INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

ClockWork-RNN based architectures for Slot Filling

Despoina Georgiadou, Vassilios Diakoloukas, Vassilios Tsiaras, Vassilios Digalakis

Technical University of Crete, Department of Electrical & Computer Engineering, Greece
dgeorgiadou@isc.tuc.gr, vdiak@telecom.tuc.gr, vass.tsiaras@gmail.com, vas@telecom.tuc.gr

Abstract

A prevalent and challenging task in spoken language understanding is slot filling. Currently, the best approaches in this domain are based on recurrent neural networks (RNNs). However, in their simplest form, RNNs cannot learn long-term dependencies in the data. In this paper, we propose the use of ClockWork recurrent neural network (CW-RNN) architectures in the slot-filling domain. CW-RNN is a multi-timescale implementation of the simple RNN architecture, which has proven to be powerful since it maintains relatively small model complexity. In addition, CW-RNN exhibits a great ability to model long-term memory inherently. In our experiments on the ATIS benchmark data set, we also evaluate several novel variants of CW-RNN and we find that they significantly outperform simple RNNs and they achieve results among the state-of-the-art, while retaining smaller complexity.

Index Terms: slot filling (SF), spoken language understanding (SLU), clock-work recurrent neural network (CW-RNN)

1. Introduction

The slot filling (SF) task is an essential issue in spoken language understanding (SLU) [1]. In recent years, artificial neural networks are on the spotlight of most researches on the SF task [2]. Studying the state-of-the-art, it is clear to see that almost all the neural network architectures proposed for the slot filling task are a variant of the simple RNN architecture [3].

RNNs are really promising when it comes to temporal dependencies. Even simple RNN architectures deal sufficiently well with short-term dependencies of the input words. However, they lack the ability to learn long-term context due to the vanishing gradient issue [4].

In the SF task, long-term dependencies which relate the current-time semantic label prediction to the observations several time instances away are often required to be considered. Consequently, the state-of-the-art approaches mainly focus on addressing this problem [5, 6, 7, 8, 9, 10]. However, as the input data dependencies that need to be modeled become more complex, the neural network architecture becomes more complex and the number of parameters that have to be estimated increases.

The work presented here has focused on adding multiple timescales in the neurons of the RNN model in an effort to better capture long-term dependencies, while retaining relatively small complexity. Specifically, we explore the use of clockwork RNN [11] for the slot filling task. In addition, we propose three novel variants, the Hybrid CW-RNN, the Convolutional CW-RNN and the bidirectional CW-RNN in an effort to further improve the model performance in this task.

Although the CW-RNN architecture has been successfully used for other tasks, such as audio signal generation, spoken word classification and on-line handwriting recognition [11], it has never been used in the SF task. A similar time-scale architecture on LSTMs was only applied for modeling long texts [12]. CW-RNNs inherently model long-term dependencies without needing extra memory or increased number of parameters. As a result, the computational cost is lower. In addition, CW-RNN is relatively simple both to understand and implement and provides a great flexibility in the architecture configuration. All these benefits make it a good neural network candidate for the SF task as well.

The rest of this paper is organized as follows: Section 2 provides a brief overview of related RNN variants. Section 3 presents the proposed CW-RNN models. Section 4 demonstrates the experimental results. Finally, section 5 offers concluding remarks.

2. RNN variants

For a long time the predominant models for the SF task [1] in SLU were the popular CRFs [13] and SVMs [14]. In 2013, simple RNNs [3] were investigated for the SF task and outperformed all previous approaches. Since then, several RNN variants were proposed, which either experiment with different connection types inside the network such as the Bi-directional [9, 15], the hybrid [15] and the RNN-SOP [10] or add extra memory such as LSTM [5] and RNN-EM [6]. Many other combinations have been investigated such as the encoder-labeler LSTM [7], Bi-directional RNN-LSTM [16] and the Bi-directional RNN using a ranking loss function [8] which showed great performance. Recently, attention-based RNN [17] and knowledge-guided structural attention networks [18] have shown benefits on the language understanding task.

