Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig

***Abstract*—Semantic slot filling is one of the most challenging problems in spoken language understanding (SLU). In this study, we propose to use recurrent neural networks (RNNs) for this task, and present several novel architectures designed to efficiently model past and future temporal dependencies. Specifically, we implemented and compared several important RNN architectures, including Elman, Jordan and hybrid variants. To facilitate reproducibility, we implemented these networks with the publicly available Theano neural network toolkit and completed experiments on the well-known airline travel information system (ATIS) benchmark. In addition, we compared the approaches on two custom SLU data sets from the entertainment and movies domains. Our results show that the RNN-based models outperform the conditional random field (CRF) baseline by 2% in absolute error reduction on the ATIS benchmark. We improve the state-of-the-art by 0.5% in the Entertainment domain, and 6.7% for the movies domain.**

***Index Terms* — spoken language understanding, word embedding, recurrent neural network, slot filling.**

I. Introduction

The term "spoken language understanding"' (SLU) refers to the targeted understanding of human speech directed at machines [1]. The goal of such "targeted" understanding is to convert the recognition of user input, S_i, into a task-specific semantic representation of the user's intention, U_i at each turn. The dialog manager then interprets U_i and decides on the most appropriate system action, A_i, exploiting semantic context, user specific meta-information, such as geo-location and personal preferences, and other contextual information.

The semantic parsing of input utterances in SLU typically consists of three tasks: domain detection, intent determination, and slot filling. Originating from call routing systems, the domain detection and intent determination tasks are typically treated as semantic utterance classification problems [2,3,4,30,62,63]. Slot filling is typically treated as a sequence classification problem in which contiguous sequences of words are assigned semantic class labels. [5,7,31,32,33,34,40,55].

In this paper, following the success of deep learning methods for semantic utterance classification such as domain detection [30] and intent determination [13,39,50], we focus on applying deep learning methods to slot filling. Standard approaches to solving the slot filling problem include generative models, such as HMM/CFG composite models [31,5,53], hidden vector state (HVS) model [33], and discriminative or conditional models such as conditional random fields (CRFs) [6,7,32,34,40,51,54] and support vector machines (SVMs) [52]. Despite many years of research, the slot filling task in SLU is still a challenging problem, and this has motivated the recent application of a number of very successful continuous-space, neural net, and deep learning approaches, e.g. [13,15,24,30,56,64].

In light of the recent success of these methods, especially the success of RNNs in language modeling [22,23] and in some preliminary SLU experiments [15,24,30,56], in this paper we carry out an in-depth investigation of RNNs for the slot filling task of SLU. In this work, we implemented and compared several important RNN architectures, including the Elman-type networks [16], Jordan-type networks [17] and their variations. To make the results easy to reproduce and rigorously comparable, we implemented these models using the common Theano neural network toolkit [25] and evaluated

Manuscript submitted for review on XXX.

This work was partially supported by Compute Canada and Calcul Québec.
G. Mesnil is with the University of Rouen, 76821 Mont-Saint-Aignan, France and the University of Montréal, Montréal QC H3T 1J4, Canada. (corresponding author e-mail: gregoire.mesnil@umontreal.ca). Y. Dauphin and Y. Bengio are with the University of Montréal, Montréal QC H3T 1J4, Canada. K. Yao, L Deng, X. He, D. Hakkani-Yur, D. Yu, and G. Zweig are with Microsoft Research, USA. G. Tur is with Apple, USA. L. Heck is with Google, USA.
Part of this work has been done while Y. Dauphin and G. Mesnil were interns at Microsoft Research.

them on the standard ATIS (Airline Travel Information Systems) benchmark. We also compared our results to a baseline using conditional random fields (CRF). Our results show that on the ATIS task, both Elman-type networks and Jordan-type networks outperform the CRF baseline substantially, and a bi-directional Jordan-type network that takes into account both past and future dependencies among slots works best.

In the next section, we formally define the semantic utterance classification problem along with the slot filling task and present the related work. In Section III, we propose a brief review of deep learning for slot filling. Section IV more specifically describes our approach of RNN architectures for slot filling. We describe sequence level optimization and decoding methods in Section V. Experimental results are summarized and discussed in section VII.

II. Slot Filling in Spoken Language Understanding

A major task in spoken language understanding in goal-oriented human-machine conversational understanding systems is to automatically extract semantic concepts, or to fill in a set of arguments or "slots" embedded in a semantic frame, in order to achieve a goal in a human-machine dialogue.

An example sentence is provided here, with domain, intent, and slot/concept annotations illustrated, along with typical domain-independent named entities. This example follows the popular in/out/begin (IOB) representation, where *Boston* and *New York* are the departure and arrival cities specified as the slot values in the user's utterance, respectively.

Sentence	*show*	*flights*	*from*	*Boston*	*To*	*New*	*York*	*today*
Slots/Concepts	O	O	O	B-dept	O	B-arr	I-arr	B-date
Named Entity	O	O	O	B-city	O	B-city	I-city	O
Intent	*Find_Flight*							
Domain	*Airline Travel*							

ATIS utterance example IOB representation

While the concept of using semantic frames (templates) is motivated by the case frames of the artificial intelligence area, the slots are very specific to the target domain and finding values of properties from automatically recognized spoken utterances may suffer from automatic speech recognition errors and poor modeling of natural language variability in expressing the same concept. For these reasons, spoken language understanding researchers employed statistical methods. These approaches include generative models such as hidden Markov models, discriminative classification methods such as CRFs, knowledge-based methods, and probabilistic context free grammars. A detailed survey of these earlier approaches can be found in [7].

