Effective Slot Filling via Weakly-Supervised Dual-Model Learning

Jue Wang[1], **Ke Chen**[1], **Lidan Shou**[1,2*], **Sai Wu**[1], **Gang Chen**[1,2]
[1]College of Computer Science and Technology, Zhejiang University
[2]State Key Laboratory of CAD&CG, Zhejiang University
{zjuwangjue,chenk,should,wusai,cg}@zju.edu.cn

Abstract

Slot filling is a challenging task in Spoken Language Understanding (SLU). Supervised methods usually require large amounts of annotation to maintain desirable performance. A solution to relieve the heavy dependency on labeled data is to employ bootstrapping, which leverages unlabeled data. However, bootstrapping is known to suffer from semantic drift. We argue that semantic drift can be tackled by exploiting the correlation between slot values (phrases) and their respective types. By using some particular weakly-labeled data, namely the plain phrases included in sentences, we propose a weakly-supervised slot filling approach. Our approach trains two models, namely a classifier and a tagger, which can effectively learn from each other on the weakly-labeled data. The experimental results demonstrate that our approach achieves better results than standard baselines on multiple datasets, especially in the low-resource setting.

1 Introduction

Slot filling is an essential and challenging task in Spoken Language Understanding (SLU). The task is usually interpreted as a sequence tagging process, during which slot values, in the form of short phrases (such as named entities), and their corresponding slot types are annotated. For example, as shown in Table 1, there are three slot values in the user utterance, namely "Atlanta", "Toronto", and "Friday afternoon", whose corresponding slot types are departure location, arrival location, and departure time respectively.

Although some existing supervised approaches (Raymond and Riccardi 2007; Yao et al. 2014; Mesnil et al. 2015) have achieved good results on slot filling, they usually require large amounts of annotated data, which is hardly available in real-world applications. Actually, acquiring labels is costly, and probably the biggest obstacle to the application of these methods. This motivates the need for effective learning techniques that leverage unlabeled data.

Bootstrapping is a popular approach using unlabeled data. The main idea is to extend new annotations based on existing annotations. However, this method may quickly introduce *semantic drift* (Curran, Murphy, and Scholz 2007): one mistaken label may cause even more wrong predictions in the subsequent iterations, causing the semantics of slot types to deviate from its original definition. For the running example in Table 1, a model could accidentally recognize both location mentions ("Atlanta" and "Toronto") as *departure locations*. Such results are probably wrong because a person can only be in one departure location at a time. With this error undetected, the slot type *departure location* may gradually drift from its original meaning. And in an extreme case, the model may consider all location mentions to be of this type, yet the loss could still be very small!

Utterance:	*List*	*flights*	*from*	*Atlanta*	*to*	*Toronto*	*Friday*	*afternoon*
Slot Tags:	O	O	O	B-dep.loc	O	B-arr.loc	B-dep.time	I-dep.time
Phrases:	O	O	O	B	O	B	B	I

Table 1: An example of annotation for slot filling.

To avoid the above problem, we advocate leveraging some special weakly-labeled data. Our approach relies on a key observation that, compared to slot type and value annotations, untyped plain phrases (i.e., text chunks, as shown at the bottom of Table 1) are much easier to acquire, since almost all phrases in a sentence (except for auxiliary words) can be regarded as potential slot values. Given an utterance and phrases included in it, we can always determine the slot type for each phrase. For example, if a user wants to book an airline ticket, the input utterance may contain phrases such as the departure location, the arrival location, the departure date and time, etc. With the plethora of off-the-shelf tools for text chunking, phrases without slot labels can be collected in large numbers.

In this paper, we assume that the dataset is partially annotated, that is, only a small number of utterances are properly labeled with both slot values and their respective slot types, while the rest are all weakly-labeled with phrases. To tackle *semantic drift*, which typically appears as wrong slot type in our task output, we propose a solution comprising two models in their own task formulations: a *Classifier* that predicts the slot type given a phrase (slot value), and a *Tagger* that predicts the phrase (slot value) given a slot type. We design a novel training target on which both models can be collaboratively trained using plain phrases without slot labels. The need for preventing semantic drift justifies our dual-model approach, since, only if the two models produce consistent

[*] Corresponding author

results, will the joint loss be minimized, which reduces the possibility of *wrong slot type* caused by a single model.

We summarize the contributions of our paper as follows:

First, we propose a weakly-supervised dual-model learning approach for slot filling. Supplied by very limited labeled utterances, the models can be effectively trained on large sets of weakly-labeled phrases.

Second, we explore variants of our method, including one that requires no additional parameters to ensure that our main idea can be easily integrated into existing slot filling models.

Third, we perform extensive experiments over several datasets, and the results show that our approach outperforms conventional supervised methods as well as bootstrapping methods, especially when given very few labeled data.[1]

2 Related Work

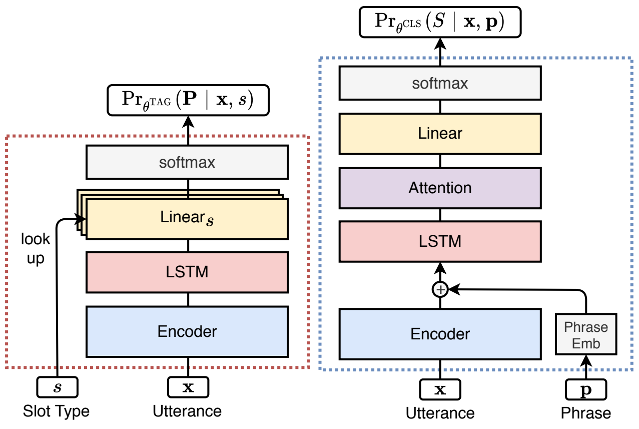
Slot filling is usually treated as a sequence tagging task. Standard approaches include MEMMs (McCallum, Freitag, and Pereira 2000) and CRFs (Raymond and Riccardi 2007). Many researchers (Mesnil et al. 2013, 2015; Yao et al. 2013, 2014) apply RNNs (Huang, Xu, and Yu 2015; Lample et al. 2016) to this task and have promising achievements. Extensions include encoder-decoder models (Liu and Lane 2016; Zhu and Yu 2017), memory networks (Chen et al. 2016), slot-gated model (Goo et al. 2018), "label-recurrent" model (Gupta, Hewitt, and Kirchhoff 2019), SF-ID interrelated model (Haihong et al. 2019), capsule networks (Zhang et al. 2019), and stack-propagation model (Qin et al. 2019). However, these approaches are sensitive to the size of training data, and cannot achieve acceptable results given very few labeled samples.

