RECURRENT CONDITIONAL RANDOM FIELD FOR LANGUAGE UNDERSTANDING

Kaisheng Yao[1]*, Baolin Peng*[1,2]*, Geoffrey Zweig*[1]*, Dong Yu*[1]*, Xiaolong Li*[1]*, and Feng Gao*[1]

[1]Microsoft Corporation
[2]Beihang University, China

{kaisheny, v-bape, gzweig, dongyu, xiaolli, fegao}@microsoft.com

ABSTRACT

Recurrent neural networks (RNNs) have recently produced record setting performance in language modeling and word-labeling tasks. In the word-labeling task, the RNN is used analogously to the more traditional conditional random field (CRF) to assign a label to each word in an input sequence, and has been shown to significantly outperform CRFs. In contrast to CRFs, RNNs operate in an online fashion to assign labels as soon as a word is seen, rather than after seeing the whole word sequence. In this paper, we show that the performance of an RNN tagger can be significantly improved by incorporating elements of the CRF model; specifically, the explicit modeling of output-label dependencies with transition features, its global sequence-level objective function, and offline decoding. We term the resulting model a "recurrent conditional random field" and demonstrate its effectiveness on the ATIS travel domain dataset and a variety of web-search language understanding datasets.

Index Terms— Conditional random fields, recurrent neural networks

1. INTRODUCTION

In recent years, Recurrent Neural Networks (RNNs) have demonstrated outstanding performance in a variety of natural language processing tasks [1–9]. In common with feed-forward neural networks [10–14], an RNN maintains a representation for each word as a high-dimensional real-valued vector. Critically, in this vector space, similar words tend to be close with each other, and relationships between words are preserved [15]; thus, adjusting the model parameters to increase the objective function for a training example which involves a particular word tends to improve performance for similar words in similar contexts.

In this paper, we focus on the use of RNNs in Spoken Language Understanding (SLU). In classical SLU systems [16–22], one of the key tasks is to label words with semantic meaning. For example, in the sentence "I want to fly from Seattle to Paris," the word "Seattle" should be labeled as the departure-city of a trip, and "Paris" as the arrival-city. Perhaps the most obvious approach to this task is the use of Conditional Random Fields (CRFs) [23], in which an exponential model is used to compute the probability of a label sequence given the input word sequence. A CRF produces the single, globally most likely label sequence, and the model has been widely used in SLU [20, 24, 25]. Other sequence labeling methods that have been investigated include Maximum Entropy Markov Models (MEMMs) [20, 26] and Machine Translation (MT) models [27].

Our current work improves upon the RNN language understanding (RNN-LU) architecture of Yao et al. [7]. This architecture consists of a layer of inputs connected to a set of hidden nodes; a fully connected set of recurrent connections amongst the hidden nodes; and a set of output nodes. In the SLU task, the inputs are the sequence of words, and the outputs are the sequence of semantic labels. This basic architecture is illustrated in Figure 1 and described in detail in Section 2.1.

While effective, the previous architecture does not explicitly model the dependencies between semantic labels. For SLU, Mesnil et al. [8] investigated another architecture, the Jordan architecture [28], that feeds back the past predictions of labels to the hidden layer. It is reported in [8] that incorporating such feedback improves label accuracy. A second drawback to the basic RNN-LU architecture is that it is optimized based on a tag-by-tag likelihood as opposed to a sequence-level objective function. In common with MEMM models, the RNN produces a sequence of locally-normalized output distributions, one for each word position. Thus, it can suffer from the same label bias [23] problem. To ameliorate these problems, we propose to combine the RNN and CRF. The combined model can be considered as an RNN that uses the CRF-like sequence-level objective function or as a CRF that uses the RNN activations as features. The whole model is jointly trained, taking advantage of the sequence-level discrimination ability of a CRF and the feature learning ability of an RNN. This model is illustrated in Figures 3 and 4.

