Deep Cascade Multi-task Learning for Slot Filling in Online Shopping Assistant

Yu Gong[1*†], **Xusheng Luo**[1*], **Yu Zhu**[1], **Wenwu Ou**[1], **Zhao Li**[1], **Muhua Zhu**[1],
Kenny Q. Zhu[2], **Lu Duan**[3], **Xi Chen**[1]
[1] Search Algorithm Team, Alibaba Group
[2] Shanghai Jiao Tong University
[3] Artificial Intelligence Department, Zhejiang Cainiao Supply Chain Management Co.
{gongyu.gy, lxs140564, zy143829, lizhao.lz, muhua.zmh}@alibaba-inc.com,
{santong.oww, gongda.cx}@taobao.com, kzhu@cs.sjtu.edu.cn, duanlu.dl@cainiao.com

Abstract

Slot filling is a critical task in natural language understanding (NLU) for dialog systems. State-of-the-art approaches treat it as a sequence labeling problem and adopt such models as BiLSTM-CRF. While these models work relatively well on standard benchmark datasets, they face challenges in the context of E-commerce where the slot labels are more informative and carry richer expressions. In this work, inspired by the unique structure of E-commerce knowledge base, we propose a novel multi-task model with cascade and residual connections, which jointly learns segment tagging, named entity tagging and slot filling. Experiments show the effectiveness of the proposed cascade and residual structures. Our model has a 14.6% advantage in F1 score over the strong baseline methods on a new Chinese E-commerce shopping assistant dataset, while achieving competitive accuracies on a standard dataset. Furthermore, online test deployed on such dominant E-commerce platform shows 130% improvement on accuracy of understanding user utterances. Our model has already gone into production in the E-commerce platform.

1 Introduction

An intelligent online shopping assistant offers services such as pre-sale and after-sale inquiries, product recommendations, and user complaints processing, all of which seek to give the customers better shopping experience. The core of such assistant is a task-oriented dialog system which has the ability to understand natural language utterances from a user and then give natural language responses (Yan et al. 2017). Natural Language Understanding (NLU), which aims to interpret the semantic meanings conveyed by input utterances, is a main component in task-oriented dialog systems. Slot filling, a sub-problem of NLU, extracts semantic constituents by using the words of input utterance to fill in predefined slots in a semantic frame (Mesnil et al. 2015).

In the case of E-commerce shopping, there are three named entity types: *Category*, *Property Key* and *Property Value*, according to typical E-commerce knowledge base such as the one in Figure 1. We show a real example in Table 1 with In/Out/Begin (**I/O/B**) scheme. In the named entity level, "dress" is a Category (**CG**), while "brand" is labeled as Property Key (**PK**), which is the name of one product property. "Nike" and "black" are labeled as Property Value (**PV**) since they are concrete property values. However, merely labeling as Property Value is not sufficient as the shopping assistant needs more fine-grained semantics. Therefore, in the Slot Filling level, we further label "Nike" as Brand Property (**Brand**), and "black" as Color Property (**Color**). In Table 1, **B-CG** refers to Begin-Category (the meaning of other labels can also be inferred). In the meantime, other words in the example utterance that carry no semantic meaning are assigned **O** label.

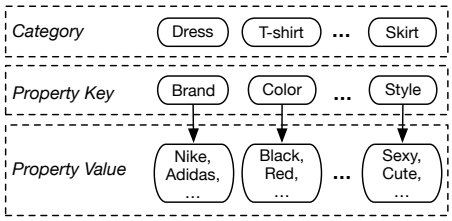


Figure 1: Structure of E-commerce knowledge-base.

Traditionally, slot filling problem can be regarded as a sequence labeling task, which assigns an appropriate semantic label to each word in the given input utterance. State-of-the-art sequence labeling models are typically based on BiLSTM-CRF (Huang, Xu, and Yu 2015; Reimers and Gurevych 2017) and evaluated on a commonly used standard dataset ATIS (Price 1990) in the slot filling area. This dataset is about airline travel in the United States. However, the vocabulary size of ATIS is small (only 572) and slot labels are not diverse enough (mostly related to only time and location) since airline travel is a relatively small and specific domain, such that recent deep learning models can achieve very high F1 scores (nearly 0.96). Recently, a detailed quantitative and qualitative study of this dataset comes to the same conclusion that slot filling models should be tested on a much more real and complex dataset (Béchet and Raymond 2018).

In this paper, we try to tackle a real-world slot filling problem for one of the largest E-commerce platform in China. The semantic slots are much more diverse and informative than ATIS. For example, to describe different properties of a

[*]Equal contribution.
[†]Corresponding author.

Utterance	我	想	买	耐	克	品	牌	的	黑	色	连	衣	裙
	I	want	buy	Nike		brand		\	black		dress		
Slot Label	**O**	**O**	**O**	**B-Brand**	**I-Brand**	**B-PK**	**I-PK**	**O**	**B-Color**	**I-Color**	**B-CG**	**I-CG**	**I-CG**
Named Entity Label	**O**	**O**	**O**	**B-PV**	**I-PV**	**B-PK**	**I-PK**	**O**	**B-PV**	**I-PV**	**B-CG**	**I-CG**	**I-CG**
Segment Label	**O**	**O**	**O**	**B**	**I**	**B**	**I**	**O**	**B**	**I**	**B**	**I**	**I**

Table 1: A real example of slot filling in online shopping scenario.

product for the purpose of utterance understanding, we define large amount of informative slot labels such as color, brand, style, season, gender and so on. In contrast, most semantic labels of ATIS are related to only time and location. Furthermore, the Chinese language used for e-commerce is more complex and the semantically rich expressions make it harder to understand. Whereas in ATIS, expression can be simpler, and most expressions are standard locations or time. Thus, large scale semantic slots and more complex expressions bring problem such as data sparsity. Traditional end-to-end sequence labeling model may not be able to handle it.

