# Recurrent Neural Network Structured Output Prediction for Spoken Language Understanding

**Bing Liu, Ian Lane**
Department of Electrical and Computer Engineering
Carnegie Mellon University
{liubing,lane}@cmu.edu

## Abstract

Recurrent Neural Networks (RNNs) have been widely used for sequence modeling due to their strong capabilities in modeling temporal dependencies. In this work, we study and evaluate the effectiveness of using RNNs for slot filling, a key task in Spoken Language Understanding (SLU), with special focus on modeling label sequence dependencies. Recent work on slot filling using RNNs did not model label sequence dependencies explicitly. We propose to introduce label dependencies in model training by feeding previous output label, using a sampling approach on true and predicted labels, to the current sequence state. The revised RNN model is evaluated on slot filling task using standard ATIS dataset. Our results show that the proposed methods consistently outperform the baseline RNN system. Error analysis illustrates clear advantages of learning output sequence dependencies in RNN sequence modeling.

## 1 Introduction

Slot filling involves searching input text to fill in values for predefined slots in a reference knowledge base [6, 16]. It is one of the key tasks in spoken language understanding. Slot filling can be framed as a sequence labeling task, which maps an observation sequence $\mathbf{x} = \{x_1, \ldots, x_T\}$ to a sequence of labels $\mathbf{y} = \{y_1, \ldots, y_T\}$, where input and output sequence are of same length. Popular approaches to solving sequence labeling problems include generative models such as Hidden Markov Models (HMMs), and discriminative models such as Conditional Random Fields (CRFs) and Support Vector Machines (SVMs). Recently, Recurrent Neural Networks (RNNs) showed promising performance in speech recognition, language modeling and other sequence labeling tasks [5, 18]. RNNs' feedback nature and non-linear activation enable such models to learn sequence representations with complex structure and over long distance, which are critical in sequence learning.

A number of discriminative models such as CRF [19] and SVM [9] have been applied to slot filling tasks. In this work, we focus on applying RNNs to slot filling, with special attention on modeling output label dependencies. Previous work on using RNNs for slot filling [12, 21] with Elman [17] and Jordan [8] type models achieved favourable F1-scores comparing to CRF approach. These models demonstrated RNNs' advantages in feature representations by using fixed length continuous word embedding and modeling long term word relations with recurrent connections. Local optimization on output label $y_t$ following maximum likelihood criteria at each step $t$ was performed using input word features till current step $x_1^t$ and potentially future time steps with a defined context window $x_{t+1}^{t+ws}$, where $ws$ denotes the size of future context. Although sequence level optimization was not addressed specifically, these RNN models still achieved reasonably good F1-measurement on ATIS (Airline Travel Information Systems) benchmark.

Recent review [11] on using RNNs for slot filling discussed potential approaches in conducting sequence level optimization, including using Viterbi decoding with slot language models and using

a combination of RNNs and CRF (Recurrent CRF) [20]. These approaches attempted to ameliorate label bias problem [10] in sequence predictions. In this paper, we propose to model output label conditional dependencies using a sampling approach with previous true and predicted labels, similar to the scheduled sampling method proposed in [1]. Instead of only using previous true labels and the summarized input in current state to maximize the likelihood of output state during training, using samples from previous predicted label distribution help to make the model more robust by forcing it to learn to handle its own mistakes.

The remainder of the paper is organized as follows. In section 2, we introduce the background on using RNNs for slot filling. Related approaches in conducting sequence level optimization for slot filling are also described. We present our approach in modeling structured output in section 3. Section 4 discusses the experiment setup and result analysis on ATIS dataset. Section 5 gives the conclusions and recommendations.

## 2 Background

### 2.1 RNNs for Slot Filling

RNNs use continuous word representations as slot filling model input. Distributed representation of words in vector space has been applied to a number of natural language processing tasks [2, 4, 11]. Comparing to using word as atomic units, distributed word representation can better capture the syntactic and semantic regularities in language [12]. One popular architecture for training such distributed word representations was introduced in [14], where word vector representation can be jointly learned with a statistical language model. Continuous bag-of-words (CBOW) and skip-gram models were later introduced as efficient methods for learning high-quality distributed vector representations using simple model architectures and very large data set [13].

