Leveraging Sentence-level Information with Encoder LSTM for Natural Language Understanding

Gakuto Kurata
IBM Research
gakuto@jp.ibm.com

Bing Xiang
IBM Watson
bingxia@us.ibm.com

Bowen Zhou
IBM Watson
zhou@us.ibm.com

Mo Yu
Harbin Institute of Technology
gflfof@gmail.com

Abstract

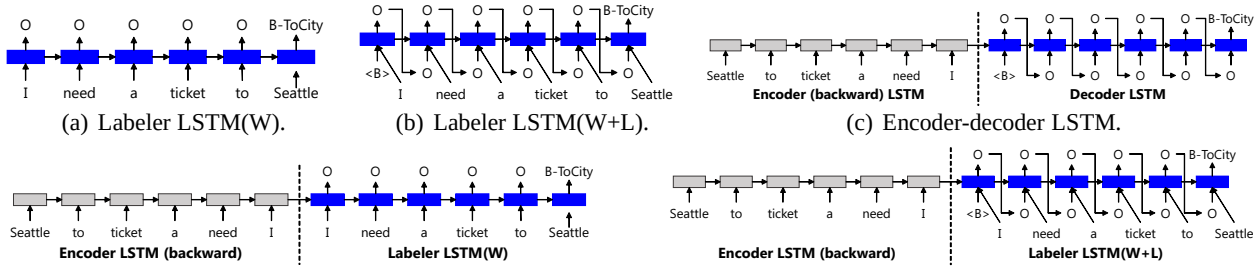
Recurrent Neural Network (RNN) and one of its specific architectures, Long Short-Term Memory (LSTM), have been widely used for sequence labeling. In this paper, we first enhance LSTM-based sequence labeling to explicitly model label dependencies. Then we propose another enhancement to incorporate the global information spanning over the whole input sequence. The latter proposed method, *encoder-labeler LSTM*, first encodes the whole input sequence into a fixed length vector with the encoder LSTM, and then uses this encoded vector as the initial state of another LSTM for sequence labeling. Combining these methods, we can predict the label sequence with considering label dependencies and information of whole input sequence. In the experiments of a slot filling task, which is an essential component of natural language understanding, with using the standard ATIS corpus, we achieved the state-of-the-art F_1-score of 95.66%.

1 Introduction

Natural language understanding (NLU) is an essential component of natural human computer interaction and typically consists of identifying the intent of the users (intent classification) and extracting the associated semantic slots (slot filling) (De Mori et al., 2008). We focus on the latter slot filling task in this paper.

Slot filling can be framed as a sequential labeling problem in which the most probable semantic slot labels are estimated for each word of the given word sequence. Slot filling is a traditional task and tremendous efforts have been done, especially since the 1980s when the Defense Advanced Research Program Agency (DARPA) Airline Travel Information System (ATIS) projects started (Price, 1990). Following the success of deep learning (Hinton et al., 2006; Bengio, 2009), Recurrent Neural Network (RNN) (Elman, 1990; Jordan, 1997) and one of its specific architectures, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), have been widely used since they can capture temporal dependencies (Yao et al., 2013; Yao et al., 2014a; Mesnil et al., 2015). The RNN/LSTM-based slot filling has been extended to be combined with explicit modeling of label dependencies (Yao et al., 2014b; Liu and Lane, 2015).

In this paper, we first enhance the LSTM-based slot filling to explicitly model label dependencies by feeding the output label of the previous time step to the hidden state of the current time step, as Mesnil et al. (2015) and Liu and Lane (2015) tried with RNN. Then we further extend the LSTM-based slot filling to consider sentence-level information. In the field of machine translation, an encoder-decoder LSTM has been gaining attention (Sutskever et al., 2014), where the encoder LSTM encodes the global information spanning over the whole input sentence in its last hidden state. Inspired by this idea, we propose an *encoder-labeler LSTM* that leverages the encoder LSTM for slot filling. First, we encode the input sentence into a fixed length vector by the encoder LSTM. Then, we predict the slot label sequence by the labeler LSTM whose hidden state is initialized with the encoded vector by the encoder LSTM. With this encoder-labeler LSTM, we can predict the label sequence while taking the sentence-level information into consideration. By combining explicit model-



(a) Labeler LSTM(W). (b) Labeler LSTM(W+L). (c) Encoder-decoder LSTM.