This approach is similar to the prior work of sentence-level neural networks proposed in [29, 30] and the NN-CRF fusion presented in [31–34]. Whereas previous work used feed-forward or convolutional networks, we propose to use an RNN. This is motivated by the RNN's demonstrated performance in word-labeling tasks [7, 8]. This approach is different from sequence-level training of deep neural networks (DNNs) for acoustic modeling in [35–37]: whereas [35–37] use HMMs and DNNs, this work uses CRFs and RNNs.

2. BACKGROUND

2.1. Recurrent Neural Networks

The RNN-LU architecture is a feature-augmented [5] Elman architecture [38]. Figure 1 shows an example of the architecture "unrolled" across time to cover three consecutive word inputs. This architecture consists of a feature layer, an input layer, a hidden layer with recurrent connections, and an output layer. Each layer represents a set of neurons, and the layers are connected with weights. The input layer $\mathbf{w}(t)$ represents input word at time t encoded using 1-of-N coding, and the feature layer $\mathbf{f}(t)$ can be used to encode additional information such as topic, or dialog state. The feature layer $\mathbf{f}(t)$ encodes side-information, and is connected to the hidden layer with weights $\mathbf{F}$ and the output layer with weights $\mathbf{G}$. Besides of encoding topical information as in [5], the feature layer can also be used to convey a redundant representation of the input by using continuous-space vector representations of the words.

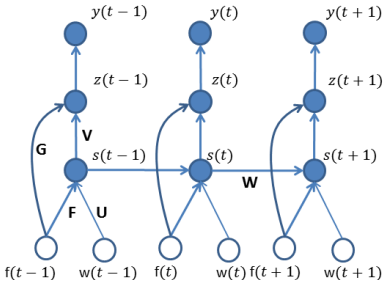

Fig. 1. RNN language understanding architecture. An open circle indicates that the value of the variable is given. Matrices are shown in bold. $\mathbf{w}(t)$ represents word input; $\mathbf{f}(t)$ auxiliary features; $\mathbf{z}(t)$ the output distribution prior to the soft-max; and $\mathbf{y}(t)$ the output label.

Such representations can be learned by a non-augmented network (in which the input layer only connects to the hidden layer) as described in [1, 7, 29].

The output layer $\mathbf{y}(t)$ produces a probability distribution over labels. The hidden layer $\mathbf{s}(t)$ maintains a representation of the sentence history. The input vector $\mathbf{w}(t)$ has a dimensionality equal to the vocabulary size, and the output vector $\mathbf{y}(t)$ has a dimensionality equal to the number of possible semantic labels. The values in the hidden and output layers are computed as follows:

$$\mathbf{s}(t) = f\left(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1) + \mathbf{F}\mathbf{f}(t)\right) \quad (1)$$

$$\mathbf{z}(t) = \mathbf{V}\mathbf{s}(t) + \mathbf{G}\mathbf{f}(t), \quad (2)$$

$$\mathbf{y}(t) = g\left(\mathbf{z}(t)\right), \quad (3)$$

where

$$f(z) = \frac{1}{1+e^{-z}}, \quad g(z_m(t)) = \frac{e^{z_m(t)}}{\sum_k e^{z_k(t)}}. \quad (4)$$

and $\mathbf{U}$, $\mathbf{W}$, $\mathbf{F}$, $\mathbf{V}$, and $\mathbf{G}$ are the connection weights. $z_m(t)$ is the m-th element in the output layer activity before softmax; i.e., $z_m(t) = \left(\mathbf{V}\mathbf{s}(t) + \mathbf{G}\mathbf{f}(t)\right)_m$. We find it useful to explicitly represent the activations before the softmax, as we will use these later in the R-CRF.

This model uses an on-line decoding process that outputs the likeliest semantic label based on only the past observations. Note that this is optimal with respect to the model as there is no direct dependence between output labels. The RNN model is trained with the maximum conditional likelihood criterion, listed in Table 1. Its error signal for error back-propagation is $\delta(y(t) = y^\star(t)) - \frac{\exp(z_{y(t)}(t))}{\sum_j \exp(z_j(t))}$, where $y^\star(t)$ represents the correct label at position t.