Besides, Chinese language, like many other Asian languages, is not word segmented by nature, and word segmentation is a difficult first step in many NLP tasks. Without proper word segmentation, sequence labeling becomes very challenging as the errors from segmentation will propagate. On the other hand, more than 97% of the chunks in ATIS data have only one or two words, in which segmentation (or chunking) is not a serious problem. Due to these reasons, if we simply apply basic sequence labeling models, which can be regarded as an end-to-end method, the sentences may not be segmented correctly in the first place. Then the errors will propagate and the resulting slot labels will be incorrect.

In this paper, we propose to employ multi-task sequence labeling model to tackle slot filling in a novel Chinese E-commerce dialog system. Inspired by the natural structure of E-commerce knowledge base shown in Figure 1, we extract two additional lower-level tasks from the slot filling task: *named entity tagging* and *segment tagging*. Example labels of these two tasks are shown in the bottom two rows of Table 1. Segment tagging and named entity tagging can be regarded as syntactic labeling, while slot filling is more like semantic labeling. With the help of information sharing ability of multi-task learning, once we learn the information of syntactic structure of an input sentence, filling the semantic labels becomes much easier. Compared to directly attacking slot filling, these two low-level tasks are much easier to solve due to fewer labels. To this end, we propose a Deep Cascade Multi-task Learning model, and co-train three tasks in the same framework with a goal of optimizing the target slot filling task.

The contributions of this paper are summarized below:

- To the best of our knowledge, this is the first piece of work focusing on slot filling in E-commerce. We propose a novel deep multi-task sequence labeling model (DCMTL) with cascading and residual connection to solve it (Section 2.3).
- We develop a Chinese E-commerce shopping assistant dataset ECSA (Section 3.1), which is much bigger and different from the common ATIS dataset, and would be a valuable contribution to dialog system research.
- We evaluate DCMTL in both offline and online settings. Offline results show the model outperforms several strong baseline methods by a substantial margin of 14.6% on $F1$ score (Section 3.3). Online testing deployed on the mentioned E-commerce platform shows that slot filling results returned by our model achieve 130% improvement on accuracy which significantly benefits to the understanding of users' utterances (Section 3.4). Our model has already gone production in the platform.

2 Approach

In this section we describe our approach in detail. Figure 2 gives an overview of the proposed architectures. First we introduce the most common and popular BiLSTM-CRF model (Figure 2(a)) for sequence labeling tasks. Then we move on to multi-task learning perspective (Figure 2(b) and (c)). Finally we propose our new method, which is called Deep Cascade Multi-task Learning in Figure 2(d).

Given an utterance containing a sequence of words $\mathbf{w} = (w_1, w_2, ..., w_T)$, the goal of our problem is to find a sequence of slot labels $\hat{\mathbf{y}} = (y_1, y_2, ..., y_T)$, one for each word in the utterance, such that:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{w}).$$

We use "word" in problem and model description, but "word" actually means Chinese char in our problem. And a "term" consists of one or several words.

2.1 RNN Sequence Labeling

Figure 2(a) shows the principle architecture of a BiLSTM-CRF model, which is the state-of-the-art model for various sequence labeling tasks (Huang, Xu, and Yu 2015; Reimers and Gurevych 2017). BiLSTM-CRF model consists of a BiLSTM layer and a CRF layer.

BiLSTM (Bidirectional-LSTM) enables the hidden states to capture both historical and future context information of the words. Mathematically, the input of this BiLSTM layer is a sequence of input vectors, denoted as $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)$. The output of BiLSTM layer is a sequence of the hidden states for each input word, denoted as $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T)$. Each final hidden state is the concatenation of the forward $\overrightarrow{\boldsymbol{h}_i}$ and backward $\overleftarrow{\boldsymbol{h}_i}$ hidden states. We view BiLSTM as a

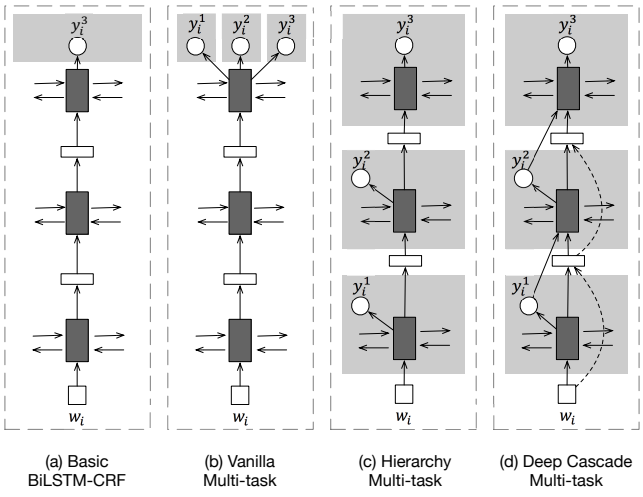


Figure 2: Sequential models for slot filling task.