For the slot filling task, the input is the sentence consisting of a sequence of words, L, and the output is a sequence of slot/concept IDs, S, one for each word. In the statistical SLU systems, the task is often formalized as a pattern recognition problem: Given the word sequence L, the goal of SLU is to find the semantic representation of the slot sequence S that has the maximum *a posteriori* probability $P(S|L)$.

In the generative model framework, the Bayes rule is applied:

$$\hat{S} = argmax_S P(S|L) = argmax_S P(L|S)P(S)$$

The objective function of a generative model is then to maximize the joint probability $P(L|S)P(S) = P(L,S)$ given a training sample of L, and its semantic annotation, S.

The first generative model, used by both the AT&T CHRONUS system [31] and the BBN Hidden Understanding Model (HUM) [35], assumes a deterministic one-to-one correspondence between model states and the segments, i.e., there is only one segment per state, and the order of the segments follows that of the states.

As another extension, in the Hidden Vector State model the states in the Markov chain representation encode all the structure information about the tree using stacks, so the semantic tree structure (excluding words) can be reconstructed from the hidden vector state sequence. The model imposes a hard limit on the maximum depth of the stack, so the number of the states becomes finite, and the prior model becomes the Markov chain in an HMM [33].

Recently, discriminative methods have become more popular. One of the most successful approaches for slot filling is the conditional random field (CRF) [6] and its variants. Given the input word sequence $L_1^N = l_1, \dots, l_N$, the linear-chain CRF models the conditional probability of a concept/slot sequence $S_1^N = s_1, \dots, s_N$ as follows:

$$P(S_1^N|L_1^N) = \frac{1}{Z}\prod_{t=1}^{N} e^{H(s_{t-1}, s_t, l_{t-d}^{t+d})} \qquad (1)$$

where

$$H\left(s_{t-1}, s_t, l_{t-d}^{t+d}\right) = \sum_{m=1}^{M} \lambda_m h_m(s_{t-1}, s_t, l_{t-d}^{t+d}) \qquad (2)$$

and $h_m(s_{t-1}, s_t, l_{t-d}^{t+d})$ are features extracted from the current and previous states s_t and s_{t-1}, plus a window of words around the current word l_t, with a window size of $2d + 1$.

CRFs have first been used for slot filling by Raymond and Riccardi [33]. CRF models have been shown to outperform conventional generative models. Other discriminative methods such as the semantic tuple classifier based on SVMs [36] has the same main idea of semantic classification trees as used by the Chanel system [37], where local probability functions are used, i.e., each phrase is separately considered to be a slot given features. More formally,

$$P(S_1^N|L_1^N) = \prod_{t=1}^{N} P(s_t|s_1^{t-1}, L_1^N) \quad (3)$$

These methods treat the classification algorithm as a black box implementation of linear or log-linear approaches but require good feature engineering. As discussed in [57,13], one promising direction with deep learning architectures is integrating both feature design and classification into the learning procedure.

III. Deep Learning Review

In comparison to the above described techniques, deep learning uses many layers of neural networks [57]. It has made strong impacts on applications ranging from automatic speech recognition [8] to image recognition [10].

A distinguishing feature of NLP applications of deep learning is that inputs are symbols from a large vocabulary, which led the initial work on neural language modeling [26] to suggest map words to a learned distributed representation either in the input or output layers (or both), with those embeddings learned jointly with the task. Following this principle, a variety of neural net architectures and training approaches have been successfully applied [11,13,20,22,23,39,49,58,59,60,61]. Particularly, RNNs [22,23,49] are also widely used in NLP. One can represent an input symbol as a one-hot vector, i.e., containing zeros except for one component equal to one, and this weight vector is considered as a low-dimensional continuous valued vector representation of the original input, called word embedding. Critically, in this vector space, similar words that have occurred syntactically and semantically tend to be placed by the learning procedure close to each other, and relationships between words are preserved. Thus, adjusting the model parameters to increase the objective function for a training example which involves a particular word tends to improve performances for similar words in similar context, thereby greatly improving generalization and addressing the curse-of-dimensionality obstacle faced with traditional n-gram non-parametric models [26].

One way of building a deep model for slot filling is to stack several neural network layers on top of each other. This approach was taken in [27], which used deep belief networks (DBNs), and showed superior results to a CRF baseline on ATIS. The DBNs were built with a stack of Restricted Boltzmann Machines (RBMs) [12]. The RBM layers were pre-trained to initialize the weights. Then the well-known back-propagation algorithm was used to fine-tune the weights of the deep network in a discriminative fashion. Once the individual local models are trained, Viterbi decoding is carried out to find the best slot sequence given the sequence of words.

In contrast to using DBNs, we propose recurrent neural networks (RNNs). The basic RNNs used in language modeling read an input word and predict the next word. For SLU, these models are modified to take a word and possibly other features as input, and to output a slot value for each word. We will describe RNNs in detail in the following section.

IV. Recurrent Neural Networks for Slot-Filling

We provide here a description of the RNN models used for the slot filling task.