(d) Encoder-labeler LSTM(W). (e) Encoder-labeler LSTM(W+L).

Figure 1: Neural network architectures for slot filling. Input sentence is "I need a ticket to Seattle". "B-ToCity" is slot label for specific meaning and "O"is slot label without specific meaning. "<B>" is beginning symbol for slot sequence.

ing of label dependencies and the proposed encoder-labeler LSTM, slot filling with jointly considering label dependencies and sentence-level information becomes possible.

The main contributions of this paper are two-folds: (1) Proposed an encoder-labeler LSTM to leverage sentence-level information for natural language understanding. (2) Achieved the state-of-the-art F_1-score of 95.66% in the slot filling task of the standard ATIS corpus.

2 Proposed Method

We first revisit the LSTM for slot filling and enhance this to explicitly model label dependencies. Then we explain the proposed encoder-labeler LSTM.

2.1 LSTM for Slot Filling

LSTM is a specific RNN architecture and is easier to train thanks to its internal memory cells and gates. Figure 1(a) shows a typical LSTM for slot filling and we call this as *labeler LSTM(W)* where words are fed to the LSTM (Yao et al., 2014a).

Slot filling is a sequential labeling task to map a sequence of T words x_1^T to a sequence of T slot labels y_1^T. Each word x_t is represented with a V dimensional one-hot-vector where V is the vocabulary size and is transferred to d_e dimensional continuous space by the word embedding matrix $E \in \mathbb{R}^{d_e \times V}$ as Ex_t. Instead of simply feeding Ex_t into the LSTM, *Context Window* is a widely used technique to jointly consider k preceding and succeeding words as $Ex_{t-k}^{t+k} \in \mathbb{R}^{d_e(2k+1)}$. The LSTM has the architecture based on Jozefowicz et al. (2015) that does not have peephole connections and yields the hidden state sequence h_1^T. For each time step t, the posterior probabilities for each slot label are calculated by the softmax layer over the hidden state h_t. The word embedding matrix E, LSTM parameters, and softmax layer parameters are estimated to minimize the negative log likelihood over the correct label sequences with Back-Propagation Through Time (BPTT) (Williams and Peng, 1990).

2.2 Explicit Modeling of Label Dependency

A shortcoming of the labeler LSTM(W) is that it does not consider label dependencies. To explicitly model label dependencies, we introduce a new architecture, *labeler LSTM (W+L)*, as shown in Figure 1(b), where the output label of previous time step is fed to the hidden state of current time step, jointly with words, as Mesnil et al. (2015) and Liu and Lane (2015) tried with RNN. For model training, one-hot-vector of ground truth label of previous time step is fed to the hidden state of current time step and for evaluation, left-to-right beam search is used.

2.3 Encoder-labeler LSTM for Slot Filling

We propose two types of the encoder-labeler LSTM that uses the labeler LSTM(W) and the labeler LSTM(W+L). Figure 1(d) shows the *encoder-labeler LSTM(W)*. The encoder LSTM, to the left of the dotted line, reads through the input sentence backward. Its last hidden state contains the encoded information of the input sentence. The labeler LSTM(W), to the right of the dotted line, is the same with the labeler LSTM(W) explained in Section 2.1, *except that its hidden state is initialized with the last hidden state of the encoder LSTM*. The labeler LSTM(W) predicts the slot label conditioned on the encoded information by the encoder LSTM, which means that slot filling is conducted with taking sentence-level information into consideration. Figure 1(e) shows the *encoder-labeler LSTM(W+L)*, which uses the labeler LSTM(W+L) and predicts the slot label considering sentence-level information and label dependencies jointly.