2.2. Conditional Random Field

The recurrent neural network produces a position-by-position distribution over output labels, and thus can suffer from the same label bias problem as MEMMs and other locally normalized models. In contrast to locally normalized models, a conditional random field (CRF) [23], illustrated in Fig 2, is a sequence model consisting of a single exponential model for the joint probability of the entire sequence of labels given the observation sequence. The joint probability $P(\mathbf{y}(1:T)|\mathbf{w}(1:T))$ has the form

$$\frac{1}{Z_{CRF}} \exp\left(\sum_{t=1}^{T}\left(\sum_m \lambda_m f_m(y(t-1), y(t)) + \sum_k \mu_k g_k(y(t), \mathbf{w}(t))\right)\right), \quad (5)$$

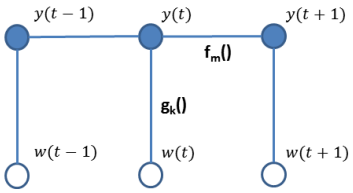

Fig. 2. CRF.

where $f_m(y(t-1), y(t))$ is the m-th edge feature between labels $y(t-1)$ and $y(t)$. $g_k(y(t), \mathbf{w}(t))$ is the k-th vertex feature at position t. Z_{CRF} is for normalization. In the CRF, the edge and vertex features are assumed to be given. For example, a Boolean vertex feature g_k might be true if the word at position t is upper case and the label $y(t)$ is "proper noun". The CRF illustrated in Fig 2 is a linear chain CRF (LC-CRF), which has dependence between labels in neighboring positions. The Semi-Markov CRF (semi-CRF) [39], which usually outperforms LC-CRF, uses a segment-level model, grouping together multiple inputs and assigning a single output label.

3. RECURRENT CONDITIONAL RANDOM FIELD

3.1. Model

In the recurrent CRF model proposed here, an RNN is used to generate the input features for a CRF. This model has two variants as illustrated in Figures 3 and 4. In both variants, the features used are the RNN scores before softmax normalization; i.e., at each position t for a label k, the new model uses $z_k(t)$ in Eq. (4). Using the activation before the softmax is important because the softmax normalizes its input scores and can thus cause the label bias problem that exactly motivates the CRF model. The same trick is also adopted in tandem ASR systems [40].

Because this new model is a CRF with features generated from an RNN, we call it a recurrent conditional random field (R-CRF). A R-CRF naturally incorporates dependencies between semantic labels via the CRF transition features. In Figure 4, we further connect the past one-hot prediction to the hidden layer, similar to that in the Jordan architecture. This prediction is the maximum of the distribution over output labels at time t, given only the observations up to time t. In this model, the one-hot predictions contribute to the subsequent output layer via a non-linear hidden layer. As an alternative to these models, we might use the maximum entropy features of Mikolov et al. [4], which consist of features defined on ngrams of output labels. However, we find that the max-entropy approach is not as effective in the SLU task. Decoding in the R-CRF is done as in a CRF, with a forward pass over all the words, followed by a backtrace.

Denote the dimensions of feature layer, hidden layer and output layer respectively as F, H, and Y. The computational cost of the R-CRF during training is $O(HH + FH + HY + FY)$ for each word. The cost of Viterbi decoding is $O(T(HH + FH + HY + FY))$ for a sequence with length T.

3.2. Objective Function

For simplicity of notation, we denote input-output pair sequence during training as $(\mathbf{w}(1:t), \mathbf{y}(1:t))$. This can be easily generalized to include the side feature inputs, for which we would then have $((\mathbf{w}(1:t), \mathbf{f}(1:t)), \mathbf{y}(1:t))$.

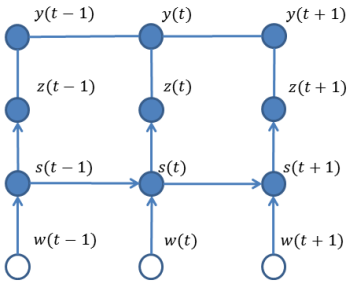

Fig. 3. R-CRF Model 1. An open circle indicates that the variable is not generated by the model. For clarity, we have omitted the feature inputs and connections.