3. Proposed methods

A typical uni-directional RNN consists of input, hidden and output layers. In the SF task, the input $x_t \in \mathbb{R}^{n_{in}}$ at time t is a real-valued vector associated to each word (word embeddings). The output of the hidden layer at a time step t is then defined as:

$$h_t = f\Big(Wx_t + Uh_{t-1} + b\Big) \tag{1}$$

where $W \in \mathbb{R}^{n_h \times n_{in}}$ and $U \in \mathbb{R}^{n_h \times n_h}$ are the weight matrices of the input and the hidden layer respectively, $b \in \mathbb{R}^{n_h}$ serves as a bias vector and $h_0 \in \mathbb{R}^{n_h}$ is the initial state. The function f at the hidden layer is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The output layer can be described by the following equation

$$o_t = softmax\left(W_o h_t + b_o\right) \tag{2}$$

where $W_o \in \mathbb{R}^{n_{out} \times n_h}$ and $b_o \in \mathbb{R}^{n_{out}}$ define the weight matrix and bias vector of the output layer.

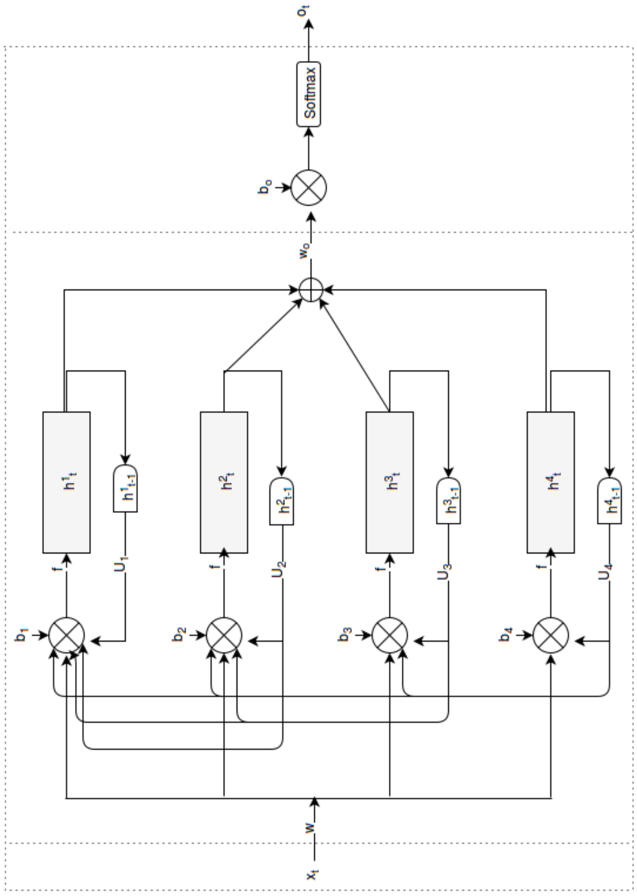

Figure 1: *A ClockWork RNN: In this example, with $T_1 < T_2 < T_3 < T_4$, the module m=1 is the fastest one and the module m=4 is the slowest one. Information flows from the slower to the faster modules.*

The CW-RNN is the same as the RNN above except that the hidden layer neurons are partitioned into m chunks (modules) of $k = n_h/m$ neurons each (figure 1). For each module $i \in \{1, \cdots, m\}$, a different clock period $T_i \in \{T_1, ..., T_m\}$ is set which defines when each module will be activated and updated. In this paper we use a period $T_i = 2^{i-1}$ as in [11], although different clock period definitions can also be investigated.

We can, therefore, partition matrix $U = (U_{ij})$, $i, j \in \{1, \ldots, m\}$ into m^2 block sub-matrices, U_{ij}, of size $k \times k$, matrix $W = (W_i)$ into m block sub-matrices of size $k \times n_{in}$, and vector $b = (b_i)$ into m sub-vectors of size $k \times 1$.

The connections among the neurons of the network depend on the time period. Specifically, although the neurons within each module are fully interconnected, the recurrent connections from module j to module i exist only if the period T_i is smaller than the period T_j. In other words, the connections between the modules propagate the hidden state from slower modules to faster modules, which means the block U_{ij} is the zero matrix when $T_i > T_j$ [11].