Model training is basically the same as with the baseline labeler LSTM(W), as shown in Section 2.1, except that the error in the labeler LSTM is propa-

Sentence	show	flights	from	Boston	to	New	York	today
Slots	O	O	O	B-FromCity	O	B-ToCity	I-ToCity	B-Date

Figure 2: Example of ATIS sentence and annotated slots.

gated to the encoder LSTM with BPTT.

This encoder-labeler LSTM is motivated by the encoder-decoder LSTM that has been applied to machine translation (Sutskever et al., 2014), grapheme-to-phoneme conversion (Yao and Zweig, 2015), and so on. The difference is that the proposed encoder-labeler LSTM accepts the same input sequence twice while the usual encoder-decoder LSTM accepts the input sequence once in the encoder. Note that the LSTMs for encoding and labeling are different in the encoder-labeler LSTM, but the same word embedding matrix is used both for the encoder and labeler since the same input sequence is fed twice.

3 Experiments

We first explain the experimental setup. Then we report the results to confirm the improvement by the proposed encoder-labeler LSTM. Finally, we compare our results with the published results while discussing the related works.

3.1 Experimental Setup

We used the ATIS corpus, which has been widely used as the benchmark for NLU (Price, 1990; Dahl et al., 1994; Wang et al., 2006; Tur et al., 2010). Figure 2 shows an example sentence and its semantic slot labels in In-Out-Begin (IOB) representation. The slot filling task was to predict the slot label sequences from input word sequences.

The performance was measured by the F_1-score: $F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$, where precision is the ratio of the correct labels in the system's output and recall is the ratio of the correct labels in the ground truth of the evaluation data (van Rijsbergen, 1979).

The ATIS corpus contains the training data of 4,978 sentences and evaluation data of 893 sentences. The unique number of slot labels is 127 and the vocabulary size is 572. In the following experiments, we randomly selected 80% of the original training data to train the model and used the remaining 20% as the heldout data (Mesnil et al., 2015). We reported the F_1-score on the evaluation data with hyper-parameters that achieved the best F_1-score on the heldout data.

For training, we randomly initialized parameters in accordance with the normalized initialization (Glorot and Bengio, 2010). We used *ADAM* for learning rate control (Kingma and Ba, 2014) and *dropout* for generalization with a dropout rate of 0.5 (Srivastava et al., 2014; Zaremba et al., 2014).

3.2 Improvement by Encoder-labeler LSTM

We conducted experiments to compare the labeler LSTM(W) (Section 2.1), the labeler LSTM(W+L) (Section 2.2), and the encoder-labeler LSTM (Section 2.3). As for yet another baseline, we tried the encoder-decoder LSTM as shown in Figure 1(c)[1].

For all architectures, we set the initial learning rate to 0.001 (Kingma and Ba, 2014) and the dimension of word embeddings to $d_e = 30$. We changed the number of hidden units in the LSTM, $d_h \in \{100, 200, 300\}$[2], and the size of the context window, $k \in \{0, 1, 2\}$[3]. We used backward encoding for the encoder-decoder LSTM and the encoder-labeler LSTM as suggested in Sutskever et al. (2014). For the encoder-decoder LSTM, labeler LSTM(W+L), and encoder-labeler LSTM(W+L), we used the left-to-right beam search decoder (Sutskever et al., 2014) with beam sizes of 1, 2, 4, and 8 for evaluation where the best F_1-score was reported. During 100 training epochs, we reported the F_1-score on the evaluation data with the epoch when the F_1-score for the heldout data was maximized. Table 1 shows the results.

By comparing labeler LSTM(W) and labeler LSTM(W+L), we found improvement by explicitly modeling label dependencies. The proposed encoder-labeler LSTM(W) and encoder-labeler LSTM(W+L) both outperformed the labeler LSTM(W) and labeler LSTM(W+L) by considering sentence-level information. It is noteworthy that the sentence-level information used by the proposed encoder-labeler LSTM(W+L) is effective on top of the label dependencies modeled by the labeler LSTM(W+L).