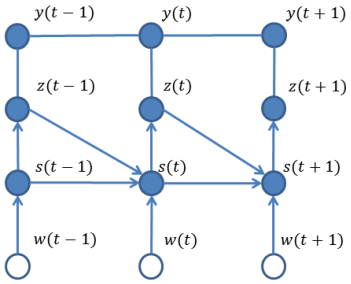

Fig. 4. R-CRF Model 2. The one-hot prediction is connected to the hidden layer in R-CRF model 2.

We slightly simplify the CRF definition of Eqn. (5) by absorbing the weight μ_k associated with a feature g_k into the feature itself, in our case the weights in the final layer of the network. This allows us to define μ_k to be 1 without a loss in generality. With this notation, the objective function for a single training example is defined as

$$\frac{\exp \sum_{t=1}^{T} \left(\eta a_{y^\star(t-1)y^\star(t)} + z_{y^\star(t)}(t)\right)}{\sum_{\forall \mathbf{y}(1:T)} \exp \left(\sum_{t=1}^{T} \eta a_{y(t-1)y(t)} + z_{y(t)}(t)\right)}, \tag{6}$$

where $\mathbf{y}^\star(1:T) = [y^\star(1) \cdots y^\star(T)]$ denotes the correct label sequence. $a_{y(t-1)y(t)}$ is the transition feature from label $y(t-1)$ to $y(t)$, which can also be considered as the label transition weight in log-scale. $\eta \in \mathcal{R}^+$ is a real value, usually set to 1.0. It is used to weight the transition feature.

For clarity, table 1 summarizes the objective functions for RNN, CRF, and R-CRF in log-scale. The RNN uses a word-level normalization $Z_{RNN}(t)$ at position t. Both CRFs and R-CRFs use sequence-level normalizations. Different from CRFs that combine fixed and pre-defined vertex features $g_k(\cdot)$ through linear combinations, R-CRFs learn vertex features through recurrent and non-linear transformations in RNNs.

To maximize the R-CRF objective function, the algorithm iterates between a forward pass and a backward pass during training. We now describe the training and decoding procedures, based on the standard CRF recursions [23].

3.3. Training

The forward pass computes the scores along all possible input-label pair sequences in the denominator in Eq. (6) and the score along the correct input-label pair sequence. As shown in [23], the necessary quantities can be computed with dynamic programming.

Table 1. Log-scale Objective Functions $Q(\theta)$ of RNN, CRF, and R-CRF. RNN normalization is $Z_{RNN}(t) = \sum_k \exp z_k(t)$. CRF normalization is $Z_{CRF} = \sum_{\forall \mathbf{y}(1:T)} \exp \sum_{t=1}^{T} \{\sum_m \lambda_m f_m(y(t-1), y(t)) + \sum_k \mu_k g_k(y(t), \mathbf{w}(t))\}$. R-CRF normalization is $Z_{R-CRF} = \sum_{\forall \mathbf{y}(1:T)} \exp \left(\sum_{t=1}^{T} \eta a_{y(t-1)y(t)} + z_{y(t)}(t)\right)$.

RNN	$\sum_{t=1}^{T} \left(z_{y^\star(t)}(t) - \log Z_{RNN}(t)\right)$
CRF	$\sum_{t=1}^{T} \{\sum_m \lambda_m f_m(y^\star(t-1), y^\star(t)) + \sum_k \mu_k g_k(y^\star(t), \mathbf{w}(t))\} - \log Z_{CRF}$
R-CRF	$\sum_{t=1}^{T} \left(\eta a_{y^\star(t-1)y^\star(t)} + z_{y^\star(t)}(t)\right) - \log Z_{R-CRF}$

Define $\alpha(t, i)$ as the sum of partial path scores ending at position t, with label i. This can be computed as follows

$$\alpha(t, i) = \exp(z_i(t)) \sum_j \alpha(t-1, j) \exp(\eta a_{ji}) \tag{7}$$

The backward pass score $\beta(t-1, q)$ is defined as the sum of partial path scores starting at position $t-1$, with label q and exclusive of observation $t-1$. It can be recursively computed [23] as

$$\beta(t-1, q) = \sum_j \beta(t, j) \exp(\eta a_{qj} + z_j(t)) \tag{8}$$

With the above forward and backward scores, we can compute gradients with respect to vertex feature $z_{y(t)=k}(t)$ at position t and label $y(t) = k$ as follows

$$\frac{\partial Q(\theta)}{\partial z_{y(t)=k}(t)} = \delta(y(t) = y^\star(t)) - \frac{\alpha(t, k)\beta(t, k)}{\sum_j \alpha(t, j)\beta(t, j)} \tag{9}$$