A. Word Embeddings

The main input to a RNN is a one-hot representation of the next input word. The first-layer weight matrix defines a vector of weights for each word, whose dimensionality is equal to the size of the hidden layer (Fig. 1) – typically a few hundred. This provides a continuous-space representation for each word. These neural word embeddings [26] may be trained a-priori on external data such as the Wikipedia, with a variety of models ranging from shallow neural networks [21] to convolutional neural networks [20] and RNNs [22]. Such word embeddings actually present interesting properties [23] and tend to cluster [20] when their *semantics* are similar.

While [15][24] suggest initializing the embedding vectors with unsupervised learned features and then fine-tune it on the task of interest, we found that directly learning the embedding vectors initialized from random values led to the same performance on the ATIS dataset, when using the SENNA word embeddings (http://ml.nec-labs.com/senna/). While this behavior seems very specific to ATIS, we considered extensive experiments about different unsupervised initialization techniques out of the scope of this paper. Word embeddings were initialized randomly in our experiments.

B. Context Word Window

Before considering any temporal feedback, one can start with a context word window as input for the model. It allows one to capture short-term temporal dependencies given the words surrounding the word of interest. Given d_e the dimension of the word embedding and $|V|$ the size of the vocabulary, we construct the d-context word window as the ordered concatenation of $2d+1$ word embedding vectors, i.e. d previous word followed by the word of interest and d next words, with the following dot product:

$$C_d\left(l_{i-d}^{i+d}\right) = \tilde{E}\tilde{l}_{i-d}^{i+d} \in \mathbb{R}^{d_e(2d+1)}$$

where $\tilde{E}$ corresponds to the embedding matrix $E \in \mathcal{M}_{d_e \times |V|}(\mathbb{R})$ replicated vertically $2d+1$ times and $\tilde{l}_{i-d}^{i+d} = [\tilde{l}_{i-d}, \dots, \tilde{l}_i, \dots, \tilde{l}_{i+d}]^T \in \mathbb{R}^{|V|(2d+1)}$ corresponds to the concatenation of one-hot word index vectors $\tilde{l}_i$.

$$\tilde{l}_i\,(\text{"}flight\text{"}) = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow$$

The index of word *flight* in the vocabulary

In this window approach, one might wonder how to build a d-context window for the first/last words of the sentence. We work around this border effect problem by padding the beginning and the end of sentences d times with a special token. Below, we depict an example of building a context window of size 3 around the word "from":

$$l(t) = [flights, \boldsymbol{from}, Boston]$$

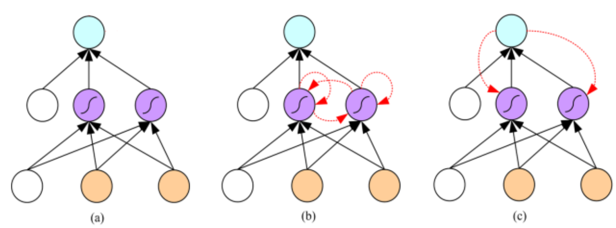
$$'from' \rightarrow w_{from} \in \mathbb{R}^{d_e}$$

$$l(t) \rightarrow C_3(t) = [l_{flights}, l_{\boldsymbol{from}}, l_{Boston}] \in \mathbb{R}^{3d_e}$$

In this example, $l(t)$ is a 3-word context window around the t-th word "from". $l_{\boldsymbol{from}}$ corresponds to the appropriate line in the embedding matrix E mapping the word "from" to its word embedding. Finally, $C_3(t)$ gives the ordered concatenated word embeddings vector for the sequence of words in $l(t)$.

C. Elman, Jordan and Hybrid architectures

As in [15], we describe here the two most common RNN architectures in the literature: the Elman [16] and Jordan [17] models. The architectures of these models are illustrated in Figure 1.



(a) Feed-forward NN; (b) Elman-RNN; (c) Jordan-RNN

Figure 1. Three types neural networks.

In contrast with classic feed-forward neural networks, the Elman neural network keeps track of the previous hidden layer states through its recurrent connections. Hence, the hidden layer at time t can be viewed as a state summarizing past inputs along with the current input. Mathematically, Elman dynamics with d_h hidden nodes at each of the H hidden layers are depicted below:

$$h^{(1)}(t) = f(U^{(1)}C_d\left(l_{t-d}^{t+d}\right) + U'^{(1)}h^{(1)}(t-1)) \quad (4)$$

$$h^{(n+1)}(t) = f(U^{(n+1)}h^{(n)}(t) + U'^{(n+1)}h^{(n+1)}(t-1)) \quad (5)$$

where we used the non-linear sigmoid function applied element wise for the hidden layer $f(x) = 1/(1+e^{-x})$ and $h^{(i)}(0) \in \mathbb{R}^{d_h}$ are parameter vectors to be learned. The superscript denotes the depth of the hidden layers and U' represents the recurrent weights connection. The posterior probabilities of the classifier for each class are then given by the softmax function applied to the hidden state:

$$P(y(t) = i|l_0^{t+d}) = \frac{e^{\sum_{j=1}^{d_h} V_{i,j}h_j^{(H)}(t)}}{\sum_{i=1}^{N} e^{\sum_{j=1}^{d_h} V_{i,j}h_j^{(H)}(t)}} \quad (6)$$

Where V correspond to the weights of the softmax top layer.