function $\mathrm{BiLSTM}(\boldsymbol{x}_i)$:

$$\overrightarrow{\boldsymbol{h}_i} = \mathrm{LSTM}(\boldsymbol{x}_i, \overrightarrow{\boldsymbol{h}_{i-1}}), \overleftarrow{\boldsymbol{h}_i} = \mathrm{LSTM}(\boldsymbol{x}_i, \overleftarrow{\boldsymbol{h}_{i+1}}),$$
$$\mathrm{BiLSTM}(\boldsymbol{x}_i) = \boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}_i}(\boldsymbol{x}_i); \overleftarrow{\boldsymbol{h}_i}(\boldsymbol{x}_i)].$$

Most of time we stack multiple BiLSTMs to make the model deeper, in which the output $\boldsymbol{h}_i^l$ of layer l becomes the input of layer $l+1$, e.g. $\boldsymbol{h}_i^{l+1} = \mathrm{BiLSTM}^{l+1}(\boldsymbol{h}_i^l)$.

It is always beneficial to consider the correlations between the current label and neighboring labels, since there are many syntactical constraints in natural language sentences. For example, **I-Brand** is never followed by a **B-Color**. If we simply feed the above mentioned hidden states independently to a softmax layer to predict the labels (Hakkani-Tür et al. 2016), such constraints are more likely to be violated. Linear-chain Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001) is the most popular way to control the structure prediction and its basic idea is to use a series of potential functions to approximate the conditional probability of the output label sequence given the input word sequence.

Formally, we take the above sequence of hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T)$ as input to a CRF layer, and the output of the CRF is the final prediction label sequence $\boldsymbol{y} = (y_1, y_2, ..., y_T)$, where y_i is in the set of pre-defined target labels. We denote $\mathcal{Y}(\boldsymbol{H})$ as the set of all possible label sequences. Then we derive the conditional probability of the output sequence, given the input hidden state sequence is:

$$p(\boldsymbol{y}|\boldsymbol{H}; \boldsymbol{W}, \boldsymbol{b}) = \frac{\prod_{i=1}^{T} \varphi(y_{i-1}, y_i, \boldsymbol{H})}{\sum_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{H})} \prod_{i=1}^{T} \varphi(y'_{i-1}, y'_i, \boldsymbol{H})},$$

where $\varphi(y', y, \boldsymbol{H}) = \exp(\boldsymbol{W}_{y',y}^T \boldsymbol{H} + \boldsymbol{b}_{y',y})$ are potential functions and $\boldsymbol{W}_{y',y}^T$ and $\boldsymbol{b}_{y',y}$ are weight vector and bias of label pair (y', y). To train the CRF layer, we use the classic maximum conditional likelihood estimate and gradient ascent. For a training dataset $\{(\boldsymbol{H}_{(i)}, \boldsymbol{y}_{(i)})\}$, the final log-likelihood is:

$$L(\boldsymbol{W}, \boldsymbol{b}) = \sum_i \log p(\boldsymbol{y}_{(i)}|\boldsymbol{H}_{(i)}; \boldsymbol{W}, \boldsymbol{b}).$$

Finally, the Viterbi algorithm is adopted to decode the optimal output sequence $\boldsymbol{y}^*$:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{H})} p(\boldsymbol{y}|\boldsymbol{H}; \boldsymbol{W}, \boldsymbol{b}).$$

2.2 Multi-task Learning

The slot labels are large-scaled, informative and diverse in the case of E-commerce, and the syntactic structure of input Chinese utterance are complicated, so that the slot filling problem becomes hard to solve. If we directly train an end-to-end sequential model, the tagging performance will suffer from data sparsity severely. When we try to handle slot filling (can be seen as semantic labeling task), some low-level tasks such as named entity tagging or segment tagging (can be seen as syntactic labeling task) may first make mistakes. If the low-level tasks get wrong, so as to the target slot filling task. That is to say it is easy to make wrong decisions in the low-level tasks, if we try to fill in all the labels at once. Then the error will propagate and lead to a bad performance of slot filling, which is our high-level target.

While directly attacking the slot filling task is hard, low-level tasks with fewer labels are much easier to solve. Once we know the syntactic structure of a sentence, filling in semantic labels will become easier accordingly. Thus, it is reasonable to solve the problem in a multi-task learning framework. In our problem, following the special structure of E-commerce knowledge base (Figure 1), we can devise three individual tasks: slot filling, named entity tagging and segment tagging. Slot filling is our target task; named entity tagging is to classify which named entity type (**PV/PK/CG**) a word is; and segment tagging is to judge whether a word is begin (**B**), in (**I**) or out (**O**) of a trunking.

In a multi-task learning (MTL) setting, we have several prediction tasks over the same input sequence, where each task has its own output vocabulary (a set of task specified labels). Intuitively, the three tasks do share a lot of information. Consider the example in Table 1 again. Knowing the named entity type of " 黑|色 " being **B-PV**|**I-PV** can definitely help determine its slot label, which is **B-Color**|**I-Color**. Similarly, knowing its segment type (**B**|**I**) also helps with both named entity tagging and slot filling. Thus it is reasonable for these tasks to share parameters and learn in the same framework cooperatively.

