IterNLU: An Iterative Learning Framework for Joint Intent Detection and Slot Filling

Anonymous ACL submission

Abstract

Exploiting the cross-impact between intent detection and slot filling, two main tasks for natural language understanding (NLU), has been a recent trend for NLU research. While the cross-impact has often been modeled implicitly through joint learning, various methods have also been proposed to explicitly use slot information to facilitate intent detection and/or extract intent information to facilitate slot filling. However, previous works haven't fully explored the potential of the cross-impact yet. To better capture the benefit of intent-slot correlation, this paper proposes a novel joint learning framework, named IterNLU, to iteratively learn intent detection and slot filling with updated slot and intent information fed in, respectively, per iteration. In each iteration, we attempt to extract the most effective intent/slot representation based on available information to assist subsequent detection of slot/intent. Our proposed approach achieves 0.95% and 1.74% absolute gains on semantic frame accuracy over the best previous state-of-the-art NLU approach on two public benchmark datasets, ATIS and Snips, respectively. We also conduct systematic analyses on the effect of feeding in different types of intent/slot information as well as on system efficiency.

1 Introduction

Natural language understanding is a technology critical for many language/speech related applications, such as dialog systems (Chen et al., 2017; Tur and De Mori, 2011). Intent detection and slot filling are two main tasks of NLU, targeting to detect the user intent (e.g., play music) and the relevant slots (e.g., song/artist names) in the input sentence (e.g., "play Faded by Alan Walker"), respectively. The two tasks can be modeled individually, with either traditional algorithms (e.g., maximum entropy model, support vector machines, and conditional random fields (CRF)) or various neural networks (Haffner et al., 2003; Hashemi et al., 2016; Raymond and Riccardi, 2007; McCallum et al., 2000; Mesnil et al., 2015). Since intent detection and slot filling are correlated by nature, jointly learning the two tasks has become increasingly popular for NLU especially after 2015 (Liu and Lane, 2016; Zhang et al., 2019; Chen et al., 2019).

As joint learning has been proven effective to enhance NLU, which indicates the benefit of capturing the cross-impact between intent detection and slot filling, an emerging recent trend for NLU research is to explicitly make use of the slot-intent correlation to facilitate the two tasks. Most of such efforts attempt to extract intent information to facilitate slot filling in certain way (Goo et al., 2018; Qin et al., 2019, 2020). Wang et al. (2018) and E et al. (2019) attempt to use the correlation to aid both tasks. While Wang et al. (2018) adopt separate bidirectional LSTM (BiLSTM) for each task and feeds the intent/slot hidden representation into the counterpart decoder, E et al. (2019) adopt a shared BiLSTM for both task, computing attention-based slot and intent context vectors (that are augmented with each other iteratively) to facilitate the intent and slot prediction, respectively.

In this paper, we aim to fully capture the benefit of slot-intent correlation in NLU, and propose a joint learning NLU approach that not only explicitly extracts intent and slot information to improve slot filling and intent detection, respectively, but also conducts iterative intent detection and slot filling. We refer to this approach as the IterNLU framework. Note that none of the previous NLU approaches conducts multiple passes of intent detection and slot filling iteratively. We believe that such iteration is beneficial, as it imitates the human behavior that human may read through a difficult/challenging sentence several times to fully understand it. We construct the IterNLU framework as a set of intent detection and slot filling modules stacked upon the BERT embedding representations.

In each intent/slot module, we use an individual BiLSTM RNN to encode the input received from the preceding module, and attempt to extract the best possible intent/slot information to feed into the subsequent module with the aim of facilitating the counterpart learning. We evaluate the proposed approach on two popular NLU benchmark datasets, ATIS and Snips. The experimental results obtained are encouraging, our approach leads to 7.6% and 16.4% relative reductions on overall error rate over the best state-of-the-art baseline performance for ATIS and Snips, respectively.

We further investigate the effect of extracting various types of intent and slot information on the performance of the IterNLU framework in this work. The efficiency of the IterNLU framework is also investigated with various number of intent-slot iterations adopted.

2 Related Works

There are many previous works in literature that jointly learn intent detection and slot filling using various approaches, ranging from Triangular CRF (Xu and Sarikaya, 2013), Capsule Neural Network (Zhang et al., 2019), to RNN based models (Hakkani-Tür et al., 2016; Guo et al., 2014; Zhang and Wang, 2016; Liu and Lane, 2016). Recently, pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Vaswani et al., 2017), which are proven effective across many NLP tasks, have been deployed in NLU too (Chen et al., 2019; Qin et al., 2019). Chen et al. (2019) successfully used the BERT model to significantly boost the NLU performance, showing the effectiveness of contextualized embedding. In this work, we follow these previous efforts to jointly learn intent detection and slot filling, using a joint cross-entropy loss function, and also adopt the BERT model to provide contextualized embeddings to the intent/slot layers. Note that BiLSTM has been found effective to model NLU in many of these previous works (Liu and Lane, 2016; Wang et al., 2018; E et al., 2019). This work thus adopts BiLSTM to implement the intent detection and slot filling modules.