The learning part then consists of tuning the parameters $\Theta = \{E, h^{(1)}(0), U^{(1)}, U'^{(1)}, \ldots, h^{(H)}(0), U^{(H)}, U'^{(H)}, V\}$ of the RNN with N output classes. Precisely, the matrix shapes are $U^{(1)} \in \mathcal{M}_{d_h \times d_e(2d+1)}(\mathbb{R})$ $U'^{(1)}, \ldots, U^{(H)}, U'^{(H)} \in \mathcal{M}_{d_h \times d_h}(\mathbb{R})$ and $V \in \mathcal{M}_{N \times d_h}(\mathbb{R})$. For training, we use stochastic gradient descent, with the parameters being updated after computing the gradient for each one of the sentences in our training set $\mathcal{D}$, towards minimizing the negative log-likelihood. Note that a sentence is considered as a tuple of words and a tuple of slots:

$$\mathcal{L}(\Theta) = -\sum_{(S,W)\in\mathcal{D}} \sum_{t=1}^{T} \log \mathrm{P}_{\Theta}(s_t|l_0^{t+d}) \quad (7)$$

Note that the length T of each sentence can vary among the training samples and the context word window size d is a hyper-parameter.

The Jordan RNN is similar to the Elman-type network except that the recurrent connections take their input from the output posterior probabilities:

$$h(t) = f(UC_d\left(l_{t-d}^{t+d}\right) + U'\mathrm{P}\big(y(t-1)\big)) \quad (8)$$

where $U' \in \mathcal{M}_{d_h \times N}(\mathbb{R})$ and $\mathrm{P}(y(0)) \in \mathbb{R}^N$ are additional parameters to tune. As pointed out in [15], three different options can be considered for the feedback connections: (a) $\mathrm{P}(y(t-1))$, (b) a one-hot vector with an active bit for $\arg\max_i P_i(y(t-1))$ or even (c) the ground truth label for training. Empirically [15], none of these options significantly outperformed all others.

In this work, we focused on the Elman-type, Jordan-type and hybrid versions of RNNs. The hybrid version corresponds to a combination of the recurrences from the Jordan and the Elman models:

$$h(t) = f(UC_d\left(l_{t-d}^{t+d}\right) + U'\mathrm{P}\big(y(t-1)\big) + U^*h(t-1))$$

D. Forward, Backward and Bidirectional variants

In slot filling, useful information can be extracted from the future and we do not necessarily have to process the sequence online in a single forward pass. It is also possible to take into account future information with a single backward pass but still, this approach uses only partial information available. A more appealing model would consider both past and future information at the same time: it corresponds to the bi-directional Elman [18][19] or Jordan [15] RNN.

We describe the bidirectional variant only for the first layer since it is straightforward to build upper layers as we did previously for the Elman RNN. First, we define the forward $\overrightarrow{h}(t)$ and the backward $\overleftarrow{h}(t)$ hidden layers:

$$\overrightarrow{h}(t) = f(\overrightarrow{U}C_d\left(l_{t-d}^{t+d}\right) + \overrightarrow{U}'\overrightarrow{h}(t-1))$$
$$\overleftarrow{h}(t) = f(\overleftarrow{U}C_d\left(l_{t-d}^{t+d}\right) + \overleftarrow{U}'\overleftarrow{h}(t+1))$$

where $\overrightarrow{U}$ corresponds to the weights for the forward pass and $\overleftarrow{U}$ for the backward pass. The superscript U'corresponds to the recurrent weights.

The bidirectional hidden layer $\overleftrightarrow{h}(t)$ then takes as input the forward and backward hidden layers:

$$\overleftrightarrow{h}(t) = f(BC_d\left(l_{t-d}^{t+d}\right) + B'\overrightarrow{h}(t-1) + B^*\overleftarrow{h}(t+1))$$

where B are the weights for the context window input, B' projects the forward pass hidden layer of the previous time step (past), and B^* the backward hidden layer of the next time step (future).

V. SEQUENCE LEVEL OPTIMIZATION AND DECODING

The previous architectures are optimized based on a tag-by-tag likelihood as opposed to a sequence-level objective function. In common with Maximum Entropy Markov Model (MEMM) [28] models, the RNNs produce a sequence of locally-normalized output distributions, one for each word position. Thus, it can suffer from the same label bias [6] problem. To ameliorate these problems, we propose two methods: Viterbi decoding with slot language models and recurrent CRF.

A. Slot Language Models

As just mentioned, one advantage of CRF models over RNN models is that it is performing global sequence optimization using tag level features. In order to approximate this behavior, and optimize the sentence level tag sequence, we explicitly applied the Viterbi [40] algorithm. To this end, a second order Markov model has been formed, using the slot tags, $s_i \in S$ as states, where the state transition probabilities, $P_{\{LM\}}(s_i|s_j)$ are obtained using a trigram tag language model (LM). The tag level posterior probabilities obtained from the RNN were used when computing the state observation likelihoods.

$$\hat{S} = \text{argmax}_S \, P(S|L) = \text{argmax}_S \, P_{\{LM\}}(S)^{\alpha} \times P(L|S) \sim \text{argmax}_S \, P_{\{LM\}}(S)^{\alpha} \times \prod_t P_{\{RecNN\}}(s_t|l_t)/P(s_t)$$

As is often done in the speech community, when combining probabilistic models of different types, it is advantageous to weight the contributions of the language and observation models differently. We do so by introducing a tunable model combination weight, α, whose value is optimized on held-out data. For computation, we used the SRILM toolkit (http://www.speech.sri.com/projects/srilm/).