At each time step t, only the output of modules i that satisfy $(t \bmod T_i = 0)$ are active, where mod denotes modulo. The other modules retain their output values from the previous time-step. Therefore, the hidden state vector h_t is partitioned into m sub-vectors of size k, $h_t = [h_t^{1\top}, \ldots, h_t^{i\top}, \ldots, h_t^{m\top}]^\top$, where h_t^i is the output of the i-th module at time-step t:

$$h_t^i = \begin{cases} f\left(\sum\limits_{j=i}^{m} U_{ij} h_{t-1}^j + W_i x_t + b_i\right) & \text{if } t \bmod T_i = 0 \\ h_{t-1}^i & \text{otherwise} \end{cases}$$

The CW-RNN is a simplified RNN architecture, since using a smaller number of connections, decreases the number of parameters and the overall complexity of the network. Since a relatively small number of modules is activated at each time step, the computational complexity is largely decreased when training is performed in normal processing units.

3.1. Bidirectional CW-RNN

Our bidirectional-CWRNN (biCW-RNN) is based on the bidirectional RNN [15, 9]. The bidirectional approach is able to exploit both future and past information in the input.

The hidden state vector h_t for the biCW-RNN is the addition of the forward $h_t^{(f)}$ as stated in Section 3 for the simple CWRNN and a backward $h_t^{(b)}$. The total hidden state is therefore $h_t = h_t^{(f)} + h_t^{(b)}$.

3.2. Hybrid CW-RNN

The hybrid RNN [15] is a simple combination of the Elman [19] and the Jordan [20] architectures. The combined model is employed to learn more data patterns as both the past hidden activity and the past prediction o_{t-1} is fed back to the hidden layer.

When CW-RNN is considered, several hidden layer modules are inactive at most time steps. The output layer of the network, which can also be seen as an average over past time-steps, can be set as feedback to the hidden layer. Specifically, we obtain the hidden layer dynamics using both the output posterior probabilities and the influence of past hidden states of the active models. In this way, we introduce a hybrid CW-RNN architecture which combines the recurrences from the Jordan and the Elman models within the CW-RNN framework. The dynamics of our hybrid CW-RNN can be described from the following equation:

$$h_t^i = \begin{cases} f\left(\sum\limits_{j=i}^{m} U_{ij} h_{t-1}^j + V_i o_{t-1} + W_i x_t + b_i\right) & \text{if } t \bmod T_i = 0 \\ h_{t-1}^i & \text{otherwise} \end{cases}$$

where o_t is given by (2) and the weight matrix, $V \in \mathbb{R}^{n_h \times n_{out}}$ of the connections from the output to the hidden layer, is partitioned into m blocks $V = [V_1^\top, \ldots, V_i^\top, \ldots, V_m^\top]^\top$ of size $k \times n_{out}$.

3.3. Convolutional CW-RNN

Previous studies [13, 21] have shown that with the introduction of a convolutional layer after the neural network's input layer it is possible to extract useful higher-level features from local patches of the raw input data which can better exploit short-term dependencies. We, therefore, explore the use of an additional convolutional layer between the input and the hidden layer of the CW-RNN. In this way, our Convolutional CW-RNN combines the strengths of both the CW-RNN architecture and the convolutional layer and will be able to better exploit both long- and short-distance dependencies among the sentence words.

Similarly to [22], we form the input matrix $X \in \mathbb{R}^{n_{in} \times N}$ by concatenating the N word embeddings x_t from an input sentence. Then a 2D filter defined through a filter matrix $F \in \mathbb{R}^{f_1 \times f_2}$ is used to perform wide convolution with the input matrix X. To address the problem of the missing words in the borders, we perform padding with past and future words. If we define $l_1 = \lfloor f_1/2 \rfloor$ and $l_2 = \lfloor f_2/2 \rfloor$, the convolution is described as follows:

$$(X*F)(i,t) = \sum_{j=-l_1}^{l_1} \sum_{\tau=-l_2}^{l_2} x_{t+\tau}(i+j)F(l_1+1-j, l_2+1-\tau)$$

where $i \in [-l_1+1, \ldots, n_{in}+l_1]\}$, $t \in [-l_2+1, N+l_2]$.

The filter values are considered to be parameters of the layer initialized randomly from a uniform(-1,1) distribution and estimated during model training with back-propagation. During the forward pass, we convolve each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. In this work, we consider the size of the filter 5×5. Since the size of the features obtained from the convolutional layer is higher that the size of the input, we add a max-pooling layer after the convolutional layer. Max-pooling is a popular technique in image processing which is a form of non-linear down-sampling. The main idea is to partition convolutional output and to keep the maximum value for each part. In this way, we perform dimensionality reduction and we obtain robust features.