For the previous NLU methods that explicitly use intent/slot information to facilitate the counterpart learning, intent information has been extracted as token-wise hidden representation, context vector obtained by attention computation, or intent prediction representation to aid the detection of slots through certain mechanism (e.g., gate) (Wang et al., 2018; Goo et al., 2018; Li et al., 2018; E et al., 2019; Qin et al., 2019). We compare these options for intent information extraction in our proposed IterNLU framework, while extending the attention computation by including sentence-level hidden representation to better capture the sentence intent. Regarding the slot information extraction, we follow (Wang et al., 2018) to investigate the token-wise slot hidden representation, and also explore the use of slot prediction representation, inspired by the intent information extraction case. Note that there is a recent work Slot Refine (Wu et al., 2020) that feeds in the slot prediction to a second pass of decoding to solve the uncoordinated slots problem, but no intent information is explicitly extracted to aid the second pass in that work.

The major difference between our proposed IterNLU approach and the previous works on joint intent detection and slot filling is that IterNLU introduces iterative intent detection and slot filling. In (E et al., 2019) and (Qin et al., 2020), a certain mechanism (i.e., mutual reinforcement, and multi-layer graph attention network) is proposed to iteratively augment the intent and slot representation extracted from the decoder, either to facilitate the slot filling alone or to improve both tasks. Since such iteration is only used as a post-processing after the RNN pass, mainly iterating between the slot/intent representations, only limited cross-impact information can be captured. In contrast, our IterNLU framework conducts much thorough iteration, iterating the whole intent detection and slot filling procedures in order to fully capture the intent-slot correlation. The IterNLU approach is similar to end-to-end memory networks in the sense that they both stack repetitive functional layers (e.g., memory layers) that each receives the output from the previous layer as input (Sukhbaatar et al., 2015; Kenter and de Rijke, 2017).

3 Approach

In this section, we propose the IterNLU framework, first introducing the overall system structure and then presenting the intent detection and slot filling modules, respectively.

3.1 Overall Framework Structure

The overall structure of the iterNLU framework is illustrated in Figure 1. The framework involves N repetitions of intent detection module and slot filling module, stacked upon an embedding module,

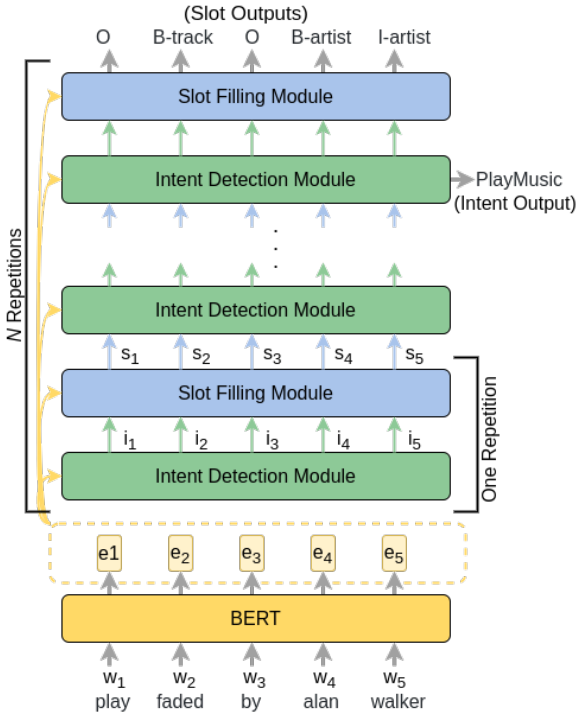

Figure 1: Overall structure of IterNLU framework.

to enable iterative intent detection and slot filling.

Given an input sentence $w_1 w_2 w_3 ... w_n$, we first encode each word w_k as an embedding vector e_k. The word embedding module can either be implemented with certain pre-trained model (Pennington et al., 2014; Joulin et al., 2017; Devlin et al., 2019; Radford et al.) or be developed from scratch. In this work, we adopt BERT (Devlin et al., 2019) to generate contextualized word embedding vectors.

In this framework, each intent module receives the word embeddings as well as the slot information representations (i.e., $s_1, s_2, s_3, ...s_n$) extracted from the preceding slot module (if available) as the module input, and generates a sequence of representations (i.e., $i_1, i_2, i_3, ...i_n$) that encode the detected intent(s). The subsequent slot module then conducts slot filling using those intent representations together with the word embeddings as the input. The last intent/slot module generates the final intent/slot prediction. Note that the order of the intent and slot modules in each repetition can be switched with minor system modifications.

We train the whole IterNLU framework using a joint cross-entropy loss function as follows,

$$\mathcal{L}_{joint} = -\sum_{j=1}^{n_I} y_j^I \, log\left(\hat{y}_j^I\right) - \sum_{i=1}^{L}\sum_{j=1}^{n_S} y_{i,j}^S \, log\left(\hat{y}_{i,j}^S\right)$$

where n_I and n_S refer to the numbers of intent and slot types respectively, $\hat{y}_j^I$ and $\hat{y}_{i,j}^S$ refer to the intent and slot predictions, y_j^I and $y_{i,j}^S$ refer to the corresponding ground-truth for intent and slot, and L refers to the number of tokens in the sentence.