Contrary to expectations, F_1-score by the encoder-labeler LSTM(W+L) was worse than that by the encoder-labeler LSTM(W). A possible reason is that the model structure of the encoder-labeler LSTM(W+L) is more complex than that of the encoder-labeler LSTM(W) and optimization is more difficult with the limited training data of ATIS.

[1] Length of the output label sequence is equal to that of the input word sequence in a slot filling task. Therefore, ending symbol for slot sequence is not necessary.

[2] When using deep architecture later in this section, d_h was tuned for each layer.

[3] In our preliminary experiments with using the labeler LSTM(W), F_1-scores deteriorated with $k \geq 3$.

		F_1-score
(c)	Encoder-decoder LSTM	80.11
(a)	Labeler LSTM(W)	94.80
(d)	**Encoder-labeler LSTM(W)**	**95.29***
(b)	Labeler LSTM(W+L)	94.91
(e)	**Encoder-labeler LSTM(W+L)**	**95.19**
	Labeler Deep LSTM(W)	94.91
	Encoder-labeler Deep LSTM(W)	**95.47***

Table 1: Experimental results on ATIS slot filling task. Leftmost column corresponds to Figure 1. Lines with bold fonts use proposed encoder-labeler LSTM. F_1-scores with $*$ outperformed published best F_1-score of 95.25%. [%]

For the encoder-labeler LSTM(W) which was better than the encoder-labeler LSTM(W+L), we tried the deep architecture of 2 LSTM layers (*Encoder-labeler deep LSTM(W)*). We also trained the corresponding *labeler deep LSTM(W)*. As in Table 1, we obtained improvement from 94.91% to 95.47% by the proposed encoder-labeler deep LSTM(W).

Lastly, F_1-score by the encoder-decoder LSTM was worse than that by the labeler LSTM(W). Since the slot label is closely related with the input word, the encoder-decoder LSTM was not an appropriate approach for the slot filling task.

3.3 Comparison with Published Results

Table 2 summarizes the recently published results on the ATIS slot filling task and compares them with the results from the proposed methods.

Recent research has been focusing on RNN and its extensions. Yao et al. (2013) used RNN and outperformed methods that did not use neural networks, such as SVM (Raymond and Riccardi, 2007) and CRF (Deng et al., 2012). Mesnil et al. (2015) tried bi-directional RNN, but reported degradation comparing with their single-directional RNN (94.98%). Bi-directional RNN was not effective after using context window in this task. Yao et al. (2014a) introduced LSTM and deep LSTM and obtained improvement over RNN. Peng and Yao (2015) proposed RNN-EM that used an external memory architecture to improve the memory capability of RNN.

Many studies have been also conducted to explicitly model the label dependencies. Xu and Sarikaya (2013) proposed CNN-CRF that explicitly models the dependencies of the output from CNN. Mesnil et al. (2015) used hybrid RNN that combined Elman-type and Jordan-type RNNs. Liu and Lane (2015) used the output label for the previous word to model label dependencies (RNN-SOP).

	F_1-score
RNN (Yao et al., 2013)	94.11
CNN-CRF (Xu and Sarikaya, 2013)	94.35
Bi-directional RNN (Mesnil et al., 2015)	94.73
LSTM (Yao et al., 2014a)	94.85
RNN-SOP (Liu and Lane, 2015)	94.89
Hybrid RNN (Mesnil et al., 2015)	95.06
Deep LSTM (Yao et al., 2014a)	95.08
RNN-EM (Peng and Yao, 2015)	95.25
Encoder-labeler LSTM(W)	**95.40**
Encoder-labeler Deep LSTM(W)	**95.66**

Table 2: Comparison with published results on ATIS slot filling task. [%]