Vanilla Multi-task Learning The general idea of multi-task learning is to share parameters of encoding part of the network. As Figure 2(b) shows, this is naturally achieved by sharing the k-layers BiLSTM part of the network across three tasks. Based on that, we use a separate CRF decoder for each task $t \in \{seg, ne, slot\}$: $p(\boldsymbol{y}^t|\boldsymbol{H}^k; \boldsymbol{W}_t, \boldsymbol{b}_t)$, where $\boldsymbol{W}_t$ and $\boldsymbol{b}_t$ are task-specific parameters. This encourages the deep BiLSTM network to learn a hidden representation $\boldsymbol{H}^k$ which benefits all three different tasks.

Hierarchy Multi-task Learning Previous discussion indicates that there is a natural order among the different tasks: slot filling may benefit more from named entity tagging, than the other way around. This motivates us to employ low-level tasks at lower BiLSTM layers, while high level tasks are trained at higher layers. We borrow the idea of involving a hierarchical neural networks structure (Peters et al. 2018; Søgaard and Goldberg 2016). As shown in Figure 2(c), instead of decoding all tasks separately at the outermost BiLSTM layer, we associate each BiLSTM layer $l(t)$ with one task t. Then the conditional probabilities of the output sequence for each task are:

$$\begin{aligned}
\text{seg_tag}(\boldsymbol{w}) &= p(\boldsymbol{y}^{seg}|\boldsymbol{H}^{l(seg)}; \boldsymbol{W}_{seg}, \boldsymbol{b}_{seg}),\\
\boldsymbol{H}^{l(seg)} &= \text{BiLSTM}^{l(seg)}(E(\boldsymbol{w})).\\
\text{ne_tag}(\boldsymbol{w}) &= p(\boldsymbol{y}^{ne}|\boldsymbol{H}^{l(ne)}; \boldsymbol{W}_{ne}, \boldsymbol{b}_{ne}),\\
\boldsymbol{H}^{l(ne)} &= \text{BiLSTM}^{l(ne)}(\boldsymbol{H}^{l(seg)}).\\
\text{slot_fill}(\boldsymbol{w}) &= p(\boldsymbol{y}^{slot}|\boldsymbol{H}^{l(slot)}; \boldsymbol{W}_{slot}, \boldsymbol{b}_{slot}),\\
\boldsymbol{H}^{l(slot)} &= \text{BiLSTM}^{l(slot)}(\boldsymbol{H}^{l(ne)}).
\end{aligned}$$

Here seg_tag, ne_tag and slot_fill represent the tasks of segment tagging, named entity tagging and slot filling, respectively. $E(\boldsymbol{w})$ is the word embeddings of input sequence $\boldsymbol{w}$ and $l(seg) < l(ne) < l(slot)$. We call this model hierarchy multi-task learning, since some layers are shared by all tasks while the others are only related to specific tasks.

2.3 Deep Cascade Multi-task Learning

Hierarchy multi-task learning share parameters among different tasks, and allow low-level tasks help adjust the result of high-level target task. It is effective for those tasks which are weakly correlated, such as POS tagging, syntactic chunking and CCG supertagging (Søgaard and Goldberg 2016). However, when it comes to problems where different tasks maintain a strict order, in another word, the performance of high-level task dramatically depends on low-level tasks, the hierarchy structure is not compact and effective enough. Therefore, we propose *cascade* and *residual* connections to allow high-level tasks to take the tagging results and hidden states from low-level tasks as additional input. These connections serves as "shortcuts" that create a more closely coupled and efficient model. We call it deep cascade multi-task learning, and the framework is shown in Figure 2(d).

Cascade Connection Here we feed the tagging output of the task at lower layer e.g. $\text{seg\_tag}^*(\boldsymbol{w})$ or $\text{ne\_tag}^*(\boldsymbol{w})$ to the upper BiLSTM layer as its additional input. Now the hidden states of each task layer become:

$$\begin{aligned}
\boldsymbol{H}^{l(seg)} &= \text{BiLSTM}^{l(seg)}(E(\boldsymbol{w})),\\
\boldsymbol{H}^{l(ne)} &= \text{BiLSTM}^{l(ne)}(\boldsymbol{W}^{seg}_{Cas.} \cdot \text{seg_tag}^*(\boldsymbol{w}) + \boldsymbol{H}^{l(seg)}),\\
\boldsymbol{H}^{l(slot)} &= \text{BiLSTM}^{l(slot)}(\boldsymbol{W}^{ne}_{Cas.} \cdot \text{ne_tag}^*(\boldsymbol{w}) + \boldsymbol{H}^{l(ne)}),
\end{aligned}$$

where $\boldsymbol{W}^{seg}_{Cas.}$ and $\boldsymbol{W}^{ne}_{Cas.}$ are the weight parameters for cascade connection.

At training time, $\text{seg\_tag}^*(\boldsymbol{w})$ and $\text{ne\_tag}^*(\boldsymbol{w})$ can be the true tagging outputs. At inference time, we simply take the greedy path of our cascade model without doing search, where the model emits the best $\text{seg\_tag}^*(\boldsymbol{w})$ and $\text{ne\_tag}^*(\boldsymbol{w})$ by Viterbi inference algorithm. Alternatively, one can do beam search (Sutskever, Vinyals, and Le 2014; Vinyals et al. 2015) by maintaining a set of k best partial hypotheses at each cascade layer. However, unlike traditional seq2seq models e.g., in machine translation, where each inference step is just based on probability of a discrete variable (by softmax function), our inference for tagging output is a structured probability distribution defined by the CRF output. Efficient beam search method for this structured cascade model is left to our future work.

