Dual Learning for Semi-Supervised Natural Language Understanding
Su Zhu, Ruisheng Cao, and Kai Yu

Abstract—Natural language understanding (NLU) converts sentences into structured semantic forms. The paucity of annotated training samples is still a fundamental challenge of NLU. To solve this data sparsity problem, previous work based on semi-supervised learning mainly focuses on exploiting unlabeled sentences. In this work, we introduce a dual task of NLU, semantic-to-sentence generation (SSG), and propose a new framework for semi-supervised NLU with the corresponding dual model. The framework is composed of dual pseudo-labeling and dual learning method, which enables an NLU model to make full use of data (labeled and unlabeled) through a closed-loop of the primal and dual tasks. By incorporating the dual task, the framework can exploit pure semantic forms as well as unlabeled sentences, and further improve the NLU and SSG models iteratively in the closed-loop. The proposed approaches are evaluated on two public datasets (ATIS and SNIPS). Experiments in the semi-supervised setting show that our methods can outperform various baselines significantly, and extensive ablation studies are conducted to verify the effectiveness of our framework. Finally, our method can also achieve the state-of-the-art performance on the two datasets in the supervised setting.

Index Terms—Natural language understanding, semi-supervised learning, dual learning, slot filling, intent detection.

I. INTRODUCTION

RECENTLY, the development of mobile internet and smart devices has led to the tremendous growth of conversational dialogue systems, such as Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana. Natural language understanding (NLU) is a key component of these systems, parsing user’s utterances into the corresponding semantic forms [1] for certain narrow domain (e.g., booking hotel, searching flight). Typically, the primary task of the NLU module in goal-oriented dialogue systems usually contains two sub-tasks: intent detection and slot filling [1], [2], [3], [4], [5], [6], [7]. The intent detection is typically treated as a sentence classification problem [8], [9], [10], while the slot filling is typically treated as a sequence labeling problem in which contiguous sequences of words are tagged with semantic labels (slots) [11], [12], [13], [14].

Deep learning has achieved great success for the intent detection and slot filling in NLU [2], [3], [4], [5], [6], [7], [14], [15], [16], [17], [18], [19], outperforming most traditional approaches [13], [20] in the field of supervised learning.

Su Zhu, Ruisheng Cao and Kai Yu are with the SpeechLab and MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. (Su Zhu and Ruisheng Cao contribute equally to this article.) (Corresponding authors: Kai Yu.) (e-mail: paul2204@sjtu.edu.cn; 211314@sjtu.edu.cn; kai.yu@sjtu.edu.cn)

However, the deep learning method is notorious for requiring large labeled data, which limits the scalability of NLU models to new domains due to the annotation cost. Semi-supervised learning methods are adopted to solve this data sparsity problem of NLU, which utilize a large number of unannotated sentences to enhance the supervised NLU training [21], [22], [23], [24]. These semi-supervised learning methods focus on exploiting the unlabeled sentences to enhance input encoders or create additional samples with predicted pseudo-labels.

Apart from pure sentences (i.e., unannotated sentences), pure semantic forms (i.e., intents and slots without sentence expressions) can also be utilized in the semi-supervised NLU. Exploiting semantic forms could be more affordable and effective than collecting in-domain sentences, since they are well-structured and could be automatically created or synthesized under domain knowledge. However, the previous methods of semi-supervised NLU cannot utilize pure semantic forms data.

In this work, we introduce the dual task of intent detection and slot filling in NLU, as shown in Fig. 1. By incorporating the dual task, a novel framework of semi-supervised NLU is proposed, which can utilize not only pure sentences but also pure semantic forms (i.e., intents and slots). Our framework consists of two parts: a dual pseudo-labeling method and dual learning algorithm. 1) Besides using a primal model to generate pseudo labels [22] for unlabeled sentences, the dual pseudo-labeling method also utilizes a dual model to generate pseudo sentences for pure semantic forms. Next, we combine these pseudo-labeled samples with the labeled dataset to retrain both the primal and dual models iteratively. 2) Furthermore, the dual learning algorithm [25] is applied to train the primal and dual models jointly in a closed-loop of the two models. New validity rewards are proposed to validate potential sentences and semantic forms.

The main contributions of this paper are summarized:

- A dual model for joint intent detection and slot filling...
in NLU is introduced to generate sentences based on structured semantic forms.

- We propose a novel framework for semi-supervised NLU by incorporating the dual model, which can better utilize unlabeled data.
- We present extensive experiments on ATIS [26] and SNIPS [27] datasets, which demonstrate the benefit of our proposed framework for semi-supervised NLU. It also achieves the state-of-the-art performance in the supervised setting.

The rest of the paper is organized as follows. The following section discusses related works. We introduce the intent detection and slot filling in NLU in Section III, then describe the details of the dual task in Section IV. A semi-supervised NLU framework with the dual task is proposed in Section V. Detailed experimental results and analysis are given in Section VI. Section VII summarizes this work and the future direction.

II. RELATED WORK

This section describes previous literature of intent detection and slot filling in NLU as well as the semi-supervised NLU.

A. Intent Detection and Slot Filling in NLU

Recently, motivated by a number of successful neural network and deep learning methods in natural language processing, many neural network architectures have been applied in the intent detection and slot filling, such as vanilla recurrent neural network (RNN) [14], [28], [29], [30], convolutional neural network (CNN) [16], [2], [31], long short-term memory (LSTM) [15], [4], [32], [33], encoder-decoder [18], [3], [34], [17], capsule neural networks [35], transformers [36], etc. Several pre-trained language models are also applied to improve generalization, like ELMo [37] and BERT [6], [38]. Most of the previous work tends to share the encoders of the intent detection and slot filling while leaves their decoders (e.g., classification layers) independent. Besides, some investigations focus on interrelated modeling of intent detection and slot filling [19], [5], [7], [39], which is orthogonal to the semi-supervised learning of NLU.

B. Semi-supervised NLU

The traditional approaches of semi-supervised NLU utilize unlabeled sentences to improve NLU performances in two ways: 1) NLU model trained with the existing labeled sentences is exploited to predict pseudo-labels for unlabeled sentences, which can be used to retrain the NLU model [21], [22], [23]. 2) Except for the pseudo-labeling method, some prior works design several unsupervised tasks to make use of the unlabeled sentences, like language models [40], [41], [24], [37], sequence-to-sequence based sentence reconstruction [42], [43]. They share partial parameters between the unsupervised tasks and the NLU task. However, we are the first to exploit pure semantic forms (without sentence expressions) by developing a dual pseudo-labeling method.