In this work, we tie all the parameters for the intent modules as well as for the slot modules. While the weight sharing may improve the system robustness by providing additional regularization, the main advantage of such design is that in this way, once the model is trained, the number of intent-slot repetitions (i.e., N) for inference can be different from (e.g., smaller than) that used in training. This may bring significant improvement on inference efficiency, as will be discussed in Section 5.3.

3.2 Intent Detection Module

Each intent detection module attempts to provide the best possible intent information of the given sentence to the subsequent slot filling module, while the last intent module also generates the intent detection result. In this work, we implement each intent module based on BiLSTM together with an attention mechanism, as illustrated in Figure 2.

For each token w_k, we concatenate the word embedding e_k with the slot information representation s_k received from the previous slot filling module (for the first intent module, s_k is invalidated as 0), and feed it into the intent module BiLSTM. Upon receiving such input, the BiLSTM processes the given sentence in both forward and backward directions. We first extract token-wise intent hidden representations $[h_1^f, h_1^b], [h_2^f, h_2^b], ...[h_n^f, h_n^b]$ by concatenating the hidden representations generated by the BiLSTM in each direction. We then concatenate the last states of both directions, each of which covers the information of the whole sentence, to generate a sentence-level intent representation, denoted as $h_{sent} = [h_n^f, h_1^b]$. This representation is effective for intent detection as it well captures the sentence context.

It is possible to directly feed each token-wise intent representation into the subsequent slot module as i_k, or broadcast the sentence-level representation h_{sent} for all tokens to the next module, as will be discussed in Section 5.1. In this work, we attempt to further enhance the sentence-level intent representation to better capture the intent information. We propose an attention mechanism that leverages h_{sent} with the token-wise intent representations through an attention network. The attention network can be potentially implemented in various

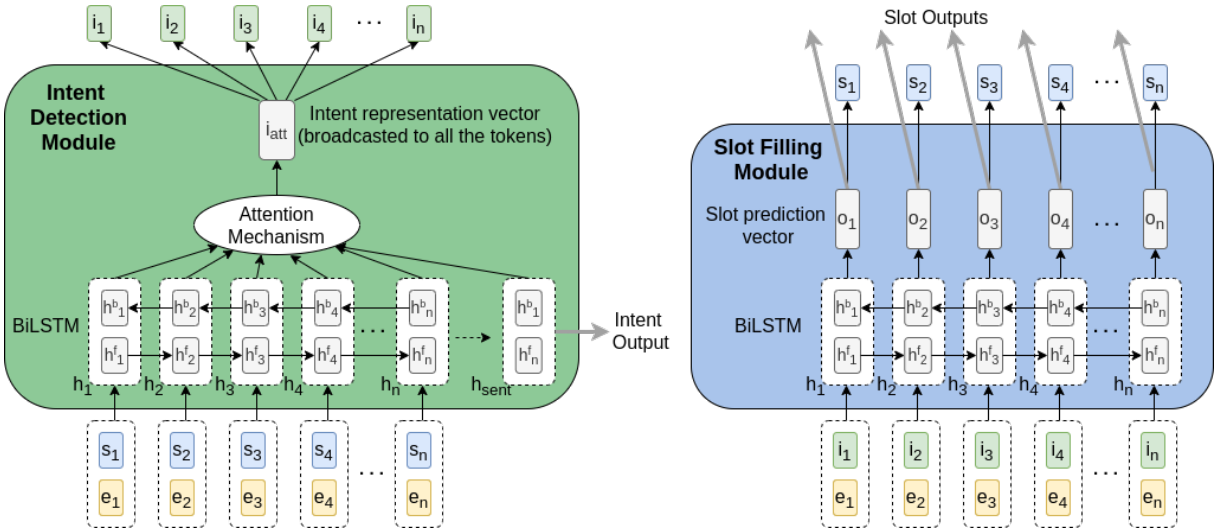

Figure 2: Illustration of the intent detection and slot filling modules.

ways (Vaswani et al., 2017; Liu and Lane, 2016). In this work, we follow (Liu and Lane, 2016) to use a feed-forward network to model attention. Different from (Liu and Lane, 2016), our attention method includes not only token-wise hidden representations but also the sentence-level hidden representation h_{sent} into the calculation of the weighted sum in a self-attention manner (see Figure 2). Our method generates an enhanced intent representation, denoted as i_{att}, as follows,

$$i_{att} = \sum_{h \in \mathcal{H}} \alpha(h) \cdot h$$

where $\mathcal{H}$ is the collection of h_{sent} and the token-wise representations. $\alpha(h)$ is the weight for h, calculated as follows:

$$\text{score}(h) = v_a^T \tanh(W_a h + U_a h_{sent}) \quad \in \mathbb{R}$$

$$\alpha(h) = \frac{\exp(\text{score}(h))}{\sum_{h' \in \mathcal{H}} \exp(\text{score}(h'))}$$

where $W_a, U_a \in \mathbb{R}^{m \times 2d}$, $v_a \in \mathbb{R}^m$ denote the weight matrices, d refers to the hidden state dimension of BiLSTM, and m refers to the attention hidden dimension.