Residual Connection To encourage the information sharing among different tasks, we also introduce the residual connection, where we add the input of a previous layer to the current input:

$$\begin{aligned}
\boldsymbol{H}^{l(seg)} &= \text{BiLSTM}^{l(seg)}(\boldsymbol{x}^{seg}),\\
\boldsymbol{x}^{l(seg)} &= E(\boldsymbol{w}).\\
\boldsymbol{H}^{l(ne)} &= \text{BiLSTM}^{l(ne)}(\boldsymbol{W}^{seg}_{Cas.} \cdot \text{seg_tag}^*(\boldsymbol{w}) + \boldsymbol{x}^{l(ne)}),\\
\boldsymbol{x}^{l(ne)} &= \boldsymbol{H}^{l(seg)} + \boldsymbol{x}^{l(seg)}.\\
\boldsymbol{H}^{l(slot)} &= \text{BiLSTM}^{l(slot)}(\boldsymbol{W}^{ne}_{Cas.} \cdot \text{ne_tag}^*(\boldsymbol{w}) + \boldsymbol{x}^{l(slot)}),\\
\boldsymbol{x}^{l(slot)} &= \boldsymbol{H}^{l(ne)} + \boldsymbol{x}^{l(ne)}.
\end{aligned}$$

Deep residual learning (He et al. 2016) is introduced to ease the gradient vanish problem for training very deep neural networks. Here we borrow the idea of cross residual learning method for multi-task visual recognition(Jou and Chang 2016) and believe the residual connection between different layers can benefit our multi-task sequence learning. We propose *cascade* residual connection instead of *cross* residual connection because different tasks are connected via cascading in our problem, while they are organized via branching in visual recognition.

2.4 Training

For our multi-task setting, we define three loss functions (refer to Section 2.1): L_{seg}, L_{ne} and L_{slot} for tasks of segment tagging, named entity tagging and slot filling respectively. We construct three training set, D_{seg}, D_{ne} and D_{slot}, where each of them (called D_t generically) contains a set of input-output sequence pair $(\boldsymbol{w}, \boldsymbol{y}^t)$. The input utterance $\boldsymbol{w}$ is shared across tasks, but the output $\boldsymbol{y}^t$ is task dependent.

For vanilla multi-task learning, we define a unified loss function $L = \alpha L_{seg} + \beta L_{ner} + (1 - \alpha - \beta) L_{slot}$, where α and β are hyper-parameters. And we update the model parameters by loss L.

As for hierarchy multi-task learning and cascade multi-task learning, we choose a random task $t \in \{seg, ne, slot\}$ at each training step, followed by a random training batch $\text{Batch}(\boldsymbol{w}, \boldsymbol{y}^t) \in D_t$. Then we update the model parameters by back-propagating the corresponding loss L_t.

3 Experiments

In this section we first introduce the popular ATIS dataset[1], then describe how we collect our **E**-**c**ommerce **S**hopping **A**ssistant (ECSA) dataset[2]. Then we show the implementation details for our model. Finally we demonstrate the evaluation results on both ATIS and ECSA dataset and give some discussions. In the following experiments, we call our proposed **D**eep **C**ascade **M**ulti-**T**ask **L**earning method as DCMTL for short.

3.1 Dataset

ATIS Dataset The ATIS corpus, the most commonly used dataset for slot filling research, contains reservation requests for air travel. It contains 4,978 training and 893 testing sentences in total, with a vocabulary size of 572 (Mesnil et al. 2015). Apart from the ground-truth slot labels, we also generate its corresponding segment labels for our multi-task model setting.

ECSA Dataset To create large amounts of gold standard data to train our model, we adopt an unsupervised method to automatically tag the input utterances. All the utterances are extracted from the user input logs (either from text or voice) on our online shopping assistant system. Besides our E-commerce knowledge-base is a dictionary consisting of pairs of word terms and their ground-truth slot labels such as "red-**Color**" or "Nike-**Brand**". Since this resource is created by human beings, we will use it to create gold standard. We use a dynamic programming algorithm of max-matching to match words in the utterances and then assign each word with its slot label in IOB scheme. We filter utterances whose matching result is ambiguous and only reserve those that can be perfectly matched (all words can be tagged by only one unique label) as our training and testing data. With the slot labels of each word, we can induce the named entity labels and segment labels straightforwardly via the E-commerce knowledge-base. For we only extract the perfectly matched sentences, the quality of our ECSA dataset can be guaranteed. It can be considered as a long-distance supervision method (**?**).

To evaluate model's ability to generalize, we randomly split the dictionary into three parts. One part is used to generate testing data and the other two to generate training data. If we don't split the dictionary and use the whole to generate both training and testing data, then the trained model may remember the whole dictionary and the results will not reflect the true performance of the models.