The dual learning algorithm is first proposed for neural machine translation [25], where translation from the target language to source language (i.e., back-translation) is the dual task. The dual learning is also applied in semantic parsing [44], [45] and natural language understanding [45]. As the most similar work, Su et al. [45] propose a dual supervised learning method for natural language understanding and generation. However, their method is not compatible with a semi-supervised problem. Moreover, they simplify the NLU task into a multi-label classification problem, which is not scalable. We are the first to propose a dual task for intent detection and slot filling in NLU and utilize the dual task in semi-supervised NLU.

III. INTENT DETECTION AND SLOT FILLING IN NLU

This section introduces the NLU task and describes the basic multitask framework of intent detection and slot filling.

A. NLU Task Formulation

Intent detection and slot filling are major tasks of NLU in task-oriented dialogue systems. An intent is a purpose or a goal that underlies a user-generated utterance [46]. Therefore, intent detection can be seen as a classification problem to determine the intent label of an input sentence. Slot filling aims to automatically extract a set of attributes or “slots”, with the corresponding values. It is typically treated as a sequence labeling problem. An example of data annotation is provided in Fig. 2. The user’s intent is to find flights. For slot annotation, it follows the popular inside/outside/beginning (IOB) schema, where Boston and New York are the departure and arrival cities specified as the slot values in the user’s utterance, respectively. In this work, we use the word tag as an alias for slot to denote semantic labels in IOB schema.

| Sentence | Show | me | flights | from | Shanghai | to | New | York |
|----------|------|----|---------|------|-----------|----|-----|------|
| Slots    | O    | O  | O       | O    | B-FronCity| O  | B-ToCity| I-ToCity |

Intent   Find_Flight

Fig. 2. An example of intent and slot annotation (IOB format) in ATIS dataset.

Let \( x = (x_1, \ldots, x_{|x|}) \) denote an input sentence (word sequence), \( o^I \) denote its intent label, and \( o^S = (o^S_1, \ldots, o^S_{|x|}) \) denote its output sequence of slot tags, where \( |x| \) is the sequence length. Each \( o^S_i \in \mathcal{T} \) and \( o^I \in \mathcal{I} \), where \( \mathcal{T} \) and \( \mathcal{I} \) are the sets of all possible slot tags and intent labels respectively in the current domain. Therefore, the intent detection and slot filling in NLU are to estimate \( p(o^I, o^S|x) \), the joint posterior probability of intent \( o^I \) and slot sequence \( o^S \) given input \( x \). Usually, the two sub-tasks are modelled independently, i.e.,

\[
p(y|x) = p(o^I, o^S|x) = p(o^I|x)p(o^S|x)
\]

where \( y = (o^I, o^S) \).

B. Preliminaries for Neural Network

Before providing details of the NLU model, we first introduce two basic NN modules for conciseness.

BLSTM: As mentioned before, many neural network architectures have been applied in intent detection and slot filling tasks. In this paper, bi-directional LSTM based RNN
(BLSTM) is adopted for sequence encoding. Given a sequence of feature vectors \( (e_1, \cdots, e_L) \), hidden vectors are recursively computed at the \( i \)-th time step \((i \in \{1, \cdots, L\}) \) via
\[
\overrightarrow{h}_i = f_{\text{LSTM}}(e_i, \overrightarrow{h}_{i-1}); \overleftarrow{h}_i = f_{\text{LSTM}}(e_i, \overleftarrow{h}_{i+1})
\]
and \( h_i = \overrightarrow{h}_i \oplus \overleftarrow{h}_i \), where \( \oplus \) denotes the vector concatenation and \( f_{\text{LSTM}} \) is the LSTM function. For convenience, we rewrite the entire operation as a mapping BLSTM\(_{\Theta} \):
\[
(h_1, \cdots, h_L) \leftarrow \text{BLSTM}_{\Theta}(e_1, \cdots, e_L)
\]

**Attention Mechanism:** Attention mechanism \([47],[48]\) is usually used to obtain a sequence-level feature vector or context representations in encoder-decoder architectures. Given a sequence of feature vectors \( (e_1, \cdots, e_L) \) and a query vector \( q \), the attention weight for \( q \) with each \( e_i \) \((i \in \{1, \cdots, L\}) \) is
\[
a_i = \exp(u_i) / \sum_{i=1}^L \exp(u_i), \quad u_i = v^\top a \tanh(W_a(q \oplus e_i))
\]
where \( v_a \) and \( W_a \) are learnable parameters. Finally, a context vector is computed as \( z = \sum_{i=1}^L a_i e_i \). For brevity, we rewrite the entire operation as a mapping returning the context vector and attention weights:
\[
z_i(a_1, \cdots, a_L) \leftarrow \text{ATTN}_{\Theta}(q, (e_1, \cdots, e_L))
\]

**C. NLU Model Architecture**

The basic multitask framework of intent detection and slot filling is comprised of three modules: sentence encoding, intent classification, and slot tagging.

**Sentence Encoding:** Every input word is mapped to a vector via \( x_i = W_x o(x_i) \), where \( W_x \) is an embedding matrix and \( o(x_i) \) a one-hot vector. An BLSTM encoder is adopted to get hidden vectors \( h_i^{\text{sen}} \in \mathbb{R}^{2n} \) \((n \text{ is the hidden size, } i \in \{1, \cdots, |x|\}) \):
\[
(h_1^{\text{sen}}, \cdots, h_{|x|}^{\text{sen}}) \leftarrow \text{BLSTM}_{\Theta_1}(x_1, \cdots, x_{|x|})
\]

**Intent Classification:** An attention model is applied to gather a sentence embedding, and then feed it into a linear output layer for intent classification:
\[
z_{\text{sen}}^{\text{sen}}(a_1^{\text{sen}}, \cdots, a_{|x|}^{\text{sen}}) \leftarrow \text{ATTN}_{\Theta_2}(h_1^{\text{sen}}, \cdots, h_{|x|}^{\text{sen}})
\]
\[
p(o^I|x) = \text{softmax}_{o^I}(W_1 z_{\text{sen}}^{\text{sen}})
\]
where \( W_1 \in \mathbb{R}^{|I| \times 2n} \) is trainable (bias is omitted).

**Slot Tagging:** Slot filling is considered as a sequence labeling problem which tags each input word sequentially. There are three typical methods for slot tagging concerning the time series dependence of slot tags, as shown below.

1) **BLSTM-softmax:** At each time step, a linear output layer is applied to predict slot tags independently, i.e.
\[
p(o^S|x) = \prod_{i=1}^{|x|} p(o_i^S|h_i^{\text{sen}}) = \prod_{i=1}^{|x|} \text{softmax}_{o^S}(W_2 h_i^{\text{sen}})
\]
where \( W_2 \in \mathbb{R}^{|T| \times 2n} \) is trainable, and \( \text{softmax}_{o^S}(.) \) is precisely the \( o^S \)-th element of the distribution defined by the softmax function.