B. Recurrent CRF

The second scheme uses the objective function of a CRF, and trains RNN parameters according to this objective function. In this scheme, the whole set of model parameters, including transition probabilities and RNN parameters, are jointly trained, taking advantage of the sequence-level discrimination ability of the CRF and the feature learning ability of the RNN. Because the second scheme is a CRF with features generated from an RNN, we call it a recurrent conditional random field (R-CRF) [41,42]. The R-CRF differs from previous works that use CRFs with feed-forward neural networks [43,44] and convolutional neural networks [45], in that the R-CRF uses RNNs for feature extraction – using RNNs is motivated by its strong performances on natural language processing tasks. The R-CRF also differs from works in sequence training of DNN/HMM hybrid systems [46-48] for speech recognition, which use DNNs and HMMs, in that R-CRF uses the CRF objective and RNNs.

The R-CRF objective function is the same as Eq. (1) defined for the CRF, except that its features are from the RNN. That is, the features $h_m(s_{t-1}, s_t, l_0^{t+d})$ in the CRF objective function (2) now consist of transition feature $h_m(s_{t-1}, s_t)$ and tag-specific feature $h_m(s_t, l_{t-d}^{t+d})$ from the RNN. Note that since features are extracted from an RNN, they are sensitive to inputs back to time t=0. Eq. (2) is re-written as follows

$$H(s_{t-1}, s_t, l_{t-d}^{t+d}) = \sum_{m=1}^{M} \lambda_m h_m(s_{t-1}, s_t, l_0^{t+d}) = \sum_{p=1}^{P} \lambda_p h_p(s_{t-1}, s_t) + \sum_{q=1}^{Q} \lambda_q h_q(s_t, l_0^{t+d}) \quad (9)$$

In a CRF, $h_m(s_{t-1}, s_t, l_{t-d}^{t+d})$ is fixed and is usually a binary value of one or zero, so the only parameters to learn are the weights λ_m. In contrast, the R-CRF uses RNNs to output $h_m(s_t, l_0^{t+d})$, which itself can be tuned by exploiting error back-propagation to obtain gradients. To avoid the label-bias problem [6] that motivated CRFs, the R-CRF uses un-normalized scores from the activations before the softmax layer as features $h_m(s_t, l_0^{t+d})$. In the future, we would like to investigate using activations from other layers of RNNs.

The R-CRF has additional transition features to estimate. The transition features are actually the transition probabilities between tags. Therefore the size of this feature set is $O(N^2)$ with N the number of slots. The number of RNN parameters is $O(NH + H^2 + HV)$. Usually the relation among vocabulary size V, hidden layer size H and slot number $N$is $V \gg H > N$. Therefore, the number of additional transition features is small in comparison.

Decoding from the R-CRF uses the Viterbi algorithm. The cost introduced from computing transition scores is $O(NT)$ and T is the length of a sentence. In comparison to the computational cost of $O(N^2T)$ in the RNN, the additional cost from transition scores is small.

VI. Experimental Results

In this section we present our experimental results for the slot filling task using the proposed approaches.

A. Datasets

We used the ATIS corpus as used extensively by the SLU community, e.g. [1,7,29,38]. The original training data include 4978 utterances selected from Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora. In this work, we randomly sampled 20% of the original training data as the held-out validation set, and used the left 80% data as the model training set. The test set contains 893 utterances from the ATIS-3 Nov93 and Dec94 datasets. This dataset has 128 unique tags, as created by [34] from the original annotations. In our first set of experiments on several training methods and different directional architectures, we only used lexical features in the experiments. Then, in order to compare with other results, we incorporated additional features in the RNN architecture.

In our experiments, we preprocessed the data as in [24]. Note that authors in [13, 15, 27, 29, 38] used a different preprocessing technique, and hence their results are not directly comparable. However, the best numbers reported on ATIS by [27] are 95.3% F1-score on manual transcriptions with DBNs, using word and named entity features (in comparison to their CRF baseline of 94.4%).

As additional sets of experiments, we report results on two other custom datasets focusing on movies [39] and entertainment. Each word has been manually assigned a slot using the IOB schema as described earlier.

B. Baseline and Models

On these datasets, Conditional Random Fields (CRF) are commonly used as a baseline [7]. The input of the CRF corresponds to a binary encoding of N-grams inside a context window. For all datasets, we carefully tuned the regularization parameters of the CRF and the size of the context window using 5-fold cross-validation. Meanwhile, we also trained a feed-forward network (FFN) for slot filling, with the architecture shown in Fig 1 (a). The size of the context window for FFN is tuned using 5-fold cross-validation.

C. RNN versus Baselines and Stochastic training versus Sentence mini-batch updates

Different ways of training the models were tested. In our experiments, the stochastic version considered a single (word, label) couple at a time for each update while the sentence mini-batch processed the whole sentence before updating the parameters. Due to modern computing architectures, performing updates after each example considerably increases training time. A way to process many examples in a shorter amount of time and exploit inherent parallelism and cache mechanisms of modern computers relies on updating parameters after examining a whole mini-batch of sentences.

First, we ran 200 experiments with random sampling [14] of the hyper-parameters. The sampling choices for each hyper-parameter were for the depth, $H \in \{1,2\}$, the context size, $d \in \{3, 5, \ldots, 17\}$, the embedding dimension, $d_e \in \{50, 100\}$ and 3 different random seed values. The learning rate was sampled from a uniform distribution in the range $[0.05, 0.1]$. The embedding matrix and the weight matrices were initialized from the uniform in the range [-1,1]. We performed early-stopping over 100 epochs, keeping the parameters that gave the best performance on the held-out validation set measured after each training epoch (pass on the training set).