Feed-forward neural networks using context window were first applied for sequence learning [2]. RNNs extend feed-forward neural networks by introducing recurrent connections. Such recurrent connections enable RNNs to deal with sequence of variable length and learn long term dependencies. RNN parameters are trained using Back-propagation Through Time (BPTT) considering influence of past states through recurrent connections. Thus, the network is capable of capturing information passed from previous states by maintaining and learning a summarized states of previous steps. This enables RNNs to perform sequence prediction tasks with longer range comparing to standard feed-forward neural networks that use input of fixed size cascaded context.

In Elman RNNs, the recurrent hidden layer state at time $t$ can be represented as:

$$h_t = f(Ux_t + Wh_{t-1}) \tag{1}$$

where $U$ is the weight matrix between input layer and hidden layer, and $W$ is the weight matrix for the recurrent connections. $f$ is the non-linear activation function. On output layer, softmax function is applied to frame network output as valid probability distribution:

$$y_t = \mathrm{softmax}(Vh_t) \tag{2}$$

where $V$ is the weight matrix between hidden layer and output layer, and

$$\mathrm{softmax}(z_m) = \frac{e^{z_m}}{\Sigma_k e^{z_k}} \tag{3}$$

### 2.2 Sequence Level Optimizations

Elman RNN performs token level optimization as opposed to sequence level optimization. The model produces locally optimized label prediction using input features without considering dependencies between output labels. Thus, it is suboptimal to model sequence that with correlations among output labels, which is typically the case in most sequence learning tasks. A number of methods have been proposed to address such label bias problems [10]. Viterbi decoding with slot language models and recurrent CRF are two of them that have been applied in recent slot filling studies.

The slot language model approach [11] borrows similar idea from the speech community that uses a weighted combination of different types of probability models to optimize the posterior probability.

In this approach, a trigram language model of output labels is built to model the state or label transition probability. Observation likelihood is modeled using output probability distribution from Elman RNNs by applying Bayes rule:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{4}$$

$$= \arg\max_{\mathbf{y}} P(\mathbf{y})^{\alpha} \times P(\mathbf{x}|\mathbf{y}) \tag{5}$$

$$\sim \arg\max_{\mathbf{y}} P(\mathbf{y})^{\alpha} \times \prod_{t} \frac{P(y_t|x_1^t)}{P(y_t)} \tag{6}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output sequences. The first term $P(\mathbf{y})$ is modeled by the trigram label language model, and the $P(y_t|x_1^t)$ is from RNNs output probability distribution. The weight $\alpha$ is a tunable value representing the contribution from the label language model. Output sequence with highest probability can then be searched via Viterbi decoding. It is reported in [11] that this Viterbi search method produced 0.01 point F1-score increase on ATIS dataset, and larger increases on a few other datasets.

Another approach proposed in doing sequence level optimization is the recurrent CRF method [20]. This method combines the feature representation power in RNNs and sequence level discriminative capability of CRF. CRF objectives are used to jointly learn the transition probability and RNNs parameters. CRF models the conditional probability of the output label sequence given input word sequence by:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \prod_{t=1}^{T} \exp(\sum_{m=1}^{M} \lambda_m h_m(y_{t-1}, y_t, x_{t-ws}^{t+ws})) \tag{7}$$

where $Z$ is the normalization constant, $h_m(y_{t-1}, y_t, x_{t-ws}^{t+ws})$ are the feature representations of word input and output labels between $t-1$ and $t$. Influence from the previous and future input context windows on $y_t$ are represented by $x_{t-ws}^{t+ws}$, with left and right context size of $ws$. In recurrent CRF, $h_m(y_{t-1}, y_t, x_{t-ws}^{t+ws})$ decomposes to label transition feature $h_m(y_{t-1}, y_t)$ and label output feature $h_m(y_t, x_{t-ws}^{t+ws})$. The latter is modeled by values in RNNs output layer before softmax activation, which helps to ameliorate label bias problem according to [20]. This approach results in 0.28 point F1-score improvement on ATIS dataset using words and named-entity information as reported in their study.