2) **BLSTM-CRF:** CRF output layer considers the correlations between tags in neighborhoods and jointly decode the best chain of tags for a given input sentence \([49],[50],[51]\). The posterior probability of slot sequence is computed via:
\[
\psi(x, o^S) = \sum_{i=1}^{|x|} ([A]_{o_{i-1}^S, o_i^S} + [W_2 h_{i}^{\text{sen}}]_{o_i^S})
\]
\[
p(o^S|x) = \frac{\exp(\psi(x, o^S))}{\sum_{o^S} \exp(\psi(x, o^S))}
\]
where \( A \in \mathbb{R}^{|T| \times |T|} \) is a transition matrix, and its element \([A]_{m,n} \) models the transition from the \( m \)-th to the \( n \)-th label for a pair of consecutive time steps.

3) **BLSTM-focus:** To consider the time series dependence of slot tags, several encoder-decoder architectures \([17],[18],[3],[34]\) are also proposed for slot filling. With the focus mechanism \([18]\), we utilize a uni-directional LSTM based decoder to model tag dependencies. The decoders hidden vector at the \( i \)-th time step is computed by \( h_i^\text{tag} = f_{\text{LSTM}}(h_i^{\text{sen}} \oplus o_i^S, h_i^\text{tag}_{i-1}) \), where \( o_i^S \) is the embedding of the previously predicted slot tag, and \( h_i^\text{tag} = \overrightarrow{h}_i \). Then we compute \( p(o^S|x) \) via:
\[
p(o^S|x) = \prod_{i=1}^{|x|} p(o_i^S|o_{i-1}^S, x) = \prod_{i=1}^{|x|} \text{softmax}_{o^S}(W_3 h_i^\text{tag})
\]
where \( W_3 \in \mathbb{R}^{|T| \times n} \). Compared with **BLSTM-CRF**, this method can model longer-range dependence of slot tags.

**Slot-Value Summary:** Though we can get a sequence of slot tags after slot tagging, it is a semifinished representation that the value for each predicted slot is not revealed. With alignment between the predicted tag sequence and input sentence, we can extract a summary of slot-value pairs easily. For instance, the list of slot-value pairs for the sample in Fig. 2 is \((\text{FromCity}=\text{Shanghai}, \text{ToCity}=\text{New York})\). Let \( o^C \) denote the list of slot-value pairs, and then we can get the final semantic form \( y \) of the input \( x \):
\[
y = (o^I, o^C) = (o^I, \text{getSummary}(o^S, x))
\]
The loss function of the NLU model given \( x \) and \( y \) is
\[
\mathcal{L}_{\text{NLU}}(x, y) = - \log p(y|x) = - \log p(o^I|x) - \log p(o^S|x)
\]

**IV. DUAL TASK OF NLU**

In this section, we will introduce the dual task of NLU, which is formulated as a semantic-to-sentence generation (SSG) task. It generates the corresponding sentence \( x \) given an intent \( o^I \) and a list of slot-value pairs \( o^C = (o_1^C, \cdots, o_M^C) \) \((M \text{ is the number of slot-value pairs})\). As a fact of the IOB annotation schema, each value in \( o^C \) must appear in \( x \) without overlapping. Thus, we choose to first generate a delexicalized form \( \tilde{x} \) comprised of words and slots, and then fill the slots up with given values in \( o^C \) to get \( x \). We wish to estimate
\[
p(\tilde{x}|y) = p(\tilde{x}|o^I, o^C)
\]
, the conditional probability of delexicalized form \( \tilde{x} \) given intent \( o^I \) and slot-value pairs \( o^C \).

1For example, the delexicalized form of the sentence in Fig. 2 is "show me flights from (FromCity) to (ToCity)".
Sequence-to-sequence based encoder-decoder architectures has achieved success in natural language generation such as machine translation [47], [48], dialogue generation [52] and text summarization [53]. However, it is non-trivial to apply the encoder-decoder architectures into SSG, since the input of SSG is not a sequence any more but a structured form (i.e., an intent and a list of slot-value pairs).

The proposed architecture for SSG is illustrated in Fig. 3. An encoder is exploited to encode the intent $\delta^I$ and the list of slot-value pairs $\delta^C$ into vector representations, and a decoder learns to generate the delexicalized form $\tilde{x}$ depending on the encoding vectors. Finally, we replace the slots in $\tilde{x}$ with the corresponding values in $\delta^C$.

**Encoder:** We exploit a hierarchical BLSTM to encode the list of slot-value pairs at local and global levels. Firstly, each slot-value pair is considered as a sub-sequence, i.e., $\delta^C = (c_{m1}, \cdots, c_{mT_m})$, where $T_m$ is the sequence length, and $m \in \{1, \cdots, M\}$. For example, a slot-value pair, $\text{ToCity}=\text{New York}$, is tokenized as (“ToCity”, “New”, “York”).

1) **Local-level:** For each slot-value pair $c_{m}^C$, we use a shared BLSTM to get local representations independently:

$$h_{m1}^C, \cdots, h_{mT_m}^C \leftarrow \text{BLSTM}_{\Theta_1}(c_{m1}, \cdots, c_{mT_m}) \tag{14}$$

where $c_{mj}$ is the embedding of $j$-th token in $c_{m}^C$. The local representation of $c_{m}^C$ is defined as $o_{m}^C = \frac{1}{T_m} \sum_j h_{mj}^C$, $o_{m}^C \in \mathbb{R}^{2n}$.

2) **Global-level:** Upon the local representations of all slot-value pairs in $\delta^C$, another BLSTM is applied to get global hidden features, $h_m^g \in \mathbb{R}^{2n}$, $m \in \{1, \cdots, M\}$:

$$h_1^g, \cdots, h_M^g \leftarrow \text{BLSTM}_{\Theta_2}(o_1^C, \cdots, o_M^C) \tag{15}$$

**Decoder:** In order to avoid generating redundant or missing slots in the prediction of sequence $\tilde{x}$, the semantically controlled LSTM (SC-LSTM) [54] is applied. The hidden vector at the $t$-th time step is computed by $(d_t, s_t) = f_{\text{SC-LSTM}}(\tilde{x}_{t-1} \oplus \delta^I, (d_{t-1}, s_{t-1}))$, where $\tilde{x}_{t-1}$ is the embedding of the previously predicted token, $\delta^I$ is the embedding of the given intent, and $d_t$ is the hidden vector. Compared with LSTM, the SC-LSTM contains a slot-value state $s_t$ which plays the role of sentence planning. $s_t$ manipulates the slot-value features during the generation process in order to produce a hidden vector which accurately encodes the input semantics. The slot-value state is initialized with the original slots 1-hot vector $s_0$ where each element is zero except for the slots in $\delta^C$.