The obtained intent representation i_{att} is then broadcast into the subsequent slot filling module for each token as all the i_k. In this study, regardless of the intent information extracted, the intent detection module always predicts the intent results based on h_{sent} (by applying a linear prediction layer on top of the h_{sent} representation), in order to facilitate the analysis.

3.3 Slot Filling Module

Each slot filling module is implemented with BiLSTM too in this work. The concatenation of the word embedding and the intent representation received from the preceding intent module per token, i.e., $[e_k, i_k](k = 1...n)$, serves as the input to the slot BiLSTM. We concatenate the hidden states generated by both directions of the BiLSTM for each token, i.e., $[h_k^f, h_k^b]$, to form the token-wise slot hidden representation, denoted as h_k^{slot}.

Similar to the extraction of intent information, various slot information can be extracted to facilitate the subsequent intent detection. The extracted slot representation s_k (as in Figures 1 and 2) could be h_k^{slot} or certain slot prediction vector. In this work, we adopt a one-hot representation of slot prediction result as s_k. We apply a linear projection layer on top of h_k^{slot} to get slot prediction vector o_k, which shows the predicted probability distribution over all the possible slot labels, for the k^{th} token. The prediction vector is then converted into a one-hot representation (by setting the dimension of the predicted label as 1 and others as 0) as s_k.

4 Experiments

4.1 Data

We conduct the experiments on two public benchmark datasets, ATIS (Hemphill et al., 1990) and Snips (Coucke et al., 2018). The ATIS (Air Travel Information Systems) corpus contains 18 distinct intents and 127 slot labels, covering air travel related topics. From the original 4978 training sentences of ATIS, we randomly reserve 500 sentences

as the development set, and use the remaining as the training set. We directly use the the original 893 testing sentences are the testing set. The Snips corpus was collected from daily smart-home conversations, involving 7 intent labels and 72 slot labels in total. There are 13084, 700 and 700 sentences in the training, development and testing sets, respectively, for Snips. In both the ATIS and Snips datasets, slots are labelled following the IOB schema (Ramshaw and Marcus, 1995).

4.2 Experimental Settings

We develop the IterNLU framework using Pytorch (Paszke et al., 2019). For the embedding layer, we adopt the BERT variant *BERT-base-uncased* (Wolf et al., 2019), and following (Chen et al., 2019), use the BERT representation of the first subword to represent a word token. For the intent detection and slot filling modules, we set the hidden state dimension of BiLSTM as 256 for each module, while in each intent module, the output dimension of the attention mechanism is set as 128. Unless stated otherwise, a same number of intent-slot repetitions (e.g., N = 5) is adopted for both training and inference. We train the whole system using Adam (with weight decay) optimizer (Loshchilov and Hutter, 2018) together with the early stopping criteria (patience= 50). A learning rate of $1e$-5 and batch size of 128 are adopted in training. The model is regularized with weight decay parameter set as 0.01 and dropout probability set as 0.2. To avoid exploding gradient problem, gradient clipping is used (maximum-gradient-norm set to 5).

We also select a set of previous NLU approaches that achieve state-of-the-art performance as the baseline algorithms for comparison, as listed in Table 1. We evaluate each of the baseline methods on ATIS and Snips separately. For those algorithms with open-source code available (i.e., Capsule-NLU, SF-ID, AGIF, and SlotRefine), we directly use the provided code to run the experiments. We re-implement the other approaches (i.e., Attention Enc.-Dec., Bi-Model, Stack Propagation + BERT, and Joint BERT) based on the corresponding papers. For each baseline algorithm, we develop it using the recommended settings in the corresponding paper if any, and manually tune the remaining hyperparameters for ATIS and Snips, respectively.

Throughout this work, the hyperparameters are tuned on corresponding development data, and the reported results are evaluated on testing data.

4.3 Results

We evaluate the proposed IterNLU approach (with various repetition number N adopted) as well as the baseline NLU algorithms using the same training/development/testing sets for ATIS and Snips. To reduce the impact of randomness and provide a fair comparison, we independently perform the training and evaluation for each of the listed NLU models five times, and report the mean performance together with the standard deviation for each evaluation metric. The comparison results are shown in Table 1. The evaluation metrics used include 1) the F1 score for slot filling, 2) the accuracy for intent detection, and 3) the sentence-level semantic frame accuracy, referred to as overall accuracy, which calculates the portion of the sentences whose slots and intents are both correctly detected.