To address this issue, some work focuses on exploiting external knowledge via transfer learning (Yang, Salakhutdinov, and Cohen 2017), so they can quickly bootstrap the model in a new domain with only a handful labeled data (Fritzler, Logacheva, and Kretov 2019) or even without any data (Bapna et al. 2017; Shah et al. 2019; Lee and Jha 2019).

Bootstrapping (Yarowsky 1995) is a well-known technique which leverages unlabeled data. The main idea is to extend new annotations based on the existing annotations. Lee (2013) propose to use pseudo labels generated by the model as if they were true labels. Thenmalar, Balaji, and Geetha (2015) define the pattern with a small set of training data, and then use it as a seed pattern to generate new patterns. Co-training (Blum and Mitchell 1998; Nigam et al. 2000) is similarly motivated but uses a pair of classifiers to iteratively learn and generate additional training labels. In mutual learning (Zhang et al. 2018), multiple models are trained jointly for the same task by minimizing the KL-divergence between their predictions.

Another related technique to leverage unlabeled data is Π-model (Sajjadi, Javanmardi, and Tasdizen 2016) where the input is perturbed with noise, and the model is then required to produce similar predictions with and without perturbation. Miyato et al. (2018) proposed an adversarial training strategy to generate noise.

[1]Our code is available at https://github.com/LorrinWWW/weakly-supervised-slot-filling



(a) Tagger (b) Classifier

Figure 1: Dual-model overview.

Dual learning (He et al. 2016) is first proposed for neural machine translation. It is also applied in SLU (Su, Huang, and Chen 2019), where natural language generation (NLG) is the dual task. Zhu, Cao, and Yu (2020) propose a semi-supervised learning approach to slot filling which improves performance by integrating a dual task of semantic-to-sentence generation. However, the work is based on a *very strong* assumption that semantic forms (namely intents and lists of slot-value pairs) are available. Given such slot-value pairs as a gazetteer alone, one can easily build a high-quality slot filler even by keyword lookup. In contrast, our method only leverages plain phrases produced by off-the-shelf chunking tools.

3 Approach

3.1 Model

We describe the model structure of *Tagger* and *Classifier* in this section. Both models share the same text encoder, which maps each word in the input utterance to a fixed-dimensional vector. Figure 1 presents the overview. *Classifier* predicts the slot type for the given phrase. Meanwhile, *Tagger* predicts the slot value for the given slot type. Being two independent parts, however, they provide each other with supervision information via a dual-model learning mechanism, as will be described in the next subsection.

Encoder We first introduce the text encoder used in *Tagger* and *Classifier*, shown in Figure 1. For an utterance containing n words $\boldsymbol{x} = [x_{(i)}]_{i=0}^{n-1}$, we define the word embeddings $\boldsymbol{x}^w \in \mathbb{R}^{n \times d_1}$, as well as character embeddings $\boldsymbol{x}^c \in \mathbb{R}^{n \times d_2}$ computed by an LSTM (Lample et al. 2016). We also consider the contextualized word embeddings $\boldsymbol{x}^l \in \mathbb{R}^{n \times d_3}$, which is produced from pre-trained language models such as BERT (Devlin et al. 2019).

We concatenate those embeddings for each word and use a linear layer to reduce the embedding size and a bidirectional LSTM to compute the final word representation $\tilde{\boldsymbol{x}} \in \mathbb{R}^{n \times d}$:

$$\tilde{\boldsymbol{x}} = \text{LSTM}(\text{Linear}([\boldsymbol{x}^c; \boldsymbol{x}^w; \boldsymbol{x}^l])) \tag{1}$$

where each word is represented as an d dimensional vector.

Tagger Figure 1a presents the structure of *Tagger*. Given an utterance $\boldsymbol{x}$ and a slot type s, it needs to predict tags for every word in BIO scheme[2] (Ratinov and Roth 2009), indicating the phrase boundary of slot type s.

We first use a bidirectional LSTM to contextualize the word embeddings $\tilde{\boldsymbol{x}}$ defined at Equation 1. And for each slot type, we use a distinct linear layer to compute the score for tag B, I, and O. Formally, we have:

$$\boldsymbol{logits}^{\text{TAG}} = \text{Linear}_s(\text{LSTM}(\tilde{\boldsymbol{x}})) \tag{2}$$

so the tag probability distribution is:

$$\Pr_{\theta^{\text{TAG}}}(\boldsymbol{P}|\boldsymbol{x}, s) = \text{softmax}(\boldsymbol{logits}^{\text{TAG}}) \tag{3}$$

where θ^{TAG} is the set of parameters related to *Tagger*; Linear_s means the model will use different parameters according to the given slot type s; $\boldsymbol{P} = [P_{(i)}]_{i=0}^{n-1}$ is a fixed-length sequence of random variables on BIO tags.

Classifier Figure 1b illustrates the model structure of *Classifier*. Given an utterance $\boldsymbol{x}$ and a phrase $\boldsymbol{p} = [p_{(i)}]_{i=0}^{n-1}$ in it, *Classifier* needs to predict the slot type of the phrase.