The F1-measure on the test set of each method was computed after the hyper-parameter search. Results are reported in Table 1. All the RNN variants and the FFN model outperform the CRF baseline. And all the RNN variants outperform the FFN model, too.

F1-score %	Elman	Jordan	Hybrid
RNN	94.98	94.29	95.06
FFN	93.32		
CRF	92.94		

Table 1. Test set F1-score of the different models after 200 runs of random sampling of the hyper-parameters. All models are trained using the stochastic gradient approach.

Then, given the best hyper-parameters found previously on the validation set, we report the average, minimum, maximum and variance of the test set accuracy over 50 additional runs by varying only the random seed. In our case, the random initialization seed impacted the way we initialized the parameters and how we shuffled the samples at each epoch. Note that for the Hybrid RNN and stochastic updates, the score obtained during hyper-parameters search corresponds to the max of the validation set score over different random seeds. The results are presented in Table 2. The observed variances from the mean are in the range of 0.3%, which is consistent with the 0.6% reported in [24] with the 95% significance level based on the binomial test. We also observe that stochastic (STO) performs better than sentence mini-batches (MB) on average. In a large-scale setting, it is always more beneficial to perform sentence mini-batches as it reduces the training complexity. On our small ATIS benchmark, it took about the same number of epochs for convergence for both training schemes STO and MB, but each epoch took longer with STO.

F1-score %		Elman	Jordan	Hybrid
STO	Min	93.23	92.91	94.19
	Max	95.04	94.31	95.06
	Avg	94.44 ±0.41	93.81 ±0.32	94.61 ±0.18
MB	Min	92.8	93.17	93.06
	Max	94.42	94.15	94.21
	Avg	93.58 ±0.30	93.72 ±0.24	93.66 ±0.30

Table 2. Measurement of the impact of using different ways of training the models and random seed on the performance.

D. Local Context Window and Bi-Directional Models

The slot-filling task is an off-line task, i.e., we have access to the whole sentence at prediction time. It should be beneficial to take advantage of all future and past available information at any time step. One way to do it consists of using bidirectional models to encode the future and past information in the input. The bidirectional approach relies on the capacity of the network to summarize the past and future history through its hidden state. Here, we compare the bidirectional approach with the local context window where the future and past information is fed as input to the model. Therefore, rather than considering a single word here, the context window allows us to encode the future and past information in the input.

We ran a set of experiments for different architectures with different context-window sizes and no local context window and compare the results to a CRF using either unigram or N-grams. Results are summarized in Table 3. Note that the CRF using no context window (e.g., using unigram features only) performs significantly worse than the CRF using a context window (e.g., using up to 9-gram features).

The absence of a context window affects the performance of the Elman RNN (-1.83%), and it considerably damages the accuracy of the Jordan RNN (-29.00%). We believe this is because the output layer is much more constrained than the hidden layer, thus making less information available through recurrence. The softmax layer defines a probability and all its components sum to 1. The components are tied together, limiting their degree of freedom. In a classic hidden layer, none of the component is tied to the others, giving the Elman hidden layer a bit more power of expression than the Jordan softmax layer. A context window provides further improvements, while the bidirectional architecture does not benefit any of the models.

F1-score	Elman	Jordan	Hybrid	CRF
Single, w/o context	93.15	65.23	93.32	69.68
BiDir, w/o context	93.46	90.31	93.16	
Single, context	94.98 (9)	94.29 (9)	95.06 (7)	92.94 (9)
Bidir, context	94.73 (5)	94.03 (9)	94.15 (7)	

Table 3. F1-score of single and Bi-Directional models with or w/o context windows. We report the best context window size hyper-parameter as the number in the round brackets.

E. *Incorporating Additional Features*

Most of the time, additional information such as look-up tables or clustering of words into categories is available. At some point, in order to obtain the best performance, we want to integrate this information in the RNN architecture. At the model level, we concatenated the Named Entity (NE) information feature as a one-hot vector feeding both to the context window input and the softmax layer [49].

F1-score	Elman	Jordan	Hybrid	CRF
Word	94.98	94.29	95.06	92.94
Word+NE	96.24	95.25	95.85	95.16

Table 4. Performance with Named Entity features.

For the ATIS dataset, we used the gazetteers of flight related entities, such as airline or airport names as named entities. In Table 4, we can observe that it yields significant performance gains for all methods, RNN and CRF included.

F. *ASR setting*

In order to show the robustness of the RNN approaches, we have also performed experiments using the automatic speech recognition (ASR) outputs of the test set. The input for SLU is the recognition hypothesis from a generic dictation ASR system and has a word error rate (WER) of 13.8%. While this is significantly higher than the best reported performances of about 5% WER [4], this provides a more challenging and realistic framework. Note that the model trained with manual transcriptions is kept the same.

F1-score	Elman	Jordan	Hybrid	CRF
Word	94.98	94.29	95.06	92.94
ASR	85.05	85.02	84.76	81.15

Table 5. Comparison between manually labeled word and ASR output.

Table 5 presents these results. As seen, the performance drops significantly for all cases, though RNN models continue to outperform the CRF baseline. We also notice that under the ASR condition, all three types of RNN perform similar to each other.