4. Experiments

4.1. Corpus Analysis

We choose to perform our experiments in the widely used Airline Travel Information System (ATIS) domain in order to be able to compare our results to other SF approaches in the literature. ATIS [23] consists of audio recordings from speakers which make flight reservations or ask relevant information. For each of the recordings there is an annotated sentence as well as an annotated synchronous semantic interpretation (label) using the BIO (**B**egin - **I**nside - **O**utside) format. The labels were created by [24] from the original annotations. In this way, all words have a corresponding semantic label of either type B, I or O. For words that do not have a semantic meaning we assign the "`null`" label. In total, there are 127 different types of slots (tags) including "`null`".

The original training-set includes 4978 sentences from Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora. In our experiments we used the same database configuration as in [3, 15]. Specifically, 20% of the original training data was randomly chosen to form the validation set and the rest 80% was used as training set. To evaluate our model we used a test set containing 893 sentences which were obtained from the ATIS-3 Nov93 and Dec94 datasets. The dataset provides us with the training set, the validation set, the test set and the dictionary. For each sentence in each set we have this information: word id, name entity id and label id.

4.2. Data Processing

The task of SF is to predict the semantic label for each of the words in the ATIS sentences. To obtain proper input for our neural network architectures we need to represent each of the words in the sentences as d-dimensional feature vectors. Specifically, following the same data processing method as in [15, 25, 26] each word in the sentence is represented as an integer corresponding to the vocabulary id. Furthermore, for each of the word in the sentence we apply a context window which creates a vector representation by embedding the previous n_w and the next n_w words. In this way, short-term dependencies can be efficiently represented into the feature vector. Thus, the sentence is now represented by a matrix of integers. Finally, we associate this matrix of indexes to randomly selected embeddings which results to a continuous real-valued representation of the input sentence. Additional features could be incorporated into our model, however we use lexical features which are automatically extracted from word sequences to keep our model as simple as possible and make the model comparison more meaningful.

4.3. Experiments and Results

Similarly to other works, in our experiments we evaluated our model performance using the macro $F1$ score, the script of which was provided in the text chunking CoNLL shared task 2000 [24, 9]. For the implementation of the artificial neural networks, we used Python and the Theano toolkit [27]. To train our model we applied sentence-level Stochastic Gradient Descent (SGD) and Early-Stopping based on the validation data set. The objective function that is minimized is the mean negative log-likelihood. For all of our neural networks we use random sampling of the hyper-parameters [28]. As can be seen in Table 1 which summarizes the experimental configuration, we considered $n_h = 200$ neurons for all models, including the Vanilla RNN. The CW-RNN hidden layer was partitioned into $m = 4$ modules with clock periods $t_1 = 1, t_2 = 2, t_3 = 4, t_4 = 8$.

Table 1: *Model's parameters*

parameter	**value**
ATIS folds	$\{1,2,3,4,5\}$
Context window size	$\{3,5,7,9\}$
Number of hidden units	$\{200\}$
Dimension of word embedding	$\{50,100\}$
initianl Learning rate	0.0001
early-stopping epochs	$\{100\}$
embedding matrix initialization	$unif(-1,1)$
weight matrices initialization	$unif(-1,1)$
convolution filter size	$\{5 \times 5\}$
convolution filter initialization	$unif(-1,1)$

Table 2 demonstrates the $F1$ score for our model in its simple Elman-type architecture and we compare its performance to the novel Hybrid CW-RNN architecture and to the convolutional CW-RNN. As can be seen the Hybrid CW-RNN achieves slightly outperforms the regular CW-RNN. The Convolutional CW-RNN on the other hand is the best performing system, obtaining a relative increase of 0.21% and 0.19% in the $F1$ score compared to the simple CW-RNN and the Hybrid CW-RNN respectively. It is important to note that all the CW-RNN architectures outperform the regular (Vanilla) RNN in the same task.