We use the text encoder to encode the sequence of words in the utterance, denoted $\tilde{\boldsymbol{x}}$. And the input phrase $\boldsymbol{p}$ is given in BIO tags, and we map each tag to a learnable embedding, so the phrase $\boldsymbol{p}$ can be represented with embeddings $\tilde{\boldsymbol{p}}$. We add the utterance embeddings $\tilde{\boldsymbol{x}}$ and phrase embeddings $\tilde{\boldsymbol{p}}$ word-wise, and then feed them to a bidirectional LSTM to obtain the contextualized hidden vectors $\boldsymbol{h} = [h_{(i)}]_{i=0}^{n-1}$:

$$\boldsymbol{h} = \text{LSTM}(\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{p}}) \tag{4}$$

Next, we use attention to aggregate the sequence of vectors into one vector:

$$\tilde{h} = \sum_{0 \le i < n} \alpha_{(i)} h_{(i)}, \alpha_{(i)} = \frac{\exp(h_{(i)} \cdot v^{\alpha})}{\sum_{0 \le j < n} \exp(h_{(j)} \cdot v^{\alpha})} \tag{5}$$

where v^{α} is a learnable vector.

Finally, we use a linear layer to calculate the slot type probability distribution of the input phrase:

$$\textit{logits}^{\text{CLS}} = \text{Linear}(\tilde{h}) \tag{6}$$

$$\Pr_{\theta^{\text{CLS}}}(S|\boldsymbol{x}, \boldsymbol{p}) = \text{softmax}(\textit{logits}^{\text{CLS}}) \tag{7}$$

where θ^{CLS} is the set of parameters related to *Classifier*, and S is a random variable of the slot type.

3.2 Training

We explore the weakly-supervised learning scenario, where the raw dataset is available in two groups: 1) utterances with correctly labeled slots, and 2) unlabeled utterances. We then use off-the-shelf text chunking tools to extract phrases from the unlabeled utterances (see §4.1 for details), which then become *weakly-labeled data* for dual-model learning. Usually, the labeled utterances are scarce but they provide examples to define slot types. In contrast, the weakly-labeled data (i.e., phrases) are abundant as they can be easily acquired, but their information is weak, and cannot directly fulfill the need for supervising slot filling.

[2] B-tag for the beginning word of a text span, I-tag for other words inside a text span, and O-tag for words outside any spans.

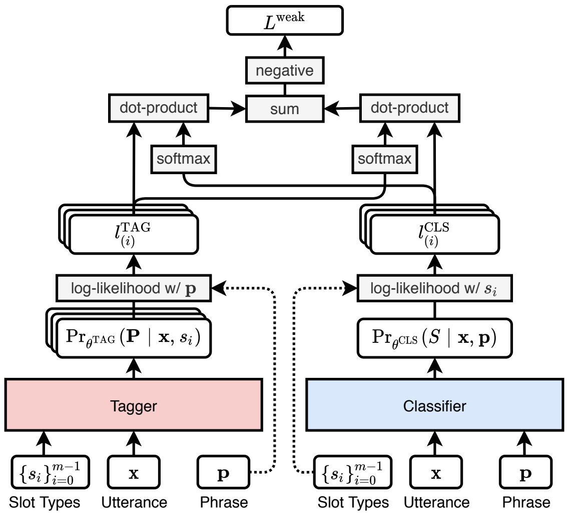


Figure 2: Weakly-supervised learning part.

Supervised Learning We begin with the supervised learning setting. Given correctly labeled data, the learning process is straightforward – to train the two models individually. For both models, we use the prevalent cross-entropy loss.

Given an utterance $\boldsymbol{x}$, a phrase $\boldsymbol{p}$ in it and its slot type s ($s \in \{s_i\}_{i=0}^{m-1}$, where m is the number of slot types), we define the loss for *Classifier*:

$$L^{\text{CLS}} = -\log \Pr_{\theta^{\text{CLS}}}(S = s|\boldsymbol{x}, \boldsymbol{p}) \tag{8}$$

Similarly, we define the loss for *Tagger*, which is the sum of cross-entropy between the prediction of each word and its corresponding phrase tag:

$$L^{\text{TAG}} = -\sum_{0 \le i < n} \log \Pr_{\theta^{\text{TAG}}}(P_{(i)} = p_{(i)}|\boldsymbol{x}, s) \tag{9}$$

We add the two losses to obtain the final loss:

$$L^{\text{sup}} = L^{\text{CLS}} + L^{\text{TAG}} \tag{10}$$

Note that we also keep the negative samples here. A negative sample is when the utterance $\boldsymbol{x}$ does not contain any valid slot value for slot type s, so the input phrase $\boldsymbol{p}$ is an all-O-tag sequence. In this case, L^{CLS} is simply discarded.

Weakly-Supervised Learning Before introducing the weakly-supervised learning part, we first rewrite the supervised learning loss in another form. For *Classifier*, we have:

$$\begin{aligned} L^{\text{CLS}} &= -\log \Pr_{\theta^{\text{CLS}}}(S = s|\boldsymbol{x}, \boldsymbol{p}) \\ &= -\sum_{0 \le i < m} \mathbb{1}_{\{s\}}(s_i) \log \Pr_{\theta^{\text{CLS}}}(S = s_i|\boldsymbol{x}, \boldsymbol{p}) \\ &= -\sum_{0 \le i < m} \mathbb{1}_{\{s\}}(s_i)\, \ell_{(i)}^{\text{CLS}} \end{aligned} \tag{11}$$

where $\mathbb{1}$ is an indicator function defined as $\mathbb{1}_E(e) = 1$ if $e \in E$, and 0 otherwise; $\ell_{(i)}^{\text{CLS}}$ represents the log-likelihood between slot type s_i and the prediction.