From Table 1, we can see that the "Joint BERT" (Chen et al., 2019) is a NLU approach challenging to beat. It outperforms all the other baseline algorithms, including those newly proposed ones, in terms of overall accuracy in our multi-run statistics based fair comparison. It is encouraging that our proposed IterNLU approach achieves significant further improvements [1]. Over the best baseline "Joint BERT (+ CRF)", our best IterNLU models (by setting N as 5 for ATIS and 2 for Snips) bring 0.95% and 1.74% absolute increases on overall accuracy for ATIS and Snips, respectively. Note that the previously reported improvement on overall accuracy over "Joint BERT" baseline is only <0.5% absolute for ATIS and <0.2% absolute for Snips (Wu et al., 2020).

The benefit of IterNLU may come from 1) the module design, 2) information sharing between the intent detection and slot filling modules, and 3) iterative learning. To analyze the individual contributions of these factors, we also implement another system (denoted as "Parallel Implementation" in Table 1) that cuts off the intent-slot information sharing in IterNLU(N=1) by applying the same intent detection module and slot filling module in parallel upon the word embedding module, both only using BERT embedding as input.

On ATIS, as shown in Table 1, the Parallel Implementation achieves similar performance as the baseline "Joint BERT (+CRF)" approach. By feeding intent information into the slot module, IterNLU(N=1) not only achieves higher slot filling

[1] $p<0.05$ in t-test,compared with each baseline on overall accuracy. The box plots in Appendix also support the claim.

	ATIS Dataset			Snips Dataset		
Model	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**
Attention Enc.-Dec. (Liu and Lane, 2016)	94.64 (±0.43)	97.42 (±0.10)	84.76 (±0.79)	91.29 (±0.83)	97.80 (±0.19)	81.29 (±1.52)
Bi-Model (Wang et al., 2018)	95.24 (±0.20)	96.68 (±0.27)	85.34 (±0.26)	93.50 (±0.23)	97.09 (±0.19)	84.37 (±0.71)
Capsule-NLU (Zhang et al., 2019)	94.56 (±0.44)	80.11 (±12.43)	71.54 (±9.4)	92.13 (±0.49)	97.86 (±0.23)	83.29 (±0.83)
SF-ID (ID first + CRF) (E et al., 2019)	95.42 (±0.11)	96.24 (±0.73)	84.82 (±0.84)	90.61 (±0.29)	96.83 (±0.54)	77.66 (±0.54)
SF-ID (SF first + CRF) (E et al., 2019)	95.33 (±0.15)	96.86 (±0.28)	84.95 (±0.30)	90.89 (±0.61)	96.91 (±0.50)	78.40 (±1.06)
Joint BERT (without CRF) (Chen et al., 2019)	95.40 (±0.25)	98.00 (±0.15)	87.38 (±0.54)	95.60 (±0.45)	98.26 (±0.29)	89.29 (±1.02)
Joint BERT (+ CRF) (Chen et al., 2019)	95.72 (±0.13)	97.91 (±0.25)	87.56 (±0.41)	95.56 (±0.61)	98.34 (±0.21)	89.37 (±1.14)
Stack Propagation + BERT (Qin et al., 2019)	94.56 (±0.46)	97.80 (±0.17)	85.07 (±1.08)	95.20 (±0.36)	98.26 (±0.17)	88.83 (±0.93)
AGIF (Qin et al., 2020)	95.72 (±0.09)	96.66 (±0.24)	86.65 (±0.32)	94.19 (±0.27)	97.28 (±0.27)	86.49 (±0.69)
SlotRefine (Wu et al., 2020)	**96.20** (±0.26)	97.34 (±0.24)	85.34 (±0.61)	92.75 (±0.70)	97.54 (±0.44)	82.94 (±1.10)
Parallel Implementation	95.71 (±0.15)	97.91 (±0.26)	87.79 (±0.43)	95.94 (±0.58)	98.34 (±0.22)	90.57 (±1.07)
IterNLU (N=1)	95.83 (±0.06)	97.96 (±0.22)	88.17 (±0.15)	95.94 (±0.50)	98.51 (±0.21)	90.37 (±1.12)
IterNLU (N=2)	95.88 (±0.05)	98.00 (±0.08)	88.17 (±0.21)	**96.26** (±0.27)	98.60 (±0.21)	**91.11** (±0.49)
IterNLU (N=3)	95.84 (±0.14)	97.87 (±0.38)	88.26 (±0.48)	96.07 (±0.16)	98.34 (±0.37)	90.57 (±0.49)
IterNLU (N=4)	95.73 (±0.20)	**98.09** (±0.14)	88.17 (±0.35)	95.93 (±0.60)	98.46 (±0.32)	90.26 (±1.28)
IterNLU (N=5)	95.97 (±0.16)	97.96 (±0.08)	**88.51** (±0.45)	96.23 (±0.12)	**98.74** (±0.14)	90.60 (±0.19)

Table 1: Performance comparison on ATIS and Snips datasets. Each result reported is the average performance across 5 independent runs together with the standard deviation listed in parentheses.