With the above equation, we obtain the error signal for RNN at each position t. The model then reuses the backpropagation procedures for updating RNN parameters, except that the error signals are now with (9).

To update the label transition weights, we compute gradients as follows:

$$\begin{aligned} \frac{\partial Q(\theta)}{\partial a_{ji}} &= \eta \sum_t \delta(y(t-1) = j, y(t) = i) \\ &- \eta \sum_t \left(\frac{\alpha(t-1, j)\beta(t, i) \exp(\eta a_{ji} + z_i(t))}{\sum_j \alpha(t, j)\beta(t, j)} \right) \end{aligned} \tag{10}$$

The model parameters θ are updated using stochastic gradient ascent (SGA) over the training data multiple passes.

4. EXPERIMENTS

4.1. Datasets

In order to evaluate the proposed model, we conducted two sets of experiments. The first set of experiments used the widely used ATIS dataset [16,41,42]. This dataset focuses on the air travel domain, and consists of audio recordings of people making travel reservations. In this database, each word in a sentence is labeled with a semantic tag.

The second set of data was obtained from Microsoft Bing's internal query understanding system. It consists of search queries from four domains: Restaurants, Movies, Flights, and Hotels. The tasks

Table 2. Training and testing setups in ATIS and other domains.

	ATIS	Flights	Restaurants	Hotels	Movies
# train words	56590	15695	28470	9159	19943
# test words	9198	4149	3047	2353	2258
# slots	128	42	32	36	38
Significance Level	0.44	0.93	1.2	1.3	1.7

Table 3. F1 score with different modeling techniques.

	Flights	Restaurants	Hotels	Movies
LC-CRF	88.58	85.63	85.80	75.70
semi-CRF	90.90	86.50	86.60	80.60
RNN	90.51	86.77	88.03	78.20
R-CRF Model 1	91.54	88.46	87.95	82.21
R-CRF Model 2	90.53	87.97	89.12	82.61

consist of labeling words with semantically-important tags such as "director" or "genre" in the movie domain. Table 2 lists the number of words in the training and test sets for all the datasets used in this work. The numbers of distinct slot labels are also listed. In the last row of Table 2 we show the amount of change in the F1 measure necessary for the improvement to be significant at the significance level of 95%.

4.2. Results on Bing Query Understanding Datasets

Table 3 lists the F1 scores achieved by the R-CRF models, together with those from a linear-chain CRF (LC-CRF), semi-Markov CRF (semi-CRF), and an RNN. The LC-CRF and semi-CRF models used four types of features that were derived from regular expression matches and context-free-grammar parses. The LC-CRF uses a context window size of 5 and the semi-CRF uses a context window size of 3, which includes left and right context. RNNs and R-CRFs used these same features (by extending the sparse input representation $w(t)$ with the additional information), with a context window size of 2. These sizes and feature extraction are optimal for the respective systems. The RNNs and R-CRFs also use SENNA word embedding [29] as redundant input representations in the feature layer. For comparison, we also applied word embedding to CRFs in the movies domain and obtained small improvements and did not exceed RNN's result.

Referring to Table 3, we notice that the semi-CRF outperforms the LC-CRF in all domains, because of its ability to model long-span dependencies between labels. The RNN outperforms the LC-CRFs in all domains, but only outperforms semi-CRFs in the restaurant and hotel domains. However, with sequence-discriminative training and transition features, R-CRF Model 1 is able to outperform semi-CRFs in all domains. The improvement is particularly significant in the movies and restaurants domains, because of the strong dependencies between labels. For instance, a movie name has many words and each of them has to have the same label of "movie_name". Therefore, it is beneficial to incorporate dependencies between labels, and train at the sequence level. Adding the feedback connections of Model 2 helps in some domains, but is not a consistent improvement.