Additional regularization term will be added to the final loss function for each sample $(s|\tilde{x})$ is the final slot state vector:

$$\mathcal{L}_{\text{SC}} = \|s|\tilde{x}||_2 + \sum_{i=1}^{||\tilde{x}||} \eta \|s_i - s_{t-1}\|_2$$

where $\eta = 10^{-4}$, $\| \cdot \|_2$ is $l_2$ norm. The first term is used to penalise generated sequences that failed to render all the required slots, while the second term discourages the decoder from turning more than one slot off in a single time step.

The hidden vector is initialized by the aggregated encoding vectors, i.e. $d_0 = W_0 z_0^{\text{sem}}$, where $W_0 \in \mathbb{R}^{n \times 2n}$, and $z_0^{\text{sem}} \in \mathbb{R}^{2n}$ is an attention vector of the encoder hidden, i.e., $z_0^{\text{sem}}, (a_{01}^{\text{sem}}, \cdots, a_{0M}^{\text{sem}}) \leftarrow \text{ATTN}_{\Theta_5}(o^I, (h_1^g, \cdots, h_M^g))$ \tag{16}

An output layer with the attention mechanism [48] and the copying mechanism [53] is applied on the SC-LSTM to predict tokens in $\tilde{x}$. The attention weight for the current step $t$ of the decoder with the $m$-th slot-value pair in the encoder ($m \in \{1, \cdots, M\}$) and the attention vector are computed via:

$$z_t^{\text{sem}}, (a_t^{\text{sem},1}, \cdots, a_t^{\text{sem},M}) \leftarrow \text{ATTN}_{\Theta_5}(d_t, (h_1^g, \cdots, h_M^g)) \tag{17}$$

Then we compute the vocabulary distribution

$$p_{\text{gen}}(\tilde{x} | x_{<t}, y) = \text{softmax}_{|V_x|} (W_v (d_t \circ z_t^{\text{sem}}))$$ \tag{18}$$

where $W_v \in \mathbb{R}^{|V_x| \times 3n}$, and $|\tilde{x}|$ is the output vocabulary size. Generation terminates once an end-of-sequence token “EOS” is emitted.
Except for directly generation, the decoder also includes the copying mechanism to improve model generalization, which copies slots from the slot-value pairs in \( o^C \). We use sigmoid gate function \( \sigma \) to make a soft decision between generation and copy at each step \( t \), i.e. \( g_t = \sigma(v^y_j (d_t, z^m_t) \sigma) \) and
\[
p(x_{t\cdot}, y) = g_t p^\text{gen}(x_t|x_{t\cdot}, y) + (1 - g_t)p^\text{copy}(x_t|x_{t\cdot}, y)
\]
where \( g_t \in [0, 1] \) is the balance score, \( v^y \) is a weight vector. Distribution \( p^\text{copy}(\cdot) \) is defined over \( M \) slots in \( (o^m_1, \ldots, o^m_M) \):
\[
p^\text{copy}(x_{t\cdot}|x_{t\cdot}) = \begin{cases} 
\alpha^m_{m}, & \tilde{x}_t \text{ is the slot of } o^c_m, m \in [1, M] \\
0, & \text{otherwise}
\end{cases}
\]
Afterward, we can get the final sentence \( x \) by substituting each slot in \( \tilde{x} \) with the value in the corresponding slot-value pair. The loss function of SSG model given \( x \) and \( y \) is
\[
\mathcal{L}_\text{SSG}(y, x) = -\sum_{i=1}^{|
\tilde{x}|} \log p(x_{t\cdot}|x_{t\cdot}, y) + \mathcal{L}_\text{SC}
\]

V. DUAL SEMI-SUPERVISED NLU

In this section, we will describe our dual semi-supervised framework for the NLU task, which contains two methods: dual pseudo-labeling and dual learning. Algorithm 1 gives an overview of the dual semi-supervised NLU. Besides the labeled data \( D^L_y \), there are two kinds of unlabeled data here: pure (unlabeled) sentences set \( D^U_y \) and pure (unexpressed) semantic forms (i.e., intents and lists of slot-value pairs without corresponding sentences) set \( D^U_y \).

A. Dual Pseudo-Labeling Method

Pseudo-Label are target labels for unannotated data as if they were true labels [22]. We first pre-train the NLU and SSG models in a supervised fashion with labeled data. Given an unlabeled sentence \( x \), we can use the NLU model to generate pseudo label \( y' \). Symmetrically, we also can obtain \( x' \) by utilizing the SSG model given an unexpressed semantic form \( y \). In other words, we can create pseudo training samples \((x, y')\) and \((x', y)\) in addition to the existing labeled data, as shown in Algorithm 1 (part 1).

Except for the supervised training stage with the labeled data, the NLU and SSG models can also be fine-tuned with the pseudo-samples by respectively minimizing losses \( w_i(\mathcal{L}_\text{NLU}(x, y') + \mathcal{L}_\text{NLU}(x', y)) \) and \( w_i(\mathcal{L}_\text{SSG}(y, x') + \mathcal{L}_\text{SC}(y', x)) \) at the \( i \)-th iteration, where \( w_i \) is an important coefficient. To prevent models stuck in poor local minima, we slowly increase \( w_i \) such that greater confidence is assigned to pseudo-samples as training goes on. Concretely, \( w_i = \frac{i}{N} \), where \( N \) is the maximum number of iterations.

Besides unlabeled \( x \) and unexpressed \( y \), we also generate pseudo-samples starting from a sentence \( x \) and a semantic form \( y \) in the labeled data. Because it may rectify annotation noise and create various expressions for the same semantic form.

Previous work [22], [23], [55] related to the pseudo-labeling method only proposes to generate pseudo-labels \( y' \) for unlabeled inputs \( x \). To the best of our knowledge, we are the first to propose the dual pseudo-labeling method, which shares pseudo-samples between the NLU and SSG tasks and optimizes the NLU and SSG models iteratively.

Algorithm 1: Dual Semi-supervised NLU.

**Require:** Labeled training set \( D^L_y \); pure (unlabeled) sentences set \( D^U_y \); pure (unexpressed) semantic forms (i.e., intents and lists of slot-value pairs) set \( D^U_y \); beam search size \( K \); weight factor \( \delta \); maximum number of iterations \( N \).