This unsupervised approach alleviates human annotations, and we can produce a large volume of labeled data automatically. The following experiments use a dataset of 24,892 training pairs and 2,723 testing pairs. Each pair contains an input utterance $\boldsymbol{w}$, its corresponding gold sequence of slot labels $\boldsymbol{y}^{slot}$, named entity labels $\boldsymbol{y}^{ne}$ and segment labels $\boldsymbol{y}^{seg}$. The vocabulary size of ECSA is 1265 (Chinese characters), and the amount of segmented terms can be much larger. The Out-of-Vocabulary (OOV) rate of ESCA dataset is 85.3% (Meaning 85.3% of terms in testing data never appear in training data) while the OOV rate of ATIS is lower than 1%. Apparently slot filling task on ESCA dataset is more challenging.

[1] https://github.com/yvchen/JointSLU/tree/master/data

[2] https://github.com/pangolulu/DCMTL

3.2 Implementation Details

For the RNN component in our system, we use a 3-layers BiLSTM networks for ECSA and 2-layers BiLSTM networks for ATIS (no named entity tagging in this case), and all LSTM networks come with hidden state size 100. The input in ECSA is a sequence of Chinese characters rather that words since there is no segmentation. The dimension of embedding layer E and BiLSTM network output state (concatenation of the forward and backward LSTM) are set to 200. We perform a mini-batch log-likelihood loss training with a batch size of 32 sentences for 10 training epochs. We use Adam optimizer, and the learning rate is initialized to 0.001. To prevent the gradient explosion problem for training LSTM networks, we set gradient clip-norm as 5.

3.3 Results and Discussions

Evaluation on ATIS We compare the ATIS results of our DCMTL model with current published results in Table 2. We split the methods into two categories: one is *Sequence Labeling* based method, and the other is *Encoder-Decoder* based method. Sequence Labeling based method generally adopts a sequential network (RNN (Yao et al. 2013; 2014; Liu and Lane 2015; Peng and Yao 2015; Vu et al. 2016) or CNN (Xu and Sarikaya 2013; Vu 2016)) and calculate a loss function (such as CRF loss (Xu and Sarikaya 2013), cross entropy loss (Yao et al. 2013; 2014) or ranking loss (Vu et al. 2016)) on top of the network output. Encoder-Decoder based method, on the other hand, usually employs a RNN to encode the whole sentence and another RNN to decode the labels (Kurata et al. 2016). The decoder will attend to the whole encoding sequence with attention mechanism (Zhu and Yu 2017; Zhai et al. 2017). Our method follows the Sequence Labeling framework and we design a novel multi-task sequence labeling model which achieve the best performance against the published Sequence Labeling based method (F1+0.22%) and compatible result against the best Encoder-Decoder based method (F1-0.03%). As we claim in Section 1, more than 97% of chunks in ATIS dataset have only one or two words and there are no named entity labels at all. These two reasons prevent our proposed DCMTL model from further improving the performance on ATIS dataset. Thus, we will mainly focus on ECSA dataset, which is much larger and more sophisticated, to prove the effectiveness of our proposed model.

Besides, almost all the methods (including ours) reach very high F1 score of around 0.96. This also makes us wonder whether it is meaningful enough to continue evaluating on this dataset, for minor differences in the results may be attributed to data variance more than the models. Apparently high performance on ATIS does not mean working on real-world application which contains more informative semantic slot labels and more complicated expressions as in the case of online shopping assistant.

Methods	F1
simple RNN (Yao et al. 2013)	0.9411
CNN-CRF (Xu and Sarikaya 2013)	0.9435
LSTM (Yao et al. 2014)	0.9485
RNN-SOP (Liu and Lane 2015)	0.9489
Deep LSTM (Yao et al. 2014)	0.9508
RNN-EM (Peng and Yao 2015)	0.9525
Bi-RNN with ranking loss (Vu et al. 2016)	0.9547
Sequential CNN (Vu 2016)	0.9561
Encoder-labeler Deep LSTM (Kurata et al. 2016)	0.9566
BiLSTM-LSTM (focus) (Zhu and Yu 2017)	0.9579
Neural Sequence Chunking (Zhai et al. 2017)	**0.9586**
DCMTL (Ours)	0.9583*

Table 2: Comparison with published results on the ATIS dataset.

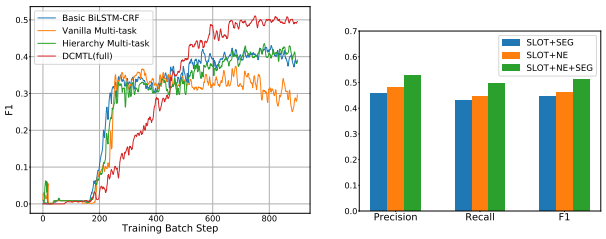
Models	Precision	Recall	F1
Basic BiLSTM-CRF	0.4330	0.4275	0.4302
* Basic BiLSTM-CRF (cond. SEG)	0.7948	0.7953	0.7950
* Basic BiLSTM-CRF (cond. NE)	0.8985	0.8986	0.8985
Vanilla Multi-task	0.3990	0.3941	0.3965
Hierarchy Multi-task	0.4417	0.4494	0.4455
** DCMTL (- cascade)	0.4654	0.4613	0.4633
** DCMTL (- residual)	0.4923	0.4760	0.4840
DCMTL (full)	**0.5281**	**0.4941**	**0.5105**

Table 3: Results for slot filling task on the ECSA dataset. Columns with highlighted boldface are the best performance. Rows with * prefix are just results for our case study. Rows with ** prefix are results for ablation test.