Similarly, for *Tagger*, we have:

$$\begin{aligned} L^{\text{TAG}} &= - \sum_{0 \le j < n} \log \Pr_{\theta^{\text{TAG}}}(P_{(j)} = p_{(j)} | \boldsymbol{x}, s) \\ &= - \sum_{0 \le i < m} \mathbb{1}_{\{s\}}(s_i) \sum_{0 \le j < n} \log \Pr_{\theta^{\text{TAG}}}(P_{(j)} = p_{(j)} | \boldsymbol{x}, s_i) \\ &= - \sum_{0 \le i < m} \mathbb{1}_{\{s\}}(s_i) \, \ell^{\text{TAG}}_{(i)} \end{aligned} \tag{12}$$

where $\ell^{\text{TAG}}_{(i)}$ represents the log-likelihood between phrase $\boldsymbol{p}$ and the prediction for slot type s_i.

Now the total loss at Equation 10 can be derived as:

$$L^{\text{sup}} = - \sum_{0 \le i < m} (\mathbb{1}_{\{s\}}(s_i) \, \ell^{\text{CLS}}_{(i)} + \mathbb{1}_{\{s\}}(s_i) \, \ell^{\text{TAG}}_{(i)}) \tag{13}$$

Note that only term $\mathbb{1}_{\{s\}}(s_i)$ depends on the gold slot type s, which might be unavailable in the weakly-supervised data.

To compensate for the missing supervision in these data, our weakly-supervised learning is motivated by the following ideas: 1) If the prediction of *Tagger* for slot type s_i is very close to the given phrase $\boldsymbol{p}$ (i.e., $\ell^{TAG}_{(i)}$ is large), then the phrase is likely to belong to this slot type; 2) If the relevance between slot type s_i and phrase $\boldsymbol{p}$, estimated by *Classifier*, is very high (i.e., $\ell^{CLS}_{(i)}$ is large), then *Tagger* should be able to retrieve phrase $\boldsymbol{p}$ for this slot type. Therefore, we can allow the two models to supervise each other, so as to compensate for the missing information $\mathbb{1}_{\{s\}}(s_i)$.

As depicted in Figure 2, we give the loss as follows:

$$L^{\text{weak}} = - \sum_{0 \le i < m} (r^{\text{TAG}}_{(i)} \ell^{\text{CLS}}_{(i)} + r^{\text{CLS}}_{(i)} \ell^{\text{TAG}}_{(i)}) \tag{14}$$

$$\boldsymbol{r}^{\text{TAG}} = \text{softmax}([\ell^{\text{TAG}}_{(i)}]_{i=0}^{m-1}) \tag{15}$$

$$\boldsymbol{r}^{\text{CLS}} = \text{softmax}([\ell^{\text{CLS}}_{(i)}]_{i=0}^{m-1}) \tag{16}$$

where we may notice $r^{\text{CLS}}_{(i)} = \Pr_{\theta^{\text{CLS}}}(S = s_i | \boldsymbol{x}, \boldsymbol{p})$.

During training, specifically for the weakly-labeled utterances, the optimizer will mainly maximize $\ell^{\text{TAG}}_{(i)}$ when $\ell^{\text{CLS}}_{(i)}$ is large, and maximize $\ell^{\text{CLS}}_{(i)}$ when $\ell^{\text{TAG}}_{(i)}$ is large. Consequently, the total loss is minimized only when *Classifier* and *Tagger* agree with each other, i.e., *Classifier* chooses the correct slot type and *Tagger* predicts phrase $\boldsymbol{p}$ for that slot type simultaneously.

Besides, another important feature of our method is that, it is encouraged to categorize phrases inside the same utterance into different slot types. For instance, assuming *Classifier* classifies two different phrases into the same slot type s_0, since we have $L \ge -r^{\text{CLS}}_{(0)} \ell^{\text{TAG}}_{(0)}$, to keep the loss small, the optimizer should at least ensure $\ell^{\text{TAG}}_{(0)}$ to be reasonably large. However, since the predicted phrase by *Tagger* for slot type s_0 is always the same, we cannot keep the likelihood with the two phrases large at the same time. As a result, as long as the algorithm converges, i.e., the training loss is reasonably small, our method will attempt to capture the subtle differences of phrases in the same context.

Note that conventional bootstrapping approach is prone to the problem of *semantic drift* when encountering similar slot types. In the scenario of slot filling, similar slot types frequently co-occur in the same utterances, such as departure location and arrival location. Therefore, the above-mentioned feature becomes a strong advantage over bootstrapping in mitigating semantic drift in our task, as it helps disambiguate phrases of similar slot types.

Joint Training For each training step, we sample from labeled utterances and weakly-labeled utterances, and we train models jointly by minimizing the total loss:

$$L = L^{\text{sup}} + \lambda L^{\text{weak}} \tag{17}$$

where λ is a hyperparameter weighting the importance of the weakly-supervised part.

4 Experiments

4.1 Data

We evaluate the performance of our method on three different datasets, namely SNIPS (Coucke et al. 2018), ATIS (Hemphill, Godfrey, and Doddington 1990; Tur, Hakkani-Tür, and Heck 2010) and MIT Rest. (Liu et al. 2013). We use the standard train-dev-test split for these datasets. For ATIS and MIT Rest., since they do not have a standard development set, we randomly pick 10% of the original training set as the development set.

To simulate low-resource scenarios, we randomly retain a small portion of the original training set as labeled data. We use all the utterances in the original training set to generate weakly-labeled data. Though text chunks are not directly available, they are not expensive to acquire. Briefly, with plain sentences available, we use off-the-shelf chunking tools, e.g., AutoPhrase[3] (Shang et al. 2018), Flair[4] (Akbik et al. 2019) and rule-based methods (e.g., gazetteers and regex rules), filtered by task specific stop-words. We merge all the outputs by majority voting to obtain the collection of plain phrases – retain text chunks that are extracted by at least half of all chunking methods.