In theory, the R-CRF has a computational cost that is between LC-CRF and semi-CRF. Empirically, on the same machine, where the average latencies of the LC-CRF and semi-CRF models are 0.1 ms and 3.8 ms, respectively, the latencies of the RNN and R-CRF models are respectively 0.51ms and 0.53ms. The improvement in latency achieved by the R-CRF over the semi-CRF is especially significant for long queries. For queries with 9 or more words, the latency is reduced from 14.5ms to 1.2ms when we switch from the semi-CRF to the R-CRF.

Table 4. F1 score for the ATIS domain.

CRF	RNN	R-CRF Model 1	R-CRF Model 2
94.40	96.37	96.41	96.65

Table 5. Sources of R-CRF performance improvement (Movies).

RNN	Train label transition only	Full backpropagation to RNN
77.04	78.54	82.40

4.3. Results on the ATIS Dataset

Table 4 lists the results of different modeling techniques on the ATIS set. The training features include word and named-entity information as described in [43]. We use a context window of 3 for bag-of-word feature [7]. RNN and R-CRF use 100-dimension hidden layer. As shown in the table, the RNN significantly outperforms CRF, improving F1 score from 94.4% by CRF to 96.4%. Using the R-CRF produced the best F1 score of 96.7%, though the RNN-based models all do very well on this task. This score is slightly higher than 96.6% reported in [7] that, together with [8], shows that RNN-based methods outperform other methods [20, 43] including DNN-based [44].

4.4. Sources of Performance Improvement

Note that the R-CRF includes transition features as well as RNN features. Hence, we conducted experiments to understand their relative importance. Because of the large improvement of the R-CRF over the RNN on the Movies task, we focus on that domain. As shown in Table 5, the RNN has a 77.04% F1 score. This differs from Table 3 due to a different model configuration. Including the label transition weights, which model dependencies between slots, improved the F1 score to 78.54%. When we updated RNN parameters in the R-CRF, we observed a further improvement in the F1 score to 82.40%. Therefore, much of the improvement can be attributed to the modeling in the RNN.

We also conducted experiments to determine the importance of using word embeddings as a redundant input representation. All models used a 200-dimension hidden layer. Without word embeddings, R-CRF Model 1 and Model 2 obtained F1 scores of 81.2% and 83.2%, compared to 71.2% with a plain RNN. With word embeddings, R-CRF Model 1 and R-CRF Model 2 obtained 82.2% and 82.6%, compared with 76.6% for the RNN. These results show that R-CRFs are less reliant on redundant input representations.

5. CONCLUSION

In this paper we have demonstrated that RNNs can be successfully merged with CRFs to do language understanding. There are two qualitative benefits from this merge. First, it naturally incorporates the CRF's global sequence-level optimization criterion that is more appropriate for sequence-tagging problems. Second, the CRF transition features directly model interactions between the output labels. Our experiments indicate that this new model outperforms linear-chain CRFs, semi-CRFs, and RNNs.

6. REFERENCES

[1] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010, pp. 1045–1048.

[2] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network based language model," in *ICASSP*, 2011, pp. 5528–5531.

[3] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Cernocky, "Empirical evaluation and combination of advanced language modeling techniques," in *INTERSPEECH*, 2011, pp. 605–608.

[4] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, "Strategies for training large scale neural network language models," in *ASRU*, 2011.

[5] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. SLT*, 2012, pp. 234–239.

[6] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, 2012, pp. 20–28.

[7] K. Yao, G. Zweig, M. Hwang, Y. Shi, and Dong Yu, "Recurrent neural networks for language understanding," in *INTERSPEECH*, 2013.

[8] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for language understanding," in *INTERSPEECH*, 2013.

[9] M. Auli, M. Galley, C. Quirk, and G. Zweig, "Joint language and translation modeling with recurrent neural networks," in *EMNLP*, 2013.

[10] H. Schwenk and J.L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *ICASSP*. IEEE, 2002, vol. 1, pp. I–765.