Evaluation on ECSA On ECSA dataset, we evaluate different models including Basic BiLSTM-CRF, Vanilla Multi-task, Hierarchy Multi-task and Deep Cascade Multi-task on testing data regarding slot filling as the target task. We report Precision, Recall and F1 in Table 3.

The Basic BiLSTM-CRF model achieves an F1 score of 0.43. To show the usefulness of the lower tasks to slot filling, we "cheated" by using the ground-truth segment type (cond. SEG) or named entity type (cond. NE) as the extra features for each word in the Basic BiLSTM-CRF model. Row 3 and 4 (with *) in Table 3 show that the slot filling performance can be improved by 85% and 109% if the correct segment type or named entity type is pre-known. It can perfectly verify our claim that low-level syntactic tasks can significantly affect to the slot filling performance. Of course in practice, the model doesn't know the true values of these types during prediction.

Our further experiments show that DCMTL outperforms the baselines on both precision and recall. DCMTL achieves the best F1 score of 0.5105, which improves by a relative margin of 14.6% against the strong baseline method (see Table 3). Multi-task models generally perform better than the Basic BiLSTM with single-task target. The exception is the vanilla multi-task setting. This is mainly because vanilla multi-task shares parameters across all the layers, and these parameters are likely to be disturbed by the interaction of three tasks. It is more desirable to let the target task dominate the weights at high-level layers.



(a) (b)

Figure 3: (a) Learning trends of F1 respectively for different methods. (b) Result of different cascade connection types in DCMTL.

We further investigate the learning trend of our proposed approach against baseline methods. Figure 3(a) shows the typical learning curves of performance measured by F1. We can observe that our method DCMTL performs worse than other baseline methods for the first 450 batch steps. After that, other methods converge quickly and DCMTL perform much better after 500 batch steps and finally converge to the best F1 score. We believe that in the beginning, high-level task in DCMTL is affected more by the noise of low-level tasks comparing to others, but as the training goes on, the high-level slot filling task slowly reaps the benefits from low-level tasks.

To make our experiments more solid, we implemented two previous best performing models on ATIS dataset: Sequential CNN (Vu 2016) (Sequence Labeling based) and Neural Sequence Chunking (Zhai et al. 2017) (Encoder-Decoder based). They achieved 0.2877 and 0.4355 F1 scores respectively, while our DCMTL model scores 0.5105 F1 and outperforms both of them (by **77%** and **17%** improvements).

Ablation Test Our "shortcuts" connections come in two flavors: cascade connection and residual connection. Multi-task outputs and "shortcuts" connections are highly related since without the multi-task framework, there will be no cascade connections. We go on to show that both multi-task setting and the "shortcuts" connections are effective and useful in Table 3, where F1 score improves from 0.4302 to 0.4455 and 0.5105 respectively. We also investigate how our model DCMTL performs with or without cascade and residual connections (rows with ** prefix in Table 3). F1 score increases from 0.4840 to 0.5105 when residual connection is applied, which verifies its benefit. If we remove cascade connection from DCMTL, the model actually degenerates into hierarchy multi-task model with residual connection and performs 0.4633 F1 score. Thus we can conclude that both connections are helpful for our DCMTL model. However, the cascade connection, which relies on the multi-task, is more effective than the residual connection. We can verify it from the fact that DCMTL model without cascade connection performs much worse than without residual connection (0.4633

vs. 0.4840 F1 scores).

Furthermore, we explore how DCMTL performs with different cascade connection methods. We compare three different types of cascade connection illustrated in Figure 4(a):

1. Segment labeling skipped to slot filling (SLOT+SEG).
2. Named entity labeling directly connected to slot filling (SLOT+NE).
3. Segment labeling, named entity labeling and slot filling in sequence (SLOT+NE+SEG).

From Figure 3(b), we find that cascade connection with type 3 performs the best and then with type 2, while cascade method with skipped connection (type 1) performs the worst. Therefore, we design the networks with a cascade connection in a hierarchical fashion and do not apply skipped connection for the cascade inputs (Figure 4(b)). This phenomenon here may also be proved by our "cheated" case study above. Slot filling performance with pre-known named entity type is much better than with pre-known segment type (rows with * in Table 3).

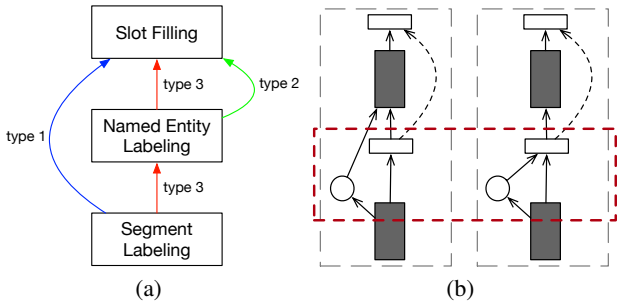


Figure 4: (a) Three types of cascade connection in our experiment. (b) Comparison between hierarchical and skipped cascade connection.

3.4 Online Testing

Previous experimental results have proven the advantages of our proposed DCMTL approach, so we deploy it in a real world online environment to test its practical performance.