Table 2: *F1-scores of CW-models in this work comparing to their simple versions for the same number of hidden units*

Architecture	**CW-RNN**	**Vanilla RNN**
Elman-type	**95.23**	94.98
Hybrid	**95.25**	95.06
Bi-directional	**95.31**	94.73
Convolutional	**95.44**	-

4.4. Comparison to the state-of-the-art

By construction, CW-RNN has less parameters than an RNN or an LSTM with the same number of neurons. For instance, assuming that a CW-RNN has its hidden layer partitioned into m modules and each module consists of k neurons, this makes a total of $n_h = k \times m$ neurons. Since each neuron receives input only from the neurons having the same or larger clock period, the number of parameters for the recurrent matrix U of CW-RNN is:

$$\text{parameters}(U) = \sum_{j=1}^{m} jk^2 = \frac{n_h^2 + n_h k}{2}$$

For a direct comparison, the recurrent matrix of a similar sized RNN has n_h^2 parameters which means that the RNN has $\frac{2m}{m+1}$ more parameters than the CW-RNN. For instance, for a typical experiment with $m = 4$, the number of RNN parameters are 1.6 times more than the corresponding CW-RNN.

Furthermore, although both the CW-RNN and the RNN have the same number of input connections, which are represented by matrix W this is not the case when LSTMs are considered. The blocks of an LSTM are described by 7 interconnection matrices, which account for $7 \times n_h^2$ parameters, and by 4 input matrices.

The CW-RNN also reduces the vanishing gradient problem which is prominent to other RNN-based architectures. Since a module i with period T_i is updated every T_i time steps and the back-propagation in time is also performed with steps of length T_i, the error signal for slow modules remains strong enough during longer time intervals. For example, given a sequence $\{(x_i, y_i) \mid i = 1, \ldots, N\}$ of data sample pairs, the parameter estimation formulae for the slowest module m of the CW-RNN are identical to the formulae corresponding to an RNN having the same size as the module m and is estimated from the following sequence of data samples:

$$\{(x_1, \bar{y}_t), (x_{T_m+1}, \bar{y}_{T_m+1}), (x_{2T_m+1}, \bar{y}_{2T_m+1}), \ldots\},$$

where x_t is now sampled at $t \in \{1, T_m + 1, 2T_m + 1, 3T_m + 1, \ldots\}$, instead of $t \in \{1, 2, \ldots\}$ in the Vanilla RNN. The mean value $\bar{y}_{iT_m+1}$ of y_t for $iT_m + 1 \leq t \leq (i+1)T_m$, $i \in \{0, 1, \ldots\}$, is defined as:

$$\bar{y}_{iT_m+1} = \frac{1}{T_m} \sum_{t=iT_m+1}^{(i+1)T_m} y_t$$

In Table 3 we compare the performance of the proposed CW-RNN variants to the current state-of-the-art. It is worth noting that all of our CW-RNN variants are among the top performers in the state-of-the-art. They significantly outperform many of the competitive approaches, among them several with a much higher complexity.

5. Concluding remarks

In this work, we presented a multi-timescale version of RNN, called clockwork-RNN, to perform the slot filling task for spoken language understanding. CW-RNN is a neural network architecture with a great potential, since by construction it can learn long-term dependencies of the input data, a property which is important in the SF task.

We investigated three novel architectures in the CW-RNN, a hybrid Elman-Jordan architecture, a convolutional CW-RNN and a bi-directional CW-RNN and we obtained a performance which is comparative to the state-of-the-art in the SF task. This is a very encouraging result, indicating that more research effort should be invested in the CW-RNN variants for SF in the future.

Table 3: *Best F1 scores on ATIS gained by the simple version of each model, without additional features or data sources*

model	**F1(%)**
RNN [3]	94.11
CNN-CRF [13]	94.35
Bi-directional RNN [15]	94.73
LSTM [5]	94.85
RNN-SOP [10]	94.89
K-SAN [18]	95
Hybrid RNN [15]	95.06
Deep LSTM [5]	95.08
RNN-EM [6]	95.22
CW-RNN	**95.23**
Hybrid CW-RNN	**95.25**
Bi-directional CW-RNN	**95.31**
Encoder-labeler LSTM(W) [7]	95.40
CNN TriCRF [13]	95.42
Convolutional CW-RNN	**95.44**
R-biRNN [9]	95.47
5xR-biRNN [9]	95.56
Bi-directional CNN [22]	95.61
Encoder-labeler Deep LSTM(W) [7]	95.66
Attention-Based RNN [17]	95.78

6. References

[1] G.Tur and R.DeMori, “Spoken language understanding: Systems for extracting semantic information from speech,” pp. 81–104, 1996.