Since our method is tolerant to possibly missing candidate slot values (meaning its performance will not be hurt much, see §4.6 for details), we give priority to ensuring the precision and, for this purpose, recall fewer phrases.

4.2 Setup

Model Setting For each mini-batch, we sample 30 utterances from labeled data and from weakly-labeled data. GloVe word vectors (Pennington, Socher, and Manning 2014) are used to initialize word embeddings, which are tuned during training. We also use BERT (bert-large-uncased, fixed without fine-tuning) to produce contextualized embeddings concatenated after the word embeddings. We set the hidden size to 200, and since we use bidirectional LSTMs, the hidden size for each LSTM is 100. We also apply 0.3 dropout after embeddings and LSTMs to mitigate the over-fitting issue. We use Adam with a learning rate of 1e-3 to train the model.

[3]https://github.com/shangjingbo1226/AutoPhrase

[4]https://github.com/zalandoresearch/flair

Dataset	SNIPS				ATIS				MIT Rest.			
# labeled utterances	100	300	1000	all	100	300	1000	all	100	300	1000	all
SF-ID Network	41.3	59.7	77.6	93.7	69.5	84.5	92.4	95.5	42.5	55.2	67.0	77.2
Stack-Propagation	44.7	64.3	81.1	94.5	69.6	86.5	92.9	95.7	48.7	59.9	68.8	78.4
JointBERT	62.8	78.8	89.4	**97.0**	79.9	89.4	94.1	**96.1**	53.1	69.1	75.7	79.3
LSTM+BERT	61.7	77.5	88.4	95.7	77.0	88.3	94.2	95.9	58.7	68.3	75.1	78.8
LSTM+CRF+BERT	62.2	77.7	88.4	95.6	77.4	88.6	94.0	95.9	59.2	68.8	75.7	**79.8**
Pseudo-Label	67.4	81.1	90.6	95.5	78.6	89.3	94.2	95.9	59.4	69.6	75.9	79.5
VAT	64.4	83.4	90.5	95.4	75.8	88.1	94.0	95.9	61.1	70.5	77.2	79.5
DML	53.9	82.7	90.9	95.5	76.5	88.3	94.9	95.9	62.7	71.8	77.5	79.6
ours w/o BERT	80.3	88.5	92.5	95.1	80.9	90.8	94.6	95.9	67.2	72.4	75.1	78.4
ours	**82.0**	**91.4**	**93.2**	95.4	**81.1**	**91.1**	**95.0**	95.9	**73.0**	**75.4**	**78.0**	78.7

Table 2: Comparison with different number of labeled utterances. The reported metric is F1 score.

Metric Following the established line of work, we report the F1 score, where a slot filling prediction is considered correct if the boundary and slot type are both correct. We report the metrics averaged on 5 runs. And for each run, we save the model checkpoint that achieves the highest F1 score on the dev set, and report its score on the test set.

Evaluation Both *Tagger* and *Classifier* can be used independently. But to ensure the test conditions are consistent with previous work, we will only use *Tagger*. Specifically, for a given utterance, we input all slot types into *Tagger* in turn to obtain a complete slot filling output. Since the datasets used do not contain overlapping slot values, when two predicted slot values overlap with each other, we preserve the one with higher confidence and discard the other one.

4.3 Baselines

We compare our approach with the following baselines:

SF-ID Network Haihong et al. (2019) introduced an SF-ID network to establish direct connections for slot filling and intent detection to help them improve each other mutually. We further enhance the model with GloVe embeddings to improve their performance under the low-resource setting.

Stack-Propagation Qin et al. (2019) proposed a joint model with Stack-Propagation to better incorporate the intent information for slot filling. We also enhance it with GloVe.

JointBERT Chen, Zhuo, and Wang (2019) fine-tuned BERT to get a joint model for slot filling and intent detection.

LSTM+BERT (+CRF) LSTM-based model is a simple yet still powerful baseline for sequence tagging tasks. In our implementation, it uses the same input features as our approach.

Pseudo-Label Lee (2013) used pseudo labels, the classes that have the maximum predicted probability as if they were true labels. We adapt the idea to LSTM+BERT, so that it can be used for slot filling.

VAT Miyato et al. (2018) proposed a virtual adversarial loss that can be applied to unlabeled data. We apply it to LSTM+BERT, and evaluate in slot filling datasets.

DML Zhang et al. (2018) proposed deep mutual learning, using two identical classifiers to supervise each other by minimizing their outputs' KL-divergence. We apply it to LSTM+BERT so as to evaluate for slot filling.

4.4 Main Results

Table 2 shows that the performance of different methods varies with the number of labeled utterances. For supervised methods, we only use labeled utterances for training, and we also retain the intent labels of these utterances for systems that are designed for joint intent detection and slot filling. However, due to limited training resources, all these baselines do not perform very well. And only when the training data become relatively abundant, can these methods gradually manifest their advantages. *Pseudo-Label*, *VAT* and *DML* leverage unlabeled data and obtain some improvements, however, their performance is still unsatisfactory.

In comparison, in the low-resource settings, our method achieves significant improvements over the baselines. When we increase the number of labeled utterances, the performance margins tend to narrow down, but our method still achieves the best scores, except for the all-retained setting, where all utterances are correctly labeled.