1. Train sentence side language model \( LM(\cdot) \) on data \( D^L_y \cup D^U_y \). Build lexicon database \( DB(\cdot) \) and intent-slot co-occurrence matrix \( COM \) on \( D^L_y \cup D^U_y \).
2. Pre-train NLU(\( \cdot|\Theta_{\text{NLU}} \)) and SSG(\( \cdot|\Theta_{\text{SSG}} \)) models on \( D^U_y \) by respectively minimizing the cross-entropy losses \( \sum_{(x, y)} \mathcal{L}_\text{NLU}(x, y) \) and \( \sum_{(x, y)} \mathcal{L}_\text{SSG}(y, x) \).
3. for \( i = 1 \) to \( N \) do
4. repeat
5. Sample sentence \( x \sim D^L_y \cup D^U_y \)
6. Sample semantic form \( y \sim D^L_y \cup D^U_y \)

**Part 1: Dual Pseudo-Labeling Method**

7. Use the current NLU model to generate pseudo labels for \( x \), i.e., \( y' = \text{NLU}(x|\Theta_{\text{NLU}}) \).
8. Use the current SSG model to generate pseudo sentences for \( y \), i.e., \( x' = \text{SSG}(y|\Theta_{\text{SSG}}) \).
9. Update \( \Theta_{\text{NLU}} \) and \( \Theta_{\text{SGG}} \) on the pseudo-samples by minimizing \( w_i(\mathcal{L}_\text{NLU}(x, y') + \mathcal{L}_\text{NLU}(x', y)) \) and \( w_i(\mathcal{L}_\text{SSG}(y, x') + \mathcal{L}_\text{SC}(y', x)) \) respectively.

**Part 2: Dual Learning Method**

10. Produce \( K \) semantic forms \( y^{(1)}, \ldots, y^{(K)} \) using beam search according to NLU(\( x|\Theta_{\text{NLU}} \)).
11. For each \( y^{(k)} \), generate \( x^{(k)} = \text{SSG}(y^{(k)}|\Theta_{\text{SSG}}) \).
12. For \( k \)-th sample, compute the total reward \( r^x_k(x) \).
13. Compute gradients \( \nabla_{\Theta_{\text{NLU}}} \psi_1(x) \) and \( \nabla_{\Theta_{\text{SSG}}} \psi_1(x) \).
14. Produce \( K \) sentences \( x^{(1)}, \ldots, x^{(K)} \) using beam search according to SSG(\( y|\Theta_{\text{SSG}} \)).
15. For each \( y^{(k)} \), generate \( x^{(k)} = \text{NLU}(x^{(k)}|\Theta_{\text{NLU}}) \).
16. For \( k \)-th sample, compute the total reward \( r^y_k(y) \).
17. Compute gradients \( \nabla_{\Theta_{\text{NLU}}} \psi_2(y) \) and \( \nabla_{\Theta_{\text{SSG}}} \psi_2(y) \).
18. Update \( \Theta_{\text{NLU}} \) with \( \delta \nabla_{\Theta_{\text{NLU}}} \psi_1(x) + (1 - \delta) \nabla_{\Theta_{\text{NLU}}} \psi_2(y) \); update \( \Theta_{\text{SSG}} \) with \( \delta \nabla_{\Theta_{\text{SSG}}} \psi_1(x) + (1 - \delta) \nabla_{\Theta_{\text{SSG}}} \psi_2(y) \).

**Part 3: Raw Supervised Training**

19. Sample training pair \((x, y) \sim D^L_y \).
20. Update \( \Theta_{\text{NLU}} \) and \( \Theta_{\text{SSG}} \) by minimizing \( \mathcal{L}_\text{NLU}(x, y) \) and \( \mathcal{L}_\text{SSG}(y, x) \) respectively.
21. until all samples in \( D^L_y \cup D^U_y \) are sampled.
22. end for

B. Dual Learning Method

Besides the pseudo-labeling method where the NLU and SSG models are updated separately, we also propose to apply the dual learning method by jointly training the two models. We use one agent to represent the model of the primal task (NLU) and another agent to represent the model of the dual task (SSG). Then a two-agent game is designed in a closed loop which can provide quality feedback to the primal and dual models even if only sentences or semantic forms are available. As the feedback rewards are non-differentiable, reinforcement learning algorithm [56] based on policy gradient [57] is applied for optimization.
As illustrated in Fig. 4, two agents, NLU and SSG, participate in the collaborative game with two directed loops. 1) **NLU2SSG** loop starts from a sentence, generates a possible semantic form by agent NLU and tries to reconstruct the original sentence by SSG. 2) **SSG2NLU** loop starts from the opposite side. Each agent will obtain quality feedback depending on reward functions defined in the directed loops. The NLU and SSG models are pre-trained on the labeled data. Let Θ_NLU and Θ_SSG denote all the parameters of the NLU and SSG models respectively. A brief description of the dual learning algorithm is provided in Algorithm 1 (part 2), comprised of the two directed loops:

1) **Loop NLU2SSG:** We sample a sentence \( x \) from the union of labeled and unlabeled data randomly. Given \( x \), the NLU model could produce \( K \) possible semantic form \( y^1, \ldots, y^K \) via beam search (\( K \) is beam size). For each \( y^k \), we can obtain a validity reward \( R^k_{val}(y^k) \) (a scalar) which reflects the likelihood of \( y^k \) being a valid semantic form. Afterwards, we pass \( y^k \) into the SSG model and get an output \( x^k \) via greedy decoding. Finally, we get a reconstruction reward \( R^X_{rec}(x^k, y^k) \) which forces the generated sentence \( x^k \) as similar to \( x \) as possible. The rewards will be elucidated in Section V-B3. A coefficient \( \alpha \in (0, 1] \) is exploited to balance these two rewards in \( r^X_1(x) = \alpha R^X_{val}(y^k) + (1 - \alpha) R^X_{rec}(x^k, y^k) \).

By minimizing the expected negative reward \( \psi_1(x) = -\mathbb{E}\left[ -\frac{1}{K} \sum_{k=1}^{K} r^X_1(x) \right] \) via policy gradient [57], the stochastic gradients of \( \Theta_{NLU} \) and \( \Theta_{SSG} \) are computed as:

\[
\nabla_{\Theta_{NLU}} \psi_1(x) = \frac{1}{K} \sum_{k=1}^{K} r^X_1(x) \nabla_{\theta_{NLU}} L_{NLU}(x, y^k)
\]

\[
\nabla_{\Theta_{SSG}} \psi_1(x) = \frac{1}{K} \sum_{k=1}^{K} R^X_{rec}(x^k, y^k) \nabla_{\theta_{SSG}} L_{SSG}(y^k, x)
\]

2) **Loop SSG2NLU:** Symmetrically, we sample a semantic form \( y \) from the labeled and unlabeled data randomly. Given \( y \), the SSG model could generate \( K \) possible sentences \( x^1, \ldots, x^K \) via beam search. For each \( x^k \), we can obtain a validity reward \( R^k_{val}(x^k, y) \) which reflects whether the sampled natural language sentence \( x^k \) is well-formed and fluent. Afterwards, we feed \( x^k \) into the NLU model, and get the top-hypothesis \( y^k \). Finally, we get a reconstruction reward \( R^Y_{rec}(y, y^k) \) which forces \( y^k \) as similar to \( y \) as possible. The rewards will be explained in Section V-B3. A coefficient \( \beta \in [0, 1] \) is exploited to balance these two rewards in \( r^Y_2(y) = \beta R^Y_{val}(x^k, y) + (1 - \beta) R^Y_{rec}(y, y^k) \).