By comparing with these methods, the main difference of our proposed encoder-labeler LSTM is the use of encoder LSTM to leverage sentence-level information. For our encoder-labeler LSTM(W) and encoder-labeler deep LSTM(W), we further conducted hyper-parameter search with a random search strategy (Bergstra and Bengio, 2012). We tuned the dimension of word embeddings, $d_e \in \{30, 50, 75\}$, number of hidden states in each layer, $d_h \in \{100, 150, 200, 250, 300\}$, size of context window, $k \in \{0, 1, 2\}$, and initial learning rate sampled from uniform distribution in range $[0.0001, 0.01]$. To the best of our knowledge, the previously published best F_1-score was 95.25%[4] (Peng and Yao, 2015). Our encoder-labeler LSTM(W) and encoder-labeler deep LSTM(W) achieved 95.40% and 95.66% F_1-score, respectively, both outperforming the previously published F_1-score as shown in Table 2.

Note some of the previous results used whole training data for model training while others used randomly selected 80% of data for model training and the remaining 20% for hyper-parameter tuning. Our results are based on the latter setup.

4 Conclusion

We enhanced the LSTM-based slot filling to model label dependencies and proposed an encoder-labeler LSTM to leverage sentence-level information. By combining these methods, slot filling with jointly considering label dependencies and sentence-level

[4] There are other published results that achieved better F_1-scores by using other information on top of word features. Vukotic et al. (2015) achieved 96.16% F_1-score by using the named entity (NE) database when estimating word embeddings. Yao et al. (2013) and Yao et al. (2014a) used NE features in addition to word features and obtained improvement with both the RNN and LSTM.

information became possible. We obtained the state-of-the-art F_1-score in a slot filling task with using the standard ATIS corpus. Our future work includes evaluation on other data sets.

References

[Bengio2009] Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[Bergstra and Bengio2012] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

[Dahl et al.1994] Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proc. HLT*, pages 43–48.

[De Mori et al.2008] Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding. *IEEE Signal Processing Magazine*, 3(25):50–58.

[Deng et al.2012] Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tur. 2012. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *Proc. SLT*, pages 210–215.

[Elman1990] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc AISTATS*, pages 249–256.

[Hinton et al.2006] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Jordan1997] Michael I Jordan. 1997. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

[Jozefowicz et al.2015] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. ICML*, pages 2342–2350.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Liu and Lane2015] Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

[Mesnil et al.2015] 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.

[Peng and Yao2015] Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

[Price1990] Patti Price. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proc. DARPA Speech and Natural Language Workshop*, pages 91–95.

[Raymond and Riccardi2007] Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Proc. INTERSPEECH*, pages 1605–1608.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112.

[Tur et al.2010] Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2010. What is left to be understood in ATIS? In *Proc. SLT*, pages 19–24.

[van Rijsbergen1979] Cornelis Joost van Rijsbergen. 1979. *Information Retrieval*. Butterworth.

[Vukotic et al.2015] Vedran Vukotic, Christian Raymond, and Guillaume Gravier. 2015. Is it time to switch to word embedding and recurrent neural networks for spoken language understanding? In *Proc. INTERSPEECH*, pages 130–134.

[Wang et al.2006] Ye-Yi Wang, Alex Acero, Milind Mahajan, and John Lee. 2006. Combining statistical and knowledge-based spoken language understanding in conditional models. In *Proc. COLING-ACL*, pages 882–889.

[Williams and Peng1990] Ronald J Williams and Jing Peng. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2(4):490–501.

[Xu and Sarikaya2013] Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular

CRF for joint intent detection and slot filling. In *Proc. ASRU*, pages 78–83.

[Yao and Zweig2015] Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *Proc. INTERSPEECH*, pages 3330–3334.

[Yao et al.2013] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Proc. INTERSPEECH*, pages 2524–2528.

[Yao et al.2014a] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014a. Spoken language understanding using long short-term memory neural networks. In *Proc. SLT*, pages 189–194.

[Yao et al.2014b] Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014b. Recurrent conditional random field for language understanding. In *Proc. ICASSP*, pages 4077–4081.

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