	ATIS Dataset			Snips Dataset		
Intent Information	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**
token-wise hidden rep.	95.73	97.87	87.70	96.01	98.49	90.69
sentence hidden rep.	95.82	97.89	87.68	96.12	98.46	**90.86**
intent prediction (prob.)	**95.99**	97.89	88.26	95.81	98.46	90.31
intent prediction (one-hot)	95.76	97.85	87.61	95.98	98.43	90.49
token-sent. attention	95.97	**97.96**	**88.51**	**96.23**	**98.74**	90.60

Table 2: IterNLU (N=5) performance with different types of intent information extracted to facilitate slot filling.

F1 as expected, but also brings higher intent detection accuracy, possibly due to backpropagation. By also feeding slot information into the intent module, IterNLU(N=2) obtains further enhanced performances on slot filling and intent detection. These observations illustrate the benefit of intent-slot information sharing. Regarding iterative learning, more iterations in general lead to small but steady performance improvements on ATIS. This is unsurprising since while IterNLU(*N*=1) can already handle most sentences, only those challenging sentences need iterative understanding. The more challenging a sentence is, the more iterations are needed. While the iteration number increases, the number of those applicable challenging sentences decreases. This leads to small but cumulatively substantial improvement for iterations.

On Snips, the Parallel Implementation already outperforms all baseline algorithms, indicating the benefit of module design. With IterNLU, the performances are substantially further improved in all the three metrics if *N*=2 is used (where intent-slot correlation in both directions is explored). However, in contrast to the case of ATIS, the performance change of IterNLU becomes noisy on Snips when *N* is increased further. This is possibly because the sentences in Snips are relatively simple, and the benefit of using more iterations to handle challenging sentences may be diluted by the added confusion from unnecessary computation on simple sentences. The performance differences on ATIS and Snips show that for IterNLU, the repetition number *N* should be chosen based on the characteristics of the target data, i.e., be tuned as a hyperparameter on the development set.

5 Analysis

5.1 Effect of Intent Information Extraction

In this subsection, we investigate the impact of extracting different types of intent representation (i.e., i_k) to facilitate subsequent slot filling on the performance of the IterNLU framework. With the remaining part of the framework fixed, we evaluate five variants of the intent representation, including (1) *token-sentence attention*: the proposed attention-based intent representation described in section 3.2, (2) *token-wise hidden rep.*: $[h_k^f, h_k^b]$ generated by the intent BiLSTM, (3) *sentence hidden rep.*: the sentence-level hidden representation h_{sent}, (4) *intent prediction (prob.)*: the intent prediction vector (showing probability distribution over all the intent labels) generated by the linear prediction layer on top of the intent BiLSTM, and (5) *intent prediction (one-hot)*: the one-hot representation converted from the intent prediction vector by setting the dimension of the predicted intent as 1 and others as 0. Except for *token-wise hidden rep.*, all the other variants are sentence-level intent representations, each broadcast into the subsequent slot module as all the i_k ($k = 1...n$). The resulting IterNLU performance using these intent information variants are shown in Table 2. To facilitate the discussion, in this subsection as well as the next one, we set the repetition number *N* as 5 for both training and evaluation, and report the average results across five independent runs.

From Table 2, we can see that compared with feeding in token-wise $[h_k^f, h_k^b]$, feeding in the sentence-level h_{sent} brings better improvement on slot filling. Note that token-wise intent representation has been used in the Bi-Model and Stack Propagation algorithms to either directly or indirectly (through token-level intent prediction) facilitate slot filling. This observation shows that providing intent information as reliably as possible is beneficial. Unsurprisingly, adopting the enhanced sentence intent representation, i.e., *token-sentence attention*, further improves the slot filling performance (Slot F1 increased from 95.73/95.82 to 95.97 on ATIS, and from 96.01/96.12 to 96.23 on Snips), which also benefits intent detection through iteration. In (Qin et al., 2019), it was reported that token-wise intent information performs better than sentence-level intent information to aid slot filling, as it may ease the error propagation. Our observation is inconsistent with that, possibly because the error propagation problem can be relieved by intent-slot iterations for IterNLU. We also evaluate the two prediction related intent representation variants, because directly feeding in the intent prediction result may ease the learning task for the slot module. The experimental results show that *intent prediction (prob.)* brings a best Slot F1 result on ATIS, but in general, the intent prediction variants fail to outperform the proposed *token-sentence attention* method for intent information extraction.

5.2 Effect of Slot Information Extraction

This subsection evaluates the IterNLU performance with different types of slot information extracted to facilitate the subsequent intent detection, with other parts of the framework fixed. Slot informa-

	ATIS Dataset			Snips Dataset		
Slot Information	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**	**Slot (F1)**	**Intent (Acc.)**	**Overall (Acc.)**
token-wise hidden rep.	95.81	97.78	87.70	**96.27**	98.69	**90.94**
slot prediction (prob.)	95.84	**98.02**	88.15	96.01	98.43	90.54
slot prediction (one-hot)	**95.97**	97.96	**88.51**	96.23	**98.74**	90.60

Table 3: IterNLU (N=5) performance with different types of slot information extracted to facilitate intent detection.