## 3 Proposed Approach

We propose to optimize slot label sequence prediction by considering not only input sequences but also label sequence from previous steps. Different from Jordan RNN architecture or hybrid Elman and Jordan RNN architecture in which output label probabilities are fed to recurrent connections, a fixed sized continuous vector representing one particular label is fed. These continuous vectors representing possible output labels are trained together with other network parameters. During training, the label to be fed to recurrent connections can be the previous true label, or a sampled label drawn from previous predicted label distribution. We train the network to find the best parameter set $\theta$ that maximizes the likelihood:

$$\arg\max_{\theta} \prod_{t=1}^{T} P(y_t|y_1^{t-1}, \mathbf{x}; \theta) \tag{8}$$

In this revised RNN, at each step the current word input, previous hidden state, and previous output label are connected to current hidden state:

$$h_t = f(Ux_t + Wh_{t-1} + Qy\_out_{t-1}) \tag{9}$$

where $y\_out_{t-1}$ is the vector representing output label at time $t-1$, and $Q$ is the weight matrix connecting output label vector and the hidden layer. Same as Elman RNNs, hidden layer is projected to the output layer, and a softmax transformation is added to obtain the probability distribution for the predicted label at time $t$.

$$y_t = \mathrm{softmax}(Vh_t) \tag{10}$$

In Elman RNNs, the hidden layer at time step $t$ carries information from all previous input sequence $x_1^t$ via recurrent connection. In the revised model, output label information from $y\_out_{t-1}$ is also fed to $h_t$ at each time step $t$. Thus, at step $t$ the label prediction $y_t$ takes into consideration of all previous output labels and the input sequence.

During inference, given an input sequence $\mathbf{x}$, we want to find the best label sequence $\mathbf{y}$ such that:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{11}$$

True label is not available during inference and sample drawn from previous predicted label distribution is used. Decoding is performed with beam search similar to [3, 18]. At each time step, each beam is expanded to all possible output labels, and only the top $\beta$ beams are kept based on the accumulated scores. The best label sequence is selected from the $\beta$ beams at the last time step.

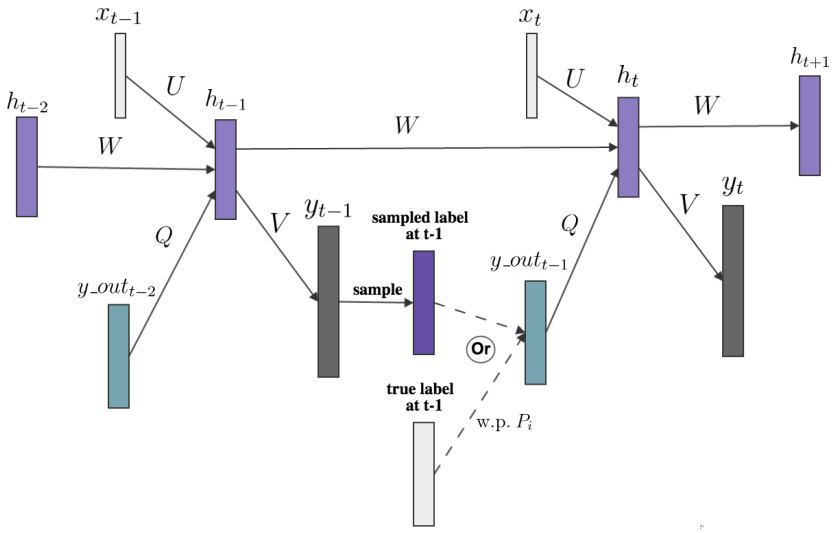


Figure 1: RNN with sampled label $y\_out$ connected to hidden layer.

Figure 1 shows the revised RNN with sampled label $y\_out$ connected to hidden layer. At time step $t$, information from word input $x_t$, previous hidden layer $h_{t-1}$, and output label from previous step $y\_out_{t-1}$ are transferred to $h_t$. During training, true label at time step $t-1$ can be used as $y\_out_{t-1}$. However, during inference, true label is not available, and only the predicted label can be used. This results in discrepancy on how the model is trained and is used during inference [1]. The prediction power of RNN trained in this manner may not generalize well during inference. Error made by wrongly predicted label in early stage of the sequence may be amplified significantly since all following predictions use such wrong label information.

To introduce robustness in RNN training, we use a sampling approach that randomly decides whether to use true label or predicted label from previous time step. We want to introduce some randomness to the output label so that the RNN model can learn to handle the mistake made by itself. The predicted label can be the label with highest probability (from softmax transformation output), or a sampled label following label output probability distribution at $t-1$. We used the former approach in our study. In RNN training, we choose the true label at step $t-1$ with probability $P_i$, and choose the predicted label at $t-1$ with probability $1-P_i$. In this work, we study the impact of using different values of $P_i$ on the RNN slot filling performance. Specifically, we study the differences between using a constant $P_i$ across entire training stage, and using a $P_i$ value that linearly decreases over training iterations.

