbors (1st column). On the other hand, the nearest neighbors of the intention representation are related to not only the keyword *app* but also the keyword *database*, which is one of user A's frequent terms (2nd column). We observe that the intention representation changes by replacing user A with user B who frequently uses the term *Try* (3rd column). The duration attribute is also well represented as events with longer durations are closer to user A's 240 minute long event (4th column).

5.4.3 Multi-Attendee Event Scheduling Analysis. In Figures 5–7, we present examples of multi-attendee event scheduling. Using NESA, we obtain each user's preferred time slots, and the suggested time slots for multi-attendee events are calculated by Equation 2. Again, the filled time slots are marked with lattices and the answers are marked with stars. We show a multi-attendee event in each row, and each row contains the preferences of two different users and their summed preference (total). We anonymized any pronouns as *UNK* tokens for privacy issues.

Figure 5 shows examples of event scheduling for meetings. The two examples clearly show that NESA understands each user's calendar context, and suggests time intervals mostly during office

hours. Figure 6 shows appointments such as lunch and dinner rather than meetings. While each example accurately represents the purpose of each event, note that NESA does not suggest weekends for "Lunch with UNK Partners." We think that NESA understands the keyword *Partner*, which is frequently related to formal meetings. In Figure 7, we show how misspellings (*Metting* for *meeting*) and non-English ("Métricas del producto" means "product's metric" in Spanish) are understood by NESA. As NESA has the Title layer that leverages the characters of infrequent words, NESA successfully suggests suitable office hours for each event.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel way to fully make use of raw online calendar data for event scheduling. Our proposed model NESA learns how to perform event scheduling directly from raw calendar data, and to consider user preferences and calendar contexts. We also showed that deep neural networks are highly effective in scheduling events. Unlike previous works, we leveraged a large-scale online calendar dataset in the Internet standard format, which makes our approach more applicable to other systems. NESA achieves the best performance among the existing baseline models on both personal and multi-attendee event scheduling tasks.

For future work, we plan to study the relationships between users for multi-attendee event scheduling. Unfortunately, such relationship information is not provided in the standard calendar format, and should be inferred from multi-attendee event scheduling examples. Once we obtain more multi-attendee calendar events, such an approach would produce more sophisticated multi-attendee scheduling systems.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). http://arxiv.org/abs/1409.0473

[2] Pauline Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. 2007. *Balancing the needs of personalization and reasoning in a user-centric scheduling assistant.* Technical Report. Artificial Intelligence Center, SRI International.

[3] Pauline Berry, Melinda Gervasio, Bart Peintner, and Neil Yorke-Smith. 2011. PTIME: Personalized assistance for calendaring. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 4 (2011), 40.

[4] Pauline Berry, Melinda Gervasio, Tomas Uribe, Karen Myers, and Ken Nitz. 2004. A personalized calendar assistant. In *Working notes of the AAAI Spring Symposium Series*, Vol. 76.

[5] Peter Bjellerup, Karl J Cama, Mukundan Desikan, Yi Guo, Ajinkya G Kale, Jennifer C Lai, Nizar Lethif, Jie Lu, Mercan Topkara, and Stephan H Wissel. 2010. FALCON: Seamless access to meeting data from the inbox and calendar. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management.* ACM, 1951–1952.

[6] Avrim Blum. 1997. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning* 26, 1 (1997), 5–23.

[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[8] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. 2017. Calendar.help: Designing a workflow-based scheduling agent with humans in the loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, 2382–2393.

[9] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems.* 1646–1654.

[10] Bernard Desruisseaux. 2009. *Internet calendaring and scheduling core object specification (iCalendar).* Technical Report.

[11] Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *COLING.* 69–78.

[12] Leonardo Garrido and Katia Sycara. 1996. Multi-agent meeting scheduling: Preliminary experimental results. In *Proceedings of the Second International Conference on Multiagent Systems.* 95–102.

[13] Melinda T Gervasio, Michael D Moffitt, Martha E Pollack, Joseph M Taylor, and Tomas E Uribe. 2005. Active preference learning for personalized calendar scheduling assistance. In *Proceedings of the 10th international conference on Intelligent user interfaces.* ACM, 90–97.

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

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15).* JMLR.org, 448–456.

[16] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-Aware Neural Language Models. In *AAAI.* 2741–2749.

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

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems.* 1097–1105.

[19] Saritha Kuruvilla. 2011. An in-depth look at the usage of calendars in the U.S. workplace, particularly the use of advertising calendars. Retrieved May 20, 2018 from http://www.ppai.org/documents/business%20study%20final%20report%20version%204.pdf

[20] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. *CoRR* abs/1603.01360 (2016). http://arxiv.org/abs/1603.01360

[21] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, Vol. 2. 3.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems.* 3111–3119.

[23] Tom M Mitchell, Rich Caruana, Dayne Freitag, John McDermott, David Zabowski, et al. 1994. Experience with a learning personal assistant. *Commun. ACM* 37, 7 (1994), 80–91.

[24] David Montoya, Thomas Pellissier Tanon, Serge Abiteboul, and Fabian M Suchanek. 2016. Thymeflow, a personal knowledge base with spatio-temporal data. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* ACM, 2477–2480.

[25] Elizabeth Mynatt and Joe Tullio. 2001. Inferring calendar event attendance. In *Proceedings of the 6th international conference on Intelligent user interfaces.* ACM, 121–128.

[26] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10).* 807–814.

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*, Vol. 14. 1532–43.

[28] Ioannis Refanidis and Neil Yorke-Smith. 2009. On scheduling events and tasks by an intelligent calendar assistant. In *Proceedings of the ICAPS Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems.* 43–52.

[29] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[30] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).

[31] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* (2015).

[32] Yu Sun, Nicholas Jing Yuan, Yingzi Wang, Xing Xie, Kieran McDonald, and Rui Zhang. 2016. Contextual intent tracking for personal assistants. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 273–282.

[33] Segaran Toby. 2007. Programming Collective Intelligence. (2007), 11.

[34] Jacques Wainer, Paulo Roberto Ferreira Jr, and Everton Rufino Constantino. 2007. Scheduling meetings through multi-agent negotiations. *Decision Support Systems* 44, 1 (2007), 285–297.

[35] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).

[36] Alejandro Zunino and Marcelo Campo. 2009. Chronos: A multi-agent system for distributed automatic meeting scheduling. *Expert Systems with Applications* 36, 3 (2009), 7011–7018.