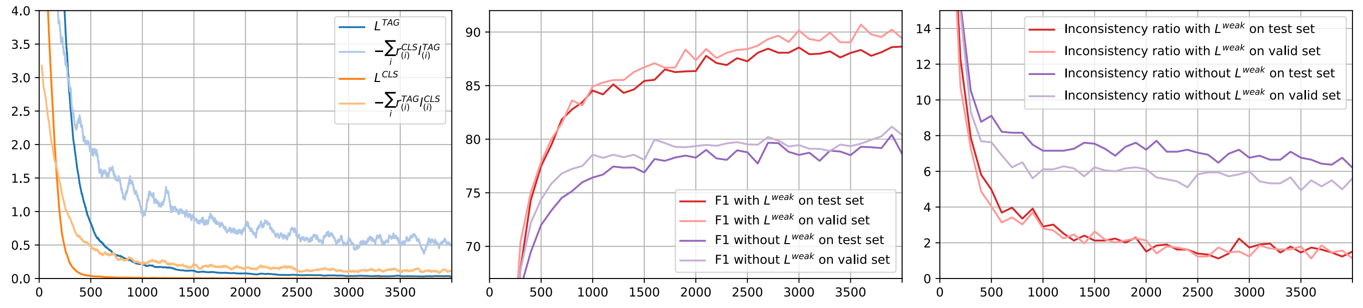
Besides, one important advantage is that our method mitigates *semantic drift* (as discussed in §3.2), which occurs in bootstrapping methods. We investigate it with case study:

Case 1: "find a show called friday download"

It contains an object type "show" and an object name "friday download". However, *Pseudo-Label* accidentally recognized both of them as object types, and in the next several iterations, the model continuously strengthened this bias and led to *semantic drift*, where the model incorrectly expanded the definition of object type and included part of the object name. In comparison, in our method, *Tagger* and *Classifier* diverged in classifying these two phrases at first, but finally agreed that "show" is closer to the object type, and thus the other phrase is more likely to be an object name.

Case 2: "... the movie time for the marcus corporation"

It contains a location name "marcus corporation" and an object type "movie time". However, *Pseudo-Label* failed to recognize "marcus corporation", which might cause more



(a) Magnitude of each loss term

(b) F1 score on dev and test set

(c) Inconsistency ratio on dev and test set

Figure 3: Statistics during training. The X-axis is the training steps.

false negatives since the prediction would be treated as ground truth. In this case, the semantics of location name learned by the model has become incomplete. Nevertheless, our method correctly detected both slots.

Case 3: "... flights between san jose and houston"

It contains a fromloc.city_name "san jose" and a toloc.city_name "houston". *Pseudo-Label* recognized both fo them as toloc.city_name. Although the utterance does not explicitly show which is the departure city and which is the destination, it is certain that they cannot share the same slot type. In our method, the model successfully distinguishes the two slots.

4.5 How Does it Work?

In this section, we attempt to give some insights into how and why our learning strategy works. We conduct experiments on SNIPS, of which 300 utterances are retained as labeled data.

We especially want to know how the weakly-supervised loss (L^{weak}) affects the training process. We investigate this by plotting each part of the loss and the F1 score according to training steps, which are shown in Figure 3a and 3b. We can make the following observations: 1) In the early stage of training, since we assign the weakly-supervised part a smaller weight ($\lambda = 0.2$), the supervised part plays a major role and thus its loss drops more rapidly; 2) When the loss of the supervised part decreases to a certain level (steps 500-1000), the weakly-supervised part gradually starts to take effect; 3) Moreover, as shown in 3b, after step 1000, the F1 score growth rate of our method without weakly-supervised loss significantly slows down; meanwhile, the one having the weakly-supervised loss is still growing steadily.

Besides, we also want to know if our approach benefits from the mutual persuasion between *Classifier* and *Tagger*. Figure 3c presents the inconsistency ratio on dev and test set during training, where we consider inconsistency occurs if $\arg\max_i \ell^{\text{CLS}}_{(i)} \neq \arg\max_i \ell^{\text{TAG}}_{(i)}$. In the supervised setting (without L^{weak}), since *Tagger* and *Classifier* are trained individually, inconsistency occurs much often than the weakly-supervised setting (with L^{weak}). This result indicates that the weakly-supervised loss can effectively promote the two models to reach agreement, and therefore explains the significant improvements in F1 compared to the baselines.

Boundary Shift	0%	7%	15%	30%
LSTM+CRF+BERT	77.7	-	-	-
w/ chunk	87.4	85.1	83.1	82.2
ours	91.5	90.4	88.8	85.8
Missing Phrase	**0%**	**7%**	**15%**	**30%**
LSTM+CRF+BERT	77.7	-	-	-
w/ chunk	87.4	85.4	84.8	84.1
ours	91.5	91.3	91.1	91.0

Table 3: Effect of chunking quality. "w/ chunk": use another linear layer to predict text chunking tags. The metric is F1.

4.6 Supervision from Text Chunking

Since we use text chunking tools to extract phrases in the unlabeled utterances, this brings additional supervision information that is beneficial to the performance, especially in the low-resource setting. Its effect needs to be studied in order to understand the contribution of our method beyond this extra knowledge. So we construct an LSTM-based sequence tagging model to predict both slot tags as well as text chunk tags, in a multi-task learning manner. Specifically, we lay another linear layer on the top of the LSTM networks, and directly train on the gold text chunks (phrases).

In addition, we are also interested in the impact of chunking quality to the performance, so we apply two types of noises to the gold text chunk labels: 1) Boundary Shift: we randomly shift the left or right boundary of $e\%$ text chunk labels by one, e.g., "New York" may become "York"; 2) Missing Phrase: we randomly drop $e\%$ text chunk labels, which cannot be used in training. We investigate the effect by varying the error rate $e\%$, and the results are shown in Table 3. The experiments are conducted on SNIPS, of which 300 utterances are retained as labeled data.