By minimizing the negative expected reward \( \psi_2(y) = \mathbb{E}[\frac{1}{K} \sum_{k=1}^{K} r^Y_2(y)] \) via policy gradient [57], the stochastic gradients of \( \Theta_{SSG} \) and \( \Theta_{NLU} \) are computed as:

\[
\nabla_{\Theta_{SSG}} \psi_2(y) = \frac{1}{K} \sum_{k=1}^{K} r^Y_2(y) \nabla_{\theta_{SSG}} L_{SSG}(y, x^k)
\]

\[
\nabla_{\Theta_{NLU}} \psi_2(y) = \frac{1 - \beta}{K} \sum_{k=1}^{K} R^Y_{rec}(y, y^k) \nabla_{\theta_{NLU}} L_{NLU}(x^k, y)
\]

To the best of our knowledge, we are the first to apply the dual learning algorithm to the intent detection and slot filling in NLU. Compared with the dual learning for neural machine translation [25] with only language models based reward, we introduce new validity and reconstruction rewards for structured data of NLU.

3) **Reward Design:** Here we will give some details about two validity and two reconstruction rewards introduced in the two directed loops above.

- **Validity reward of \( R^Y_{val}(y') \)** measures whether a possible semantic form \( y' \) is valid. It can be jointly evaluated on intents and slots from two relations:
  - **Slot-value:** whether a given slot-value pair is valid, e.g. "boston" is a valid city name for slot `FromCity`.
  - **Slot-intent:** whether a slot is likely to co-occur with respect to the predicted intent, e.g. `FromCity` often co-occurs with intent `find_flight`.

To this end, a lexicon database \( DB() \) is created from the training set, which specifies any possible value \( v \) for each slot \( s \). A co-occurrence matrix (COM) is leveraged from the training set, where \( COM(i, s) \in \{0, 1\} \) indicates whether the slot \( s \) co-occurs with the user intent \( i \). Concretely, \( R^Y_{val}(y') \) is defined as

\[
\text{score}(s, v) = \max_{e \in DB(s)} \left( 1 - \frac{\text{Edit-Distance}(e, v)}{|v|} \right)
\]

\[
r_{sv}(o^C) = \begin{cases} 
\frac{1}{|o^C|} \sum_{(s, v) \in o^C} \text{score}(s, v), & \text{if } |o^C| \neq 0 \\
1.0, & \text{otherwise}
\end{cases}
\]

\[
r_{sl}(o^I, o^C) = \begin{cases} 
\frac{1}{|o^C|} \sum_{(s, v) \in o^C} \text{COM}(o^I, s), & \text{if } |o^C| \neq 0 \\
1.0, & \text{otherwise}
\end{cases}
\]

\[
R^Y_{val}(y') = \lambda \cdot r_{sv}(o^C) + (1 - \lambda) \cdot r_{sl}(o^I, o^C)
\]

where \( y' \) contains an intent \( o^I \) and a list of slot-value pairs \( o^C \), \( \text{Edit-Distance}(e, v) \) calculates a word-level edit distance between two values, and \( \lambda \) is a weight factor.

- **Validity reward of \( R^X_{val}(x', y) \)** measures whether a generated natural language sentence \( x' \) is well-formed and fluent. We also evaluate it from two aspects:
  - **Semantic integrity:** whether \( x' \) expresses all slots in the input \( y \) precisely. This can be measured by a metric of slot accuracy, i.e. \( \text{SlotAcc}(x', y) = 1 - \frac{p - q}{m} \) where \( m \) is the total number of slot-value pairs in \( y \), \( p \) and \( q \) are the number of omitted and redundant slots in the delexicalized form of \( x' \) respectively.
  - **Word fluency:** the probability of \( x' \) to be a natural language sentence. We train a LSTM based language model [58] with sentences of both the labeled and...
unlabeled data to evaluate the quality of \( x' \). Length-normalization [59] is applied to make a fair competition between short and long sentences, i.e. \( \frac{1}{|x'|} \log \text{LM}(x') \).

A weight factor \( \gamma \) is used to combine these two aspects:

\[
R_{\text{val}}^X(x', y) = \gamma \cdot \text{SlotAcc}(x', y) + (1 - \gamma) \cdot \frac{1}{|x'|} \log \text{LM}(x')
\]

**Reconstruction reward of** \( R_{\text{rec}}^X(x, x') \) **measures** the similarity score between the finally generated sentence \( x' \) and the raw input \( x \). The \( \text{BLEU} \) score on the validation set. Finally, we report the intent accuracy and \( F_1 \) score of slot-value pairs on the test set with parameters that have achieved the best average of intent accuracy and slot \( F_1 \) score of slot-value pairs on the test set with parameters that can be updated during training. The hidden size \( n \) is 256. Hyperparameters \( \alpha, \beta, \gamma, \omega, \delta \) are set to 0.5, and \( \lambda \) is 0.25 empirically. For the dual learning, the beam size \( K \) is set to 5. The network parameters are randomly initialized under the uniform distribution \([-0.2, 0.2]\), except for the pre-trained word embeddings. We use optimizer Adam [63] with learning rate 0.001 for all experiments. The dropout with a probability of 0.5 is applied to the non-recurrent connections during the training stage. The batch size is 16 for all datasets. The maximum norm for gradient clipping is set to 5, and we use \( l_2 \) norm regularization on all weights with factor 1e-5 to avoid over-fitting. We keep the learning rate for 50 epochs and save the parameters that give the best performance on the validation set. Finally, we report the intent accuracy and \( F_1 \)-score of slot-value pairs on the test set with parameters that achieved the best average of intent accuracy and slot \( F_1 \)-score on the validation set. The \( F_1 \)-score is calculated using CoNLL evaluation script.

**C. Experimental Setup**

1) **Training Details:** The word embeddings with 400 dimensions are initialized by concatenating pre-trained Glove embeddings\(^3\) [61] and character embeddings [62], which can be used to simulate unlabeled data. We use optimizer Adam [63] with learning rate 0.001 for all experiments. The dropout with a probability of 0.5 is applied to the non-recurrent connections during the training stage. The batch size is 16 for all datasets. The maximum norm for gradient clipping is set to 5, and we use \( l_2 \) norm regularization on all weights with factor 1e-5 to avoid over-fitting. We keep the learning rate for 50 epochs and save the parameters that give the best performance on the validation set. Finally, we report the intent accuracy and \( F_1 \)-score of slot-value pairs on the test set with parameters that achieved the best average of intent accuracy and slot \( F_1 \)-score on the validation set. The \( F_1 \)-score is calculated using CoNLL evaluation script.\(^\text{2}\)

Besides using pre-trained word embeddings, some advanced pre-trained language models (e.g., ELMo [64], BERT [65]) can also be used to get input embeddings. It is investigated in the following ablation studies. We employ the pre-trained BERT model (bert-base-cased) with 12 layers of 768 hidden units and 12 self-attention heads\(^5\). We update all the parameters using the Adam with a learning rate 5e-5.