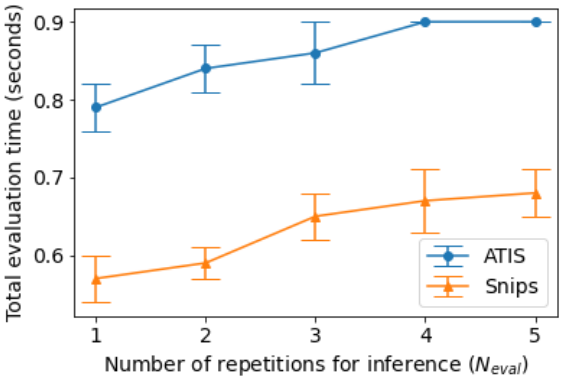

Figure 3: Inference time with varying N_{eval}.

tion is token-wise by nature. We evaluated three slot representation variants in this work: (1) *slot prediction (one-hot)*: the proposed one-hot slot prediction representation described in section 3.3, (2) *slot prediction (prob.)*: the slot prediction vector o_k, and (3) *token-wise hidden rep.*: the token-wise hidden representation h_k^{slot}. The comparison results are shown in Table 3. On ATIS data, using slot prediction representations leads to better intent detection performance compared with adopting the token-wise hidden representation, and *slot prediction (one-hot)* achieves the best overall accuracy. However, for Snips, *token-wise hidden rep.* obtains relatively similar Slot F1 and intent accuracy as *slot prediction (one-hot)*, while achieving the best overall accuracy. The discrepancy is probably due to the difference in slot distribution between ATIS and Snips datasets. It indicates that selecting a suitable slot representation based on the target data could be beneficial.

5.3 System Efficiency

We evaluate the training/inference efficiency of the IterNLU framework using a single NVIDIA Geforce RTX 2080 GPU. The training procedure takes about 1.1 and 1.6 hours on average for ATIS and Snips, respectively, and is insensitive to the repetition number used (N ranging from 1 to 5). For inference, given a trained model, iterating the intent detection and slot filling modules various times leads to different efficiencies. Figure 3 illustrates the impact of the repetition number used for inference, denoted as N_{eval}, on inference time, evaluated on the ATIS and Snips testing sets.

One interesting observation is that given a trained model, varying N_{eval} for inference leads to almost the same prediction performance. This is mainly because all parameters are tied for the intent detection modules and for the slot filling modules. The knowledge learned through iterated training is thus captured in the parameters of the intent/slot module. Note that if the parameters are not tied, iterations will benefit both training and inference in general. For the proposed approach with tied parameters, we thus recommend to adopt multiple repetitions (e.g., $N = 5$ for ATIS and $N = 2$ for Snips) for the training of IterNLU, and only run the intent detection module followed by the slot filling module once for inference. In this way, we can maximize the inference efficiency with negligible performance loss. For IterNLU, the majority (i.e., 95.8%) of parameters come from the BERT embedding module. The benefit of setting $N_{eval} = 1$ on inference efficiency will be even more significant if the BERT is replaced by a parameter-efficient alternative such as DistillBERT (Sanh et al., 2019; Jiao et al., 2019; Iandola et al., 2020) in the framework.

6 Conclusion

This paper proposes a new NLU framework, called IterNLU, to iteratively learn intent detection and slot filling, with not only intent but also slot information explicitly extracted to aid the subsequent counterpart processing. The aim is to fully capture the intent-slot correlation to benefit NLU. Our experimental results show that the proposed approach outperforms previous state-of-the-art NLU methods on both ATIS and Snips datasets. Systematic analyses of the IterNLU framework on the effect of various intent/slot information extraction and system efficiency are also conducted in this work.

References

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

A. Coucke, A. Saade, Adrien Ball, Théodore Bluche, A. Caulier, D. Leroy, Clément Doumouro, Thibault Gisselbrecht, F. Caltagirone, Thibaut Lavril, Maël Primet, and J. Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ArXiv*, abs/1805.10190.

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

Haihong E, Peiqing Niu, Zhongfu Chen, and Meina Song. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5467–5471.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE.

Patrick Haffner, Gokhan Tur, and Jerry H Wright. 2003. Optimizing svms for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 1, pages I–I. IEEE.

Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Interspeech*, pages 715–719.

Homa B Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In *International Conference on Web Search and Data Mining, Workshop on Query Understanding*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Forrest N Iandola, Albert E Shaw, Ravi Krishna, and Kurt W Keutzer. 2020. Squeezebert: What can computer vision teach nlp about efficient neural networks? *arXiv preprint arXiv:2006.11316*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Tom Kenter and Maarten de Rijke. 2017. Attentive memory networks: Efficient machine reading for conversational search. *arXiv preprint arXiv:1712.07229*.

Changliang Li, Liang Li, and Ji Qi. 2018. A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, pages 685–689.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, pages 591–598.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2078–2087.

Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. AGIF: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NIPS*.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *NAACL-HLT (2)*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Di Wu, Liang Ding, Fan Lu, and Jian Xie. 2020. SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1932–1937, Online. Association for Computational Linguistics.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*, pages 78–83. IEEE.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and S Yu Philip. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, volume 16, pages 2993–2999.