For online A/B testing, we extracted users query log with the slot filling results for one day. There are in total 251,409 unique queries. We let three persons to manually evaluate whether a query is slotted perfectly with the strategy where the minority obeys the majority. A query is slotted perfectly means all terms in query are assigned with the correct slot labels. Our DCMTL model results in 152,178 perfectly slotted queries which is **60.53%** accuracy[3]. While the original online max-matching algorithm with E-commerce knowledge base[4] (more details in Section 3.1) only covers 66,302 perfectly slotted queries with **26.37%** accuracy. Thus, the accuracy of query slot filling in such online shopping assistant system is improved by 130% after deploying DCMTL model. This demonstrates that our model can effectively extract the semantic attributes of users query which is extremely helpful E-commerce Shopping Assistant system.

[3]We only report accuracy as evaluation metric, because precision and recall are the same in such case.

[4]As we have showed that our DCMTL model outperforms several strong baselines in the offline evaluation, and the gap between online and offline is minor since our offline dataset also comes from online queries, we only deploy DCMTL model online since such evaluation is costly.

4 Related Work

Slot Filling is considered a sequence labeling problem that is traditionally solved by generative models. In recent years, deep learning approaches have been explored due to its successful application in many NLP tasks. Many neural network architectures have been used such as simple RNNs (Yao et al. 2013; Mesnil et al. 2015), convolutional neural networks (CNNs) (Xu and Sarikaya 2013), LSTMs (Yao et al. 2014) and variations like encoder-decoder (Zhu and Yu 2017; Zhai et al. 2017) and external memory (Peng and Yao 2015). In general, these works adopt a BiLSTM as the major labeling architecture to extract various features, then use a CRF layer (Huang, Xu, and Yu 2015) to model the label dependency. We also adopt a BiLSTM-CRF model as baseline and claim that a multi-task learning framework is working better than directly applying it on Chinese E-commerce dataset. Previous works only apply joint model of slot filling and intent detection (Zhang and Wang 2016; Liu and Lane 2016). Our work is the first to propose a multi-task sequence labeling model with novel cascade and residual connections based on deep neural networks to tackle real-world slot filling problem.

Multi-task Learning (MTL) has attracted increasing attention in both academia and industry recently. By jointly learning across multiple tasks (Caruana 1998), we can improve performance on each task and reduce the need for labeled data. There has been several attempts of using multi-task learning on sequence labeling task (Peng and Dredze 2016b; 2016a; Yang, Salakhutdinov, and Cohen 2017), where most of these works learn all tasks at the outmost layer. Sgaard and Goldberg (2016) is the first to assume the existence of a hierarchy between the different tasks in a stacking BiRNN model. Compared to these works, our DCMTL model further improves this idea even thorough with cascade and residual connection.

5 Conclusion

In this paper, we tackle the real-world slot filling task in a novel Chinese online shopping assistant system. We proposed a deep multi-task sequence learning framework with cascade and residual connection. Our model achieves comparable results with several state-of-the-art models on the common slot filling dataset ATIS. On our real-world Chinese E-commerce dataset ECSA, our proposed model DCMTL also achieves best F1 score comparing to several strong baselines. DCMTL has been deployed on the online shopping assistant of a dominant Chinese E-commerce platform. Online testing results show that our model meets better understanding of users utterances and improves customers shopping experience. Our future research may include a joint model for category classification and slot filling.

References

Béchet, F., and Raymond, C. 2018. Is atis too shallow to go deeper for benchmarking spoken language understanding models? In *InterSpeech*.

Caruana, R. 1998. Multitask learning. In *Learning to learn*. Springer.

Hakkani-Tür, D.; Tur, G.; Celikyilmaz, A.; Chen, Y.-N.; Gao, J.; Deng, L.; and Wang, Y.-Y. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv*.

Jou, B., and Chang, S.-F. 2016. Deep cross residual learning for multitask visual recognition. In *ACM MM*.

Kurata, G.; Xiang, B.; Zhou, B.; and Yu, M. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv*.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Liu, B., and Lane, I. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *NIPS Workshop*.

Liu, B., and Lane, I. 2016. Joint online spoken language understanding and language modeling with recurrent neural networks. *arXiv*.

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. *TASLP*.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Peng, N., and Dredze, M. 2016a. Improving named entity recognition for chinese social media with word segmentation representation learning. In *ACL*.

Peng, N., and Dredze, M. 2016b. Multi-task multi-domain representation learning for sequence tagging. *arXiv*.

Peng, B., and Yao, K. 2015. Recurrent neural networks with external memory for language understanding. *arXiv*.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *NAACL*.

Price, P. J. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language*.

Reimers, N., and Gurevych, I. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv*.

Søgaard, A., and Goldberg, Y. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Vu, N. T.; Gupta, P.; Adel, H.; and Schütze, H. 2016. Bidirectional recurrent neural network with ranking loss for spoken language understanding. In *ICASSP*.

Vu, N. T. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. *arXiv*.

Xu, P., and Sarikaya, R. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *ASRU*.

Yan, Z.; Duan, N.; Chen, P.; Zhou, M.; Zhou, J.; and Li, Z. 2017. Building task-oriented dialogue systems for online shopping. In *AAAI*.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv*.

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

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 *SLT Workshop*.

Zhai, F.; Potdar, S.; Xiang, B.; and Zhou, B. 2017. Neural models for sequence chunking. In *AAAI*.

Zhang, X., and Wang, H. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*.

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