[2] G. Mesnil, X. He, L. Deng, and Y. Bengio, “Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding.” in *INTERSPEECH*. ISCA, 2013, pp. 3771–3775.

[3] K. Yao, G. Zweig, M. Hwang, Y. Shi, and D. Yu, “Recurrent neural networks for language understanding.” Interspeech, 2013.

[4] S. Hochreiter and J. Schmidhuber, “Long short-term memory, neural computation,” vol. 9(8), 1997, pp. 1735–1780.

[5] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory,” in *Neural Networks*, 2014, pp. 189–194.

[6] B. Peng, K. Yao, L. Jing, and K. Wong, “Recurrent neural networks with external memory for spoken language understanding.” in *NLPCC*, ser. Lecture Notes in Computer Science, vol. 9362. Springer, 2015, pp. 25–35.

[7] G. Kurata, B. Xiang, B. Zhou, and M. Yu, “Leveraging sentence-level information with encoder LSTM for natural language understanding.” *CoRR*, vol. abs/1601.01530, 2016.

[8] Y. Shi, K. Yao, H. Chen, D. Y. Y.-C. Pan, and M. Hwang, “Recurrent support vector machines for slot tagging in spoken language understanding,” in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Microsoft, 2016.

[9] N. T. Vu, P. Gupta, H. Adel, and H. Schütze, “Bi-directional recurrent neural network with ranking loss for spoken language understanding,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, 2016, pp. 6060–6064.

[10] B. Liu and I. Lane, “Recurrent neural network structured output prediction for spoken language understanding,” in *Proc. NIPS*

Workshop on Machine Learning for Spoken Language Understanding and Interactions, 2015.

[11] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber, “A clockwork RNN,” in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1863–1871.

[12] P. Liu, X. Qiu, X. Chen, S. Wu, and X. Huang, “Multi-timescale long short-term memory neural network for modelling sentences and documents,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 2015, pp. 2326–2335.

[13] P. Xu and R. Sarikaya, “Convolutional neural network based triangular crf for joint intent detection and slot filling.” IEEE, Institute of Electrical and Electronics Engineers, 2013.

[14] T. Kudo and Y. Matsumoto, “Chunking with support vector machines,” in *NAACL*, 2001.

[15] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, and e. a. D. Yu, “Using recurrent neural networks for slot filling in spoken language understanding.” *IEEE/ACM Trans. Audio, Speech and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[16] D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, “Multi-domain joint semantic frame parsing using bi-directional rnn-lstm,” in *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH)*. San Francisco, CA: ISCA, 2016.

[17] B. Liu and I. Lane, “Attention-based recurrent neural network models for joint intent detection and slot filling,” *CoRR*, vol. abs/1609.01454, 2016. [Online]. Available: http://arxiv.org/abs/1609.01454

[18] Y. Chen, D. Z. Hakkani-Tür, G. Tür, A. Çelikyilmaz, J. Gao, and L. Deng, “Knowledge as a teacher: Knowledge-guided structural attention networks,” *CoRR*, vol. abs/1609.03286, 2016. [Online]. Available: http://arxiv.org/abs/1609.03286

[19] J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[20] M. I. Jordan, “Serial order: A parallel distributed processing approach,” no. ICS Report 8604, 1986.

[21] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, “A latent semantic model with convolutional-pooling structure for information retrieval,” in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, 2014, pp. 101–110.

[22] N. T. Vu, “Sequential convolutional neural networks for slot filling in spoken language understanding,” *CoRR*, vol. abs/1606.07783, 2016.

[23] P. J. Price, “Evaluation of spoken language systems: The ATIS domain,” in *Proc. of the Speech and Natural Language Workshop*, 1990, pp. 91–95.

[24] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in *INTERSPEECH 2007, 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, August 27-31, 2007*, 2007, pp. 1605–1608.

[25] Y. Bengio, R. Ducharme, and P. Vincent, “A neural probabilistic language model,” in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, 2000, pp. 932–938.

[26] R. Collobert and J. Weston, “A unified architecture for natural language processing: deep neural networks with multitask learning,” in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, 2008, pp. 160–167.

[27] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. T. D. Warde-Farley, and Y.Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

[28] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.