In Table 3, when using gold text chunk labels, i.e., error rate is 0%, the multi-task setting ("w/ chunk") outperforms the single-task setting, which indicates that text chunk labels do facilitate better slot value detection. However, the improvement of our method does not just come from this, and in fact, "w/ chunk" still has a big performance gap with our method. Table 3 also shows the results under a certain amount of noise. For Boundary Shift, our method has been

Setting	P	R	F1
default	90.5	92.4	91.4
w/o Char Emb	90.6	92.2	91.4
w/o Word Emb	89.3	91.1	90.2
w/o GloVe	88.7	90.5	89.6
w/o BERT	87.6	89.5	88.5
w/o $r^{\text{TAG}}_{(i)}\ell^{\text{CLS}}_{(i)}$	89.1	92.1	90.6
w/o $r^{\text{CLS}}_{(i)}\ell^{\text{TAG}}_{(i)}$	86.9	81.9	84.3
w/o L^{weak}	85.4	75.8	80.3
w/o L^{weak} w/ L^{weakS}	88.9	91.8	90.3
w/o L^{weak} w/o *Cls* w/ L^{weakS}	88.5	91.6	90.0
w/o L^{weak} w/o *Cls*	84.4	75.1	79.4

Table 4: Ablation study.

affected a bit, but the decline is less than "w/ chunk", which indicates that our approach is robust enough to withstand a certain percentage of noise in the phrase; for Missing Phrase, it did not bring much negative impact to our method, which indicates that our approach can tolerate missing phrase labels and its performance will not be hurt much.

4.7 Ablation Study

We design several additional experiments to understand the effectiveness of components in our system. The experiments are conducted on SNIPS with 300 labeled utterances.

Features The features used in our experiments include word embeddings (GloVe), character embeddings, and contextualized embeddings from language models (BERT). Table 4 shows the results removing each of them. Among them, character embedding is not very useful, possibly because the SLU datasets are not case sensitive.

Loss Terms Table 4 presents the performance changes by removing loss term $r^{\text{CLS}}_{(i)}\ell^{\text{TAG}}_{(i)}$, $r^{\text{TAG}}_{(i)}\ell^{\text{CLS}}_{(i)}$, and both of them, respectively. We see that both of them have a certain contribution to the final performance, of which $r^{\text{CLS}}_{(i)}\ell^{\text{TAG}}_{(i)}$ plays a more important role. An interesting finding is that, the term $r^{\text{CLS}}_{(i)}\ell^{\text{TAG}}_{(i)}$ affects recall more, while the term $r^{\text{TAG}}_{(i)}\ell^{\text{CLS}}_{(i)}$ improves precision more. The former makes *Tagger* tend to predict more slot values, as the false positive ones can be masked by $r^{\text{CLS}}_{(i)}$. The latter makes *Tagger* more cautious since a phrase can only have one slot type.

Additional Classifier Since the dual models gain supervision from each other, an intuitive attempt is to only avail each model of self-supervision. Specifically, we define the loss:

$$L^{\text{weakS}} = -\sum_{0\le i<m} (r^{\text{CLS}}_{(i)}\ell^{\text{CLS}}_{(i)} + r^{\text{TAG}}_{(i)}\ell^{\text{TAG}}_{(i)}) \qquad (18)$$

where the "S" in "weakS" is for "self-supervision". In addition, since L^{weak} is removed and *Classifier* is not used in the evaluation, we also try to remove *Classifier* (w/o *Cls* in Table 4). In this case, the term $r^{\text{CLS}}_{(i)}\ell^{\text{CLS}}_{(i)}$ is removed from L^{weakS}.

Setting	P	R	F1
LSTM+BERT	76.9	78.1	77.5
w/ L^{weak} w/ *Cls*	89.7	91.0	90.3
w/ L^{weakS} w/o *Cls*	86.8	89.3	88.0
LSTM+CRF+BERT	77.3	78.2	77.7
w/ L^{weak} w/ *Cls*	89.5	91.0	90.3
w/ L^{weakS} w/o *Cls*	87.5	89.5	88.5
ours	90.5	92.4	91.4

Table 5: Integration with common sequence tagging models.

Table 4 shows the results. Although the results of self-supervision (w/o L^{weak} w/ L^{weakS}) seems good, they are still lower than the default dual-model learning setting. This may be due to the different induction biases of the two different models. The mutual supervision makes it possible to take advantage of this difference, thereby reducing overfitting and semantic drift in the iterative process.

4.8 Integration with Other Tagging Models

Our weakly-supervised learning technique can be integrated into the training process of a regular sequence tagging model in two adaptation schemes. Without making any change to the supervised learning part, we add a weakly-supervised part to be jointly trained, in which the regular model is considered as "*Tagger*" and supplied with phrase information. *Tagger* can be trained 1) using L^{weak} with an additional *Classifier*; or 2) using L^{weakS} without adding a *Classifier*.

Table 5 presents the results of incorporating our schemes into the prevalent LSTM and LSTM+CRF with BERT embeddings. It can be observed that, for both *LSTM+BERT* and *LSTM+BERT+CRF*, adding the weakly-supervised learning part brought significant improvements to the performance. Among them, "w/ L^{weak} w/ *Cls*" shows a clear advantage over "w/ L^{weakS} w/o *Cls*", indicating the effectiveness of the dual-model setting. However, it is still slightly worse than the original *Tagger*, mainly due to the inconsistency between the targets in different learning parts, especially when using CRF. On the contrary, our original *Tagger* does not switch between different learning targets, thus avoiding the performance loss caused by inconsistency.

5 Conclusion

In this paper, we studied the slot filling problem and used two different models in different task formulations, namely a *Classifier* that predicts the slot type given a phrase (slot value), and a *Tagger* that predicts the phrase (slot value) given a slot type. To leverage the large amounts of unlabeled data, we propose to let the dual models supervise each other in a weakly-supervised learning manner. Our experimental results showed that our method worked especially well when given very few labeled data. In the future, we would like to investigate methods that do not rely on text chunking tools as a preprocessing step. We will also try to select suitable utterances from a larger general-purpose corpus to improve the performance on full datasets.