A Box plots

Given the randomness involved in the evaluation of neural network based approaches, box plots can be helpful in differentiating algorithms based on their performance. We thus represent the comparison results in Section 4.3 with box plots as well, in addition to Table 1. For each dataset, we select the top five baseline approaches in terms of overall accuracy from Table 1, and compare them with the best IterNLU model (i.e., IterNLU(N=5) for ATIS; IterNLU(N=2) for Snips) in the box plots. We also include the Parallel Implementation (denoted as "IterNLU (parallel implementation)" in the box-plots figures) to illustrate the benefit of iterative learning with intent-slot information sharing. The resulting box plots, which are drawn with Plotly library[2] (version 4.14.3) in Python3, are listed in the two figures below for ATIS and Snips, respectively. In the box plots, the solid line inside each box indicates the median, while the dashed line indicates the mean.

[2]https://plotly.com/

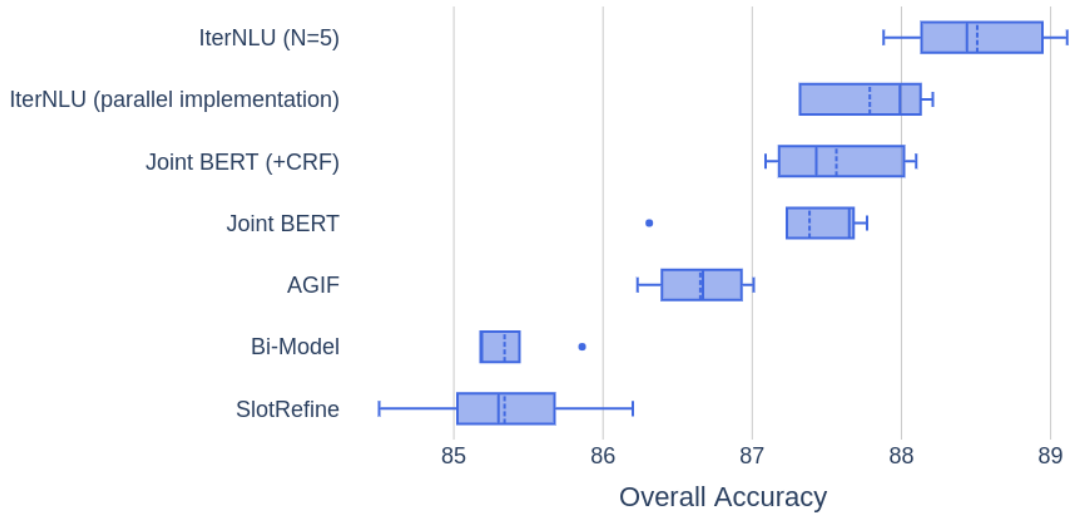

Figure 4: Performance comparison of the best IterNLU model with the top-5 baseline approaches and the Parallel Implementation, evaluated in terms of overall accuracy (%) on ATIS data.

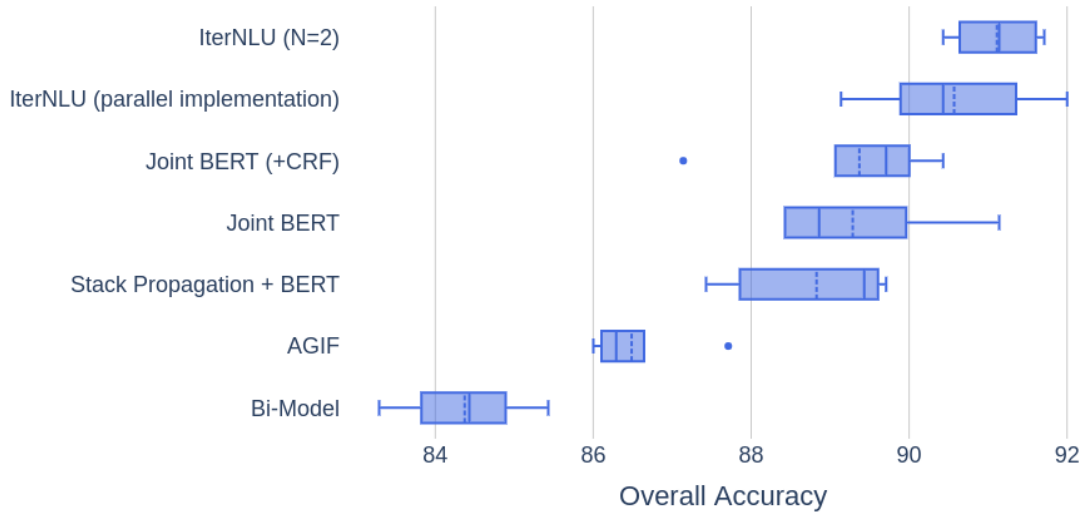

Figure 5: Performance comparison of the best IterNLU model with the top-5 baseline approaches and the Parallel Implementation, evaluated in terms of overall accuracy (%) on Snips data.