## 4 Experiments

ATIS (Airline Travel Information Systems) dataset [7] is widely used in spoken language understanding research. The dataset contains audio recordings of people making flight reservations. Text and corresponding named entities and semantic labels are also provided. In this experiment, only word features are used in building RNN slot filling model.

The slot filling task training set contains 4978 utterances (56590 tokens) from ATIS-2 and ATIS-3 corpora, and test set contains 893 utterances (9198 tokens) from ATIS-3 NOV93 and DEC94 datasets. This is the same corpus that was used in other related slot filling studies with ATIS dataset [11, 20]. There are a total number of 127 distinct slot labels (including null label). Vocabulary size is 572. Precision, recall, and F1-score were measured as evaluation matrics.

The network structure is as described in previous section. We used sigmoid function for non-linear activation. To prevent the gradient from exploding, a clipping range between -5 to 5 is defined. Dropout (dropout rate $p = 0.2$ on input nodes) and L2 regularization $\lambda = 1e-7$ were used to prevent the RNNs from overfitting. However, we found the improvement brought by these regularization methods were limited for ATIS slot filling task. To incorporate future context features, we used context window approach with window size of 4 on both sides of current time step. During decoding, we kept top 8 beams at each time step for beam search.

The Elman RNN model trained with task specific embedding and random initialization achieved best F1-score of 94.52, advancing best score using CRF model 92.36 by large margin. RNNs using Google News embedding [15] achieved best F1-score of 93.56 without fine-tuning on word embeddings. After fine tuning, F1-score increased to 94.65. This fine-tuned system was used as baseline system for our study. For the numbers reported in below tables, we used Google News word embedding to initialize the word vectors. Only word features in ATIS dataset were used for slot filling in our study. Name-entities and intent information were not used.

Table 1: *Performance Measurement on ATIS dataset (Word feature only)*

| Model | Precision (%) | Recall (%) | F1-Score |
|---|---|---|---|
| CRF | 93.12 | 91.61 | 92.36 |
| Elman RNN baseline | 94.87 | 94.43 | 94.65 |
| RNN trained with true labels | 95.17 | 94.40 | 94.78 |
| RNN trained with sampled label constant $P_i$ = 0.80 | 95.24 | 94.54 | **94.89** |
| RNN trained with sampled label linearly deceasing $P_i$ | 95.27 | 94.47 | 94.87 |

First set of experiments applied constant $P_i$ values over training iterations, therefore true label and predicted label were used with fixed probability during the training stage. We experimented with a fixed $P_i$ values over iterations ranging from 0.20 to 1.0, and the corresponding system performance was illustrated in Figure 2. $X$ axis represents systems using different $P_i$ values, and $Y$ axis gives the F1-score on ATIS test set. As can be seen from the results, all systems with output label feedback connections outperformed the baseline RNN system. With fixed $P_i$ value of 1.0, only true labels were used in the output feedback connection. As discussed in section 3, system trained in this approach gained capability in modeling output sequence dependencies, but with discrepancy between training and inference. The higher F1-score achieved with lower $P_i$ values (e.g. $P_i$ = 0.8) was consistent with such analysis. When $P_i$ values dropped further, predicted labels were more likely used during RNNs training, and the F1-score started to decrease. However, even with very low $P_i$ value, the model still beat the baseline system.

A typical example of label prediction error that appeared in baseline system and was corrected by the revised model is as below. Given a test sentence of "`...on May seventeenth one way with dinner`", the baseline system predicted corresponding output sequence as "`...O B-depart_date.month_name B-arrive_date.day_number B-round_trip I-round_trip O B-meal_description`", where the label for "`seventeenth`" was predicted wrongly. Without modeling output label dependencies, the semantic label for "`seventeenth`" can be hard to decide as it can either be related with departure or arrival.