G. *Entertainment dataset*

As an additional experiment, we ran our best models on a custom dataset from the entertainment domain. Table 6 shows these results. For this dataset, the CRF outperformed RNN approaches. There are two reasons for this:

- The ATIS and Entertainment datasets are semantically very different. While the main task in ATIS is disambiguating between a departure and an arrival city/date, for the entertainment domain, the main challenge is detecting longer phrases such as movie names.
- While RNNs are powerful, the tag classification is still local, and the overall sentence tag sequence is not optimized directly as with CRFs.

However, as we shall cover in the next sections, the performance of the RNN approach can be improved using three techniques: Viterbi decoding, Dropout regularization, and fusion with the CRF framework.

H. *Slot Language Models and Decoding*

Using the Viterbi algorithm with the output probabilities of the RNN boosts the performance of the network in the Entertainment domain, while on ATIS, the improvement is much less significant. This shows the importance of modeling the slot dependencies explicitly and demonstrates the power of dynamic programing.

F1-score	Elman	Jordan	Hybrid
ATIS Word	94.98	94.29	95.06
ATIS Word +Viterbi	94.99 (+0.01)	94.25 (-0.04)	94.77 (-0.29)
ATIS Word/CRF		92.94	
ATIS ASR	85.05	85.02	84.76
ATIS ASR +Viterbi	86.16 (+1.11)	85.21 (+0.19)	85.36 (+0.6)
ATIS ASR/CRF		81.15	
Entertainment	88.67	88.70	89.04
Entertainment +Viterbi	90.19 (+1.42)	90.62 (+1.92)	90.01 (+0.97)
Entertainment +Viterbi +Dropout	-	91.14 (+2.44)	-
Entertainment /CRF		90.64	

Table 6. Comparison with Viterbi decoding with different methods on several datasets

I. Dropout regularization

While deep networks have more capacity to represent functions than CRFs, they might suffer from overfitting. Dropout [10] is a powerful way to regularize deep neural networks. It is implemented by randomly setting some of the hidden units to zero with probability *p* during training, then dividing the parameters by *1 / p* during testing. In fact, this is an efficient and approximate way of training an exponential number of networks that share parameters and then averaging their answer, much like an ensemble. We have found it further improves the performance on the Entertainment dataset, and beats the CRF by 0.5% as seen in Table 6 (i.e., 91.14% vs. 90.64%).

J. R-CRF results

We now compare the RNN and R-CRF models on the ATIS, Movies and Entertainment datasets. For this comparison, we have implemented the models with C code rather than Theano. On the ATIS data, the training features include word and named-entity information as described in [29], which aligns to the "Word+NE" line in table 4. Note that performances between RNNs in Theano and C implementations are slightly different on ATIS. The C implementation of RNNs obtained 96.29% F1 score and Theano obtained 96.24% F1 score. We used a context window of 3 for bag-of-word feature [24]. In this experiment, the RNN and R-CRF both are of the Elman type and use a 100-dimension hidden layer. On the Movies data, there are four types of features. The n-gram features are unigrams and bi-grams appeared in the training data. The regular expression features are those tokens, such as zip code and addresses, that can be defined in regular expressions. The dictionary features include domain-general knowledge sources such as US cities and domain-specific knowledge sources such as hotel names, restaurant names, etc. The context-free-grammar features are those tokens that are hard to be defined in a regular expression but have context free generation rules such as time and date. Both RNNs and CRFs are optimal for the respective systems on the ATIS and Movies domains. On the Entertainment dataset, both RNN and R-CRF used 400 hidden layer dimension and momentum of 0.6. Features include a context window of 3 as a bag-of-words. The learning rate for RNNs is 0.1 and for R-CRFs it is 0.001.

F1-score	CRF	RNN	R-CRF
ATIS Word+NE	95.16	96.29	96.46
Movies	75.50	78.20	82.21
Entertainment	90.64	88.11	88.50

Table 7. Comparison with R-CRF and RNN on ATIS, Movies, and Entertainment datasets.

As shown in Table 7, the RNNs outperform CRFs on ATIS and Movies datasets. Using the R-CRF produces an improved F1 score on ATIS. The improvement is particularly significant on Movies data, 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. On the Entertainment dataset, the RNN and R-CRF did not perform as well as the CRF. However, results confirm that the R-CRF improves over a basic RNN.

VII. Conclusions

We have proposed the use of recurrent neural networks for the SLU slot filling task, and performed a careful comparison of the standard RNN architectures, as well as hybrid, bi-directional, and CRF extensions. Similar to the previous work on application of deep learning methods for intent determination and domain detection, we find that these models have competitive performances and have improved performances over the use of CRF models. The new models set a new state-of-the-art in this area. Investigation of deep learning techniques for more complex SLU tasks, for example ones that involve hierarchical semantic frames, is part of future work.

References

[1] G. Tur and R. De Mori, "Spoken Language Understanding: Systems for Extracting Semantic Information from Speech" in *Eds. John Wiley and Sons*, 2011

[2] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization" in *Machine Learning*, vol. 39, no.2/3, pp. 135-168, 2000.

[3] P. Haffner, G. Tur and J.Wright, "Optimizing SVMs for complex call classification" in *ICASSP*, 2003.