Acknowledgments

This work was supported by the Key Research and Development Program of Zhejiang Province of China (No. 2021C01009), and the Natural Science Foundation of Zhejiang Province of China (No. LY18F020005), and the National Natural Science Foundation of China under Grant No. 61672455.

References

Akbik, A.; Bergmann, T.; Blythe, D.; Rasul, K.; Schweter, S.; and Vollgraf, R. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proc. of ACL.*

Bapna, A.; Tur, G.; Hakkani-Tur, D.; and Heck, L. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *Proc. of INTERSPEECH.*

Blum, A.; and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of COLT.*

Chen, Q.; Zhuo, Z.; and Wang, W. 2019. BERT for Joint Intent Classification and Slot Filling. *arXiv preprint arXiv:1902.10909* .

Chen, Y.-N.; Hakkani-Tür, D.; Tür, G.; Gao, J.; and Deng, L. 2016. End-to-End Memory Networks with Knowledge Carryover for Multi-Turn Spoken Language Understanding. In *Proc. of INTERSPEECH.*

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190* .

Curran, J. R.; Murphy, T.; and Scholz, B. 2007. Minimising semantic drift with mutual exclusion bootstrapping. In *Proc. of PACLING.*

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT.*

Fritzler, A.; Logacheva, V.; and Kretov, M. 2019. Few-shot classification in named entity recognition task. In *Proc. of SAC.*

Goo, C.-W.; Gao, G.; Hsu, Y.-K.; Huo, C.-L.; Chen, T.-C.; Hsu, K.-W.; and Chen, Y.-N. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proc. of NAACL-HLT.*

Gupta, A.; Hewitt, J.; and Kirchhoff, K. 2019. Simple, Fast, Accurate Intent Classification and Slot Labeling for Goal-Oriented Dialogue Systems. In *Proc. of SIGdial.*

Haihong, E.; Niu, P.; Chen, Z.; and Song, M. 2019. A novel bi-directional interrelated model for joint intent detection and slot filling. In *Proc. of ACL.*

He, D.; Xia, Y.; Qin, T.; Wang, L.; Yu, N.; Liu, T.-Y.; and Ma, W.-Y. 2016. Dual learning for machine translation. In *Proc. of NIPS.*

Hemphill, C. T.; Godfrey, J. J.; and Doddington, G. R. 1990. The ATIS spoken language systems pilot corpus. In *Proc. of HLT.*

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural Architectures for Named Entity Recognition. In *Proc. of HLT-NAACL.*

Lee, D.-H. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Proc. of Workshop on challenges in representation learning, ICML.*

Lee, S.; and Jha, R. 2019. Zero-shot adaptive transfer for conversational language understanding. In *Proc. of AAAI.*

Liu, B.; and Lane, I. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Proc. of INTERSPEECH.*

Liu, J.; Pasupat, P.; Cyphers, S.; and Glass, J. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *Proc. of ICASSP.*

McCallum, A.; Freitag, D.; and Pereira, F. C. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proc. of ICML.*

Mesnil, G.; Dauphin, Y.; Yao, K.; Bengio, Y.; Deng, L.; Hakkani-Tur, D.; He, X.; Heck, L.; Tur, G.; Yu, D.; et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE ACM Trans. Audio Speech Lang. Process.* .

Mesnil, G.; He, X.; Deng, L.; and Bengio, Y. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proc. of INTERSPEECH.*

Miyato, T.; Maeda, S.-i.; Koyama, M.; and Ishii, S. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Do. Intell.* .

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Mach. learn.* .

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *Proc. of EMNLP.*

Qin, L.; Che, W.; Li, Y.; Wen, H.; and Liu, T. 2019. A Stack-Propagation Framework with Token-Level Intent Detection for Spoken Language Understanding. In *Proc. of EMNLP-IJCNLP.*

Ratinov, L.; and Roth, D. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of CoNLL.*

Raymond, C.; and Riccardi, G. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proc. of INTERSPEECH.*

Sajjadi, M.; Javanmardi, M.; and Tasdizen, T. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proc. of NIPS.*

Shah, D.; Gupta, R.; Fayazi, A.; and Hakkani-Tur, D. 2019. Robust Zero-Shot Cross-Domain Slot Filling with Example Values. In *Proc. of ACL.*

Shang, J.; Liu, J.; Jiang, M.; Ren, X.; Voss, C. R.; and Han, J. 2018. Automated phrase mining from massive text corpora. *IEEE Trans. on Knowl. Data Eng.* .

Su, S.-Y.; Huang, C.-W.; and Chen, Y.-N. 2019. Dual Supervised Learning for Natural Language Understanding and Generation. In *Proc. of ACL*.

Thenmalar, S.; Balaji, J.; and Geetha, T. 2015. Semi-supervised bootstrapping approach for named entity recognition. *arXiv preprint arXiv:1511.06833* .

Tur, G.; Hakkani-Tür, D.; and Heck, L. 2010. What is left to be understood in ATIS? In *Proc. of SLT*.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. In *Proc. of ICLR*.

Yao, K.; Peng, B.; Zhang, Y.; Yu, D.; Zweig, G.; and Shi, Y. 2014. Spoken language understanding using long short-term memory neural networks. In *Proc. of SLT*.

Yao, K.; Zweig, G.; Hwang, M.-Y.; Shi, Y.; and Yu, D. 2013. Recurrent neural networks for language understanding. In *Proc. of INTERSPEECH*.

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

Zhang, C.; Li, Y.; Du, N.; Fan, W.; and Yu, P. S. 2019. Joint slot filling and intent detection via capsule neural networks. In *Proc. of ACL*.

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2018. Deep mutual learning. In *Proc. of CVPR*.

Zhu, S.; Cao, R.; and Yu, K. 2020. Dual Learning for Semi-Supervised Natural Language Understanding. *IEEE ACM Trans. Audio Speech Lang. Process.* .

Zhu, S.; and Yu, K. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *Proc. of ICASSP*.