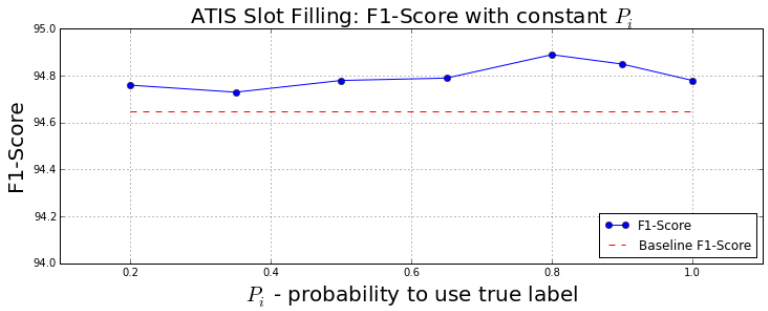


Figure 2: F1-score with constant $P_i$ over training iterations.

This error was avoided in the revised system when we modeled output label dependencies explicitly with label feedback connection. By seeing "`B-depart_date.month_name`" as previous label, the system was able to predict current label as "`B-depart_date.day_number`" instead of "`B-arrive_date.day_number`" with high confidence.

We further studied RNNs that were trained with time varying $P_i$. As a system is more likely to make prediction errors at the beginning stage of the training, we applied a linearly decreasing $P_i$ value for output label feedback connection. A minimum $P_i$ value was defined for each experiment, and $P_i$ value dropped linearly from 1.0 to the minimum value defined along the training epochs. Results of systems using different minimum $P_i$ value were reported in Table 2. Again, all systems beat the baseline RNN system consistently. Best F1-score 94.87 was achieved using minimum $P_i$ value of 0.65. However, systems with different $P_i$ decay rate resulted in very similar performance in this ATIS slot filling evaluation. Training logs revealed that training and test errors converged very quickly during the initial 20+ training epochs. Further tuning started from a point where predicted output label could be generated with reasonably low error rate. This might partially explain the similar F1-scores achieved by systems using various minimum $P_i$ value in this evaluation.

Table 2: *Performance Measurement on ATIS dataset - with linearly decreasing $P_i$*

| Model | Precision (%) | Recall (%) | F1-Score |
|---|---|---|---|
| RNN with $P_i = 1.0 \rightarrow 0.80$ | 95.24 | 94.43 | 94.83 |
| RNN with $P_i = 1.0 \rightarrow 0.65$ | 95.27 | 94.47 | **94.87** |
| RNN with $P_i = 1.0 \rightarrow 0.50$ | 95.20 | 94.36 | 94.78 |
| RNN with $P_i = 1.0 \rightarrow 0.35$ | 95.24 | 94.43 | 94.83 |
| RNN with $P_i = 1.0 \rightarrow 0.20$ | 95.20 | 94.43 | 94.82 |

## 5 Conclusion

In this work, we studied using RNNs for SLU slot filling task, with particular attention on modeling output sequence dependencies. Modeling structured output is vital to many sequence learning tasks. We proposed to model slot label dependencies using a sampling approach, by feeding sampled output label (true or predicted) back to the sequence state. We evaluated this approach in slot filling task on ATIS dataset, and observed consistent performance gain over the baseline RNN system. Error analysis showed clear advantage of learning output sequence dependencies in RNN model training. Performance of a number of sampling approaches were also compared and discussed. Further investigation of output label error propagation in the network memory is part of future work. Moreover, it is worth exploring combining sampling strategy with other sequence level optimization methods for enhanced sequence labeling capability.

## References


[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*, 2015.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

[4] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[5] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[6] Y. He and S. Young. A data-driven spoken language understanding system. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 583–588. IEEE, 2003.

[7] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington. The atis spoken language systems pilot corpus. In *Proceedings, DARPA speech and natural language workshop*, pages 96–101, 1990.

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

[9] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

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

[11] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):530–539, 2015.

[12] G. Mesnil, X. He, L. Deng, and Y. Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775, 2013.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] T. Mikolov, S. Kombrink, L. Burget, J. H. Černockỳ, and S. Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[16] C. Raymond and G. Riccardi. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*, pages 1605–1608, 2007.

[17] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

[18] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[19] Y.-Y. Wang and A. Acero. Discriminative models for spoken language understanding. In *INTERSPEECH*, 2006.

[20] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao. Recurrent conditional random field for language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4077–4081. IEEE, 2014.

[21] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu. Recurrent neural networks for language understanding. In *INTERSPEECH*, pages 2524–2528, 2013.