2) **Data settings for semi-supervised learning:** To evaluate the effectiveness and efficiency of different methods for semi-supervised NLU, we discuss the experimental configuration for semi-supervised settings below. In order to simulate the annotation scarcity problem in the real world, a part of the training set is kept as fully labeled data (\( D_{\text{x}}^L \)), and the rest is left as unpaired sentences and semantic forms (\( D_{\text{x}}^U \) and \( D_{\text{y}}^U \) respectively) to simulate unlabeled data. For the part of labeled data, we randomly select 5, 10, 15, 20, 30 and 50 percent of the training set in each dataset for experiments.

3) **Significance Test:** We use McNemar’s test to establish the statistical significance of a method over another (\( p < 0.05 \)).

**D. Overall Results**

We first compare different methods on ATIS and SNIPS datasets with the simulated semi-supervised settings as well as

---

\(^1\)https://nlp.stanford.edu/data/glove.840B.300d.zip
\(^2\)https://www.clips.uantwerpen.be/conll2000/chunking/output.html
\(^3\)http://nlp.stanford.edu/data/glove.840B.300d.zip
\(^4\)https://github.com/google-research/bert#pre-trained-models
Table II shows slot F1 scores and intent accuracies of different methods on ATIS and SNIPS datasets. We randomly select {5,10,20,30,50}% of the training set as labeled data, and leave the rest to simulate unlabeled data. The results in bold black are the best slot F1 scores and intent accuracies. ‡ indicates our results that significantly outperform the best baseline.

Table III

Comparison among BLSTM-softmax, BLSTM-CRF and BLSTM-focus for supervised NLU on ATIS and SNIPS datasets.

Table IV

Data analysis of test sets compared with training sets.

E. Analysis

In Section VI-D, significant improvements of two metrics have been witnessed on the two datasets. However, we would like to figure out the potential factors for the improvement. In this sub-section, we will show ablation studies on the SSG model, the dual pseudo-labeling and dual learning methods to reveal the effects of different components. Finally, we analyze the effect of BERT in our framework.
1) Ablation studies of the SSG model: To verify the effectiveness of the SSG model for the dual task of NLU, we apply ablation studies of supervised SSG on ATIS and SNIPS datasets, as shown in Table V. BLEU score [60] is exploited to measure the similarity between generated sentences and references in the test set. We also utilize the slot accuracy mentioned in Section V-B3 to measure the semantic integrity of generated sentences. From the result of “(-) w/o feeding intent” row, we can observe that the intent is essential for BLEU scores, whereas intent detection is much easy in NLU. The global-level BLSTM in the encoder, copy mechanism and SC-LSTM cell in the decoder are also important components of the SSG model.

Besides the supervised training, we may further want to know whether the SSG model will be improved in our proposed dual semi-supervised NLU. As shown in Table VI, the dual pseudo-labeling and dual learning methods can also improve the performance of the SSG model as well as the NLU model, where 10% and 5% of the training sets are selected as labeled data in ATIS and SNIPS respectively.

2) Ablation studies of the dual pseudo-labeling method: The dual pseudo-labeling method creates pseudo-samples in two ways: a) obtaining predicted semantic forms of sentences with the NLU model and b) generating sentences with the SSG model given intents and slot-value pairs. Experiments are conducted to make a comparison of these two ways, as shown in Table VII. From the results, we can find that pseudo-samples created from SSG are more essential. The reason may be that the SSG model tends to generate sentences semantically consistent with the given semantic forms though the sentences are not natural enough. However, the NLU model may predict wrong semantic labels. Meanwhile, without pseudo-samples from SSG model, the dual pseudo-labeling method reduces to the traditional pseudo-labeling without dual task.

From the result of “(+) \( w_i = 1 \)” row in Table VII, we can see that the increasing coefficient \( w_i \) at each iteration helps. We believe that the NLU and SSG models updated with more iterations could provide more qualified pseudo-samples.

Therefore, the performance decreases if we keep using pseudo-samples generated at the first iteration, as shown in “(-) w/o iterative generation” row.

3) Ablation studies of the dual learning method: Several experiments are conducted to show the effects of different components in the dual learning method, as illustrated in Table VIII. From the results of “(-) w/o unlabeled sentences” and “(-) w/o unexpressed semantic forms” rows, we can find that unexpressed semantic forms are more important, which is consistent with the findings in the ablation studies of the dual pseudo-labeling method. It may facilitate semi-supervised NLU, since semantic forms are well-structured and could be easily synthesized under domain knowledge. The last two rows show that two kinds of rewards are essential, while the validity rewards impact more on the slot F₁ score.

4) Effect of BERT: Besides using pre-trained word embeddings, the BERT model can also be used to get input embeddings. However, it is orthogonal to the investigation of semi-supervised NLU. Table IX shows results on ATIS and SNIPS, where 10% and 5% of the training sets are selected as labeled data in respective. The results show that a pre-trained BERT model can further enhance our dual semi-supervised NLU as well as the baseline. Although BERT embeddings can bridge the gap between our method and the baseline, the dual semi-supervised NLU still outperforms the baseline significantly.

F: Compared with the Previous Results of the Supervised NLU

Finally, we make a comparison with the previous results on ATIS and SNIPS datasets using full training sets, as illustrated in Table X. Our proposed method (+ dual PL + dual learning) can achieve the state-of-the-art performances on the two datasets, but not significantly outperforming the previous state-of-the-art. We can also find that BERT boosts the performance of ATIS less than SNIPS, which may occur due to the much smaller vocabulary of ATIS. Our method

---

### Table V

| Model                | ATIS | SNIPS |
|----------------------|------|-------|
|                     | BLEU | Slot Acc | BLEU | Slot Acc |
| Supervised SSG       | 47.17 | 97.72 | 39.18 | 100.00 |
| (-) w/o feeding intent | 44.86 | 98.26 | 38.15 | 99.70 |
| (-) w/o global BLSTM | 41.14 | 96.15 | 31.65 | 98.95 |
| (-) w/o copy mechanism | 44.08 | 96.58 | 38.67 | 99.98 |
| (-) w/o SC-LSTM      | 46.78 | 97.25 | 37.99 | 100.00 |