[11] Y. Bengio, R. Ducharme, Vincent, P., and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Reseach*, vol. 3, no. 6, 2003.

[12] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492 – 518, 2007.

[13] H.-S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *ICASSP*, 2011, pp. 5524–5527.

[14] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the international workshop on artificial intelligence and statistics*, 2005, pp. 246–252.

[15] T. Mikolov, W.T. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *NAACL-HLT*, 2013.

[16] C. Hemphill, J. Godfrey, and G. Doddington, "The ATIS spoken language systems pilot corpus," in *Proceedings of the DARPA speech and natural language workshop*, 1990, pp. 96–101.

[17] P. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the Third DARPA Speech and Natural Language Workshop*. Morgan Kaufmann, 1990, pp. 91–95.

[18] W. Ward et al., "The CMU air travel information service: Understanding spontaneous speech," in *Proceedings of the DARPA Speech and Natural Language Workshop*, 1990, pp. 127–129.

[19] Y. He and S. Young, "A data-driven spoken language understanding system," in *ASRU*, 2003, pp. 583–588.

[20] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," *INTERSPEECH*, pp. 1605–1608, 2007.

[21] R. De Mori, "Spoken language understanding: A survey," in *ASRU*, 2007, pp. 365–376.

[22] F. Béchet, "Processing spontaneous speech in deployed spoken language understanding systems: a survey," *SLT*, vol. 1, 2008.

[23] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *ICML*, 2001, pp. 282–289.

[24] Y.-Y. Wang, A. Acero, M. Mahajan, and J. Lee, "Combining statistical and knowledge-based spoken language understanding in conditional models," in *COLING/ACL*, 2006, pp. 882–889.

[25] A. Moschitti, G. Riccardi, and C. Raymond, "Spoken language understanding with kernels for syntactic/semantic structures," in *ASRU*, 2007, pp. 183–188.

[26] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proc. EMNLP*, 1996.

[27] K. Macherey, F. Och, H. Ney, et al., "Natural language understanding using statistical machine translation," in *European Conf. on Speech Communication and Technology*. Citeseer, 2001, pp. 2205–2208.

[28] M. Jordan, "Serial order: A parallel distributed processing approach," Tech. Rep. 8604, Institute for Cognitive Science, University of California at San Diego, 1986.

[29] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, Aug 2011.

[30] A.-R. Mohamed, D. Yu, and L. Deng, "Investigation of full-sequence training of deep belief networks for speech recognition," in *INTERSPEECH*, 2010, pp. 2846–2849.

[31] J. Peng, L. Bo, and J. Xu, "Conditional Neural Fields," in *Advances in Neural Information Processing Systems*, December 2009.

[32] Y. Fujii, K. Yamamoto, and S. Nakagawa, "Deep-hidden conditional neural fields for continuous phoneme speech recognition," in *Proc. International Workshop of Statistical Machine Learning for Speech Processing (IWSML)*, 2012.

[33] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep-structured conditional random fields," *Journal of Selected Topics in Signal Processing*, vol. 4, no. 6, pp. 965–973, 2010.

[34] P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *ASRU*, 2013.

[35] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *INTERSPEECH*, 2013.

[36] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *INTERSPEECH*, 2012.

[37] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep neural networks for conversational speech transcription," in *ICASSP*, 2013.

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

[39] S. Sarawagi and W. Cohen, "Semi-Markov Conditional Random Fields for Information Extraction," in *Proc. NIPS*, 2005.

[40] H. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Press, 1993.

[41] Y.-Y. Wang and A. Acero, "Evaluation of spoken language grammar learning in the ATIS domain," in *ICASSP*, 2002.

[42] D. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg, "Expanding the scope of the ATIS task: The ATIS-3 corpus," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 43–48.

[43] G. Tur, D. Hakkani-Tur, L. Heck, and S. Parthasarathy, "Sentence simplification for spoken language understanding," in *ICASSP*, 2011, pp. 5628–5631.

[44] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernal deep convex networks and end-to-end learning for spoken language understanding," in *SLT*. IEEE, 2013, pp. 210–215.