### Table VI

| Method                | ATIS (10%) | SNIPS (5%) |
|----------------------|------------|------------|
|                     | BLEU Slot Acc | BLEU Slot Acc |
| Supervised SSG       | 39.53 | 87.49 | 29.94 | 94.85 |
| + dual PL            | 40.28 | 93.90 | 35.26 | 98.85 |
| + dual learning      | 38.84 | 91.30 | 32.19 | 98.10 |
| + dual PL + dual learning | 41.85 | 95.10 | 36.61 | 99.43 |

### Table VII

| Method                      | ATIS 5% | SNIPS 5% |
|-----------------------------|---------|----------|
|                             | Slot F₁ | Intent Acc |
|                             | 93.86 | 98.57 |
| + dual PL                   | 93.51 | 98.00 |
| (-) w/o pseudo-samples from NLU model | 90.67 | 97.57 |
| (+) \( w_i = 1 \)           | 93.65 | 98.14 |
| (-) w/o iterative generation | 91.49 | 97.71 |

### Table VIII

| Method                                   | ATIS 5% | SNIPS 5% |
|------------------------------------------|---------|----------|
|                                          | Slot F₁ | Intent Acc |
|                                          | 92.85 | 98.29 |
| + dual learning                          | 92.96 | 98.14 |
| (-) w/o unlabeled sentences              | 91.41 | 97.71 |
| (-) w/o unexpressed semantic forms       | 91.55 | 97.71 |
| (-) w/o validity rewards                 | 93.74 | 97.86 |
| (-) w/o reconstruction rewards           |         |          |

---

\(^\text{5}\)We only consider BERT embeddings of the first subword if a word is broken into multiple subwords.
enhanced with BERT gets a decrease of slot F1 score (from 96.4% to 96.0%) and an increase of intent accuracy (from 98.5% to 99.1%) on ATIS, while it achieves a better average score. It shows that our proposed method can also work well in fully supervised settings.

VII. CONCLUSION

This paper has introduced a dual task for SLU, which is semantic-to-sentence generation (SSG). It is incorporated in our proposed dual semi-supervised NLU to utilize unexpressed semantic forms as well as unlabeled sentences. The dual semi-supervised NLU includes the dual pseudo-labeling and dual learning methods which can learn the NLU and SSG models iteratively in the closed-loop of the primal and dual tasks. The proposed approaches are evaluated on two public datasets (ATIS and SNIPS). From the experimental results, we find that the dual semi-supervised NLU involving SSG could significantly improve the performances over traditional semi-supervised methods. We also provide extensive ablation studies to verify the effectiveness of our methods. Meanwhile, our methods can also achieve the state-of-the-art performance on the two datasets in the supervised setting. The proposed framework of dual semi-supervised NLU shows promising perspectives of future improvements.

- Exploiting semantic forms for semi-supervised learning could be more affordable and effective than collecting in-domain sentences, since semantic forms are well-structured and could be automatically synthesized under domain knowledge.
- Validity and reconstruction rewards are important for softly validating pseudo-samples. We will explore appropriate rewards to improve the effectiveness and efficiency of the dual semi-supervised NLU in our future work.
- This work has shown the effectiveness of incorporating dual task in semi-supervised NLU. For developing a conversational dialogue system with wide application domains, the domain adaptation and transfer problems of the dual task will be an interesting future research direction.

REFERENCES

[1] Y.-Y. Wang, L. Deng, and A. Acero, “Spoken language understanding—an introduction to the statistical framework,” IEEE Signal Processing Magazine, vol. 22, no. 5, pp. 16–31, 2005.
[2] P. Xu and R. Sarikaya, “Convolutional neural network based triangular crf for joint intent detection and slot filling,” in Proc. IEEE ASRU, 2013, pp. 78–83.
[3] B. Liu and I. Lane, “Attention-based recurrent neural network models for joint intent detection and slot filling,” in Proc. INTERSPEECH, 2016, pp. 685–689.
[4] X. Zhang and H. Wang, “A joint model of intent determination and slot filling for spoken language understanding,” in Proc. IJCAI, 2016, pp. 2993–2999.
[5] C.-W. Guo, G. Gao, Y.-K. Huo, T.-C. Chen, K.-W. Hsu, and Y.-N. Chen, “Slot-gated modeling for joint slot filling and intent prediction,” in Proc. NAACL, 2018, pp. 753–757.
[6] Q. Chen, Z. Zhuo, and W. Wang, “BERT for joint intent classification and slot filling,” arXiv preprint arXiv:1902.10909, 2019.
[7] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, “A stack-propagation framework with token-level intent detection for spoken language understanding,” in Proc. EMNLP-IJCNLP, 2019, pp. 2078–2087.
[8] R. De Mori, F. Bechet, D. Hakkani-Tur, M. McTear, G. Riccardi, and G. Tur, “Spoken language understanding,” IEEE Signal Processing Magazine, vol. 25, no. 3, pp. 50–58, 2008.
[9] G. Tur and L. Deng, “Intent determination and spoken utterance classification,” Spoken language understanding: systems for extracting semantic information from speech, Wiley, Chichester, pp. 93–118, 2011.
[10] R. Sarikaya, G. E. Hinton, and A. Deoras, “Application of deep belief networks for natural language understanding,” IEEE/ACM Trans. on Audio, Speech, Lang. Process., vol. 22, no. 4, pp. 778–784, 2014.
[11] Y. Wang, L. Deng, and A. Acero, “Semantic frame-based spoken language understanding,” Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, pp. 41–91, 2011.
[12] Y. He and S. Young, “A data-driven spoken language understanding system,” in Proc. IEEE ASRU, 2003, pp. 583–588.
[13] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in Proc. INTERSPEECH, 2007, pp. 1605–1608.
[14] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu et al., “Using recurrent neural networks for slot filling in spoken language understanding,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 23, no. 3, pp. 530–539, 2015.
[15] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory neural networks,” in IEEE SLT Workshop, 2014, pp. 189–194.
[16] N. T. Vu, “Sequential convolutional neural networks for slot filling in spoken language understanding,” in Proc. INTERSPEECH, 2016, pp. 3250–3254.
[17] G. Kurata, B. Xiang, B. Zhou, and M. Yu, “Leveraging sentence-level information with encoder lstm for semantic slot filling,” in Proc. EMNLP, 2016, pp. 2077–2083.
[18] S. Zhu and K. Yu, “Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding,” in Proc. ICASSP, 2017, pp. 5675–5679.
[19] C. Li, L. Li, and J. Qi, “A self-attentive model with gate mechanism for spoken language understanding,” in Proc. EMNLP, 2018, pp. 3824–3833.
[20] L. S. Zettlemoyer and M. Collins, “Online learning of relaxed ccg grammars for parsing to logical form,” in Proc. EMNLP-CoNLL, 2007, pp. 678–687.
[21] G. Tur, D. Hakkani-Tur, and R. E. Schapire, “Combining active and semi-supervised learning for spoken language understanding,” Speech Communication, vol. 45, no. 2, pp. 171–186, 2005.