[4] S. Yaman, L. Deng, D. Yu, Y.-Y. Wang and A. Acero, "An integrative and discriminative technique for spoken utterance classification" in *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 6, pp. 1207-1214, 2008

[5] Y. Wang, L. Deng and A. Acero, “Spoken Language Understanding — An Introduction to the Statistical Framework” in *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16-31, 2005.
[6] J. Lafferty, A. McCallum and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data” in *ICML*, 2001
[7] Y. Wang, L. Deng and A. Acero, “Semantic Frame Based Spoken Language Understanding,” in Chapter 3, Tur and De Mori “Spoken Language Understanding: Systems for Extracting Semantic Information from Speech” pp. 35-80, *Eds. John Wiley and Sons*, 2011.
[8] G. Dahl, D. Yu, L. Deng and A. Acero, “Context-dependent pre-trained deep neural networks for large vocabulary speech recognition” in *IEEE Transactions on Audio, Speech, and Language Processing*, 20 (1), 33–42, 2012.
[9] G. Mesnil, Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent, A. Courville and J. Bergstra, “Unsupervised and transfer learning challenge: a deep learning approach” in *JMLR W&CP: Proc. Unsupervised and Transfer Learning*, volume 7, 2011.
[10] A. Krizhevsky, I. Sutskever and G. Hinton, “ImageNet classification with deep convolutional neural networks” in *Advances in Neural Information Processing Systems* 25, 2012.
[11] P-S Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, Learning Deep Structured Semantic Models for Web Search using Clickthrough Data, in *ACM International Conference on Information and Knowledge Management (CIKM)*, 2013.
[12] G. Hinton, S. Osindero and Y. Teh, “A fast learning algorithm for Deep Belief Nets” in *Neural Computation* 18, 1527–1554, 2006.
[13] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, “Use of Kernel Deep Convex Networks and End-To-End Learning for Spoken Language Understanding”, in *IEEE SLT*, 2012.
[14] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization”, in *Journal of Machine Learning Research* 13 (2012) 281-305, 2012.
[15] G. Mesnil, X. He, L. Deng and Y. Bengio, “Investigation of Recurrent-Neural-Network architectures and learning methods for spoken language understanding” in *Interspeech*, 2013.
[16] J. Elman “Finding structure in time” in *Cognitive Science*, 14 (2), 1990.
[17] M. Jordan “Serial order: a parallel distributed processing approach” in *Tech. Rep no 8604,* San Diego, University of California, Institute of Computer Science, 1997.
[18] M. Schuster and K. Paliwal, “Bidirectional recurrent neural networks,” in *IEEE Transactions on Signal Processing*, 1997.
[19] A. Graves, A. Mohamed and G. Hinton, “Speech recognition with deep recurrent neural networks” in *ICASSP*, 2013.
[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, “Natural language processing (almost) from scratch” in *Journal of Machine Learning Research*, vol 12, pages 2493-2537, 2011.
[21] H. Schwenk and J-L. Gauvain, “Training neural network language models on very large corpora” in *HLT/EMNLP*, 2005.
[22] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky and S. Khudanpur, “Extensions of recurrent neural network based language model” in *ICASSP,* 2011.
[23] T. Mikolov, W. Yih and G. Zweig, “Linguistic regularities in continuous space word representations” in *NAACL-HLT,* 2013.
[24] K. Yao, G. Zweig, M-Y. Hwang, Y. Shi and D. Yu, “Recurrent neural networks for language understanding” in *Interspeech,* 2013.
[25] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio, “Theano: A CPU and GPU Math Expression Compiler,” in *Proc. Python for Scientific Computing Conference (SciPy)*, 2010.
[26] Y. Bengio, R. Ducharme and P. Vincent, “A Neural Probabilistic Language Model”, in *NIPS*, 2000.
[27] A. Deoras and R. Sarikaya, “Deep belief network based semantic tagger for spoken language understanding”, in *Interspeech*, 2013.
[28] A. McCallum, D. Freitag, and F. Pereira, “Maximum entropy Markov models for information extraction and segmentation”, in ICML. Pp. 591-598, 2000.
[29] G. Tur and D. Hakkani-Tur and L. Heck and S. Parthasarathy, “Sentence simplification for spoken language understanding”, in *ICASSP*. pp. 5628-5631, 2011.
[30] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, “Deep belief nets for natural language call-routing,” in *ICASSP*, 2011.
[31] R. Pieraccini, E. Tzoukermann, Z. Gorelov, J.-L. Gauvain, E. Levin, C.-H. Lee, and J. G. Wilpon, “A speech understanding system based on statistical representation of semantics,” in *ICASSP*, 1992.
[32] Y.-Y. Wang and A. Acero, “Discriminative models for spoken language understanding,” in *ICSLP*, 2006.
[33] Y. He and S. Young, “A data-driven spoken language understanding system,” in *IEEE ASRU*, pp. 583–588, 2003.
[34] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in *INTERSPEECH*, 2007.
[35] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, “Hidden understanding models of natural language,” in *ACL*, 1994.
[36] M. Henderson, M. Gasic, B. Thomson, P. Tsiakoulis, K. Yu, S. Young, “Discriminative Spoken Language Understanding Using Word Confusion Networks”, in *IEEE SLT*, 2012.
[37] R. Kuhn and R. De Mori, “The Application of Semantic Classification Trees to Natural Language Understanding”, in *IEEE Transactions on Pattern Analysis and Machine Intelligence* v. 17, pp. 449-460, 1995.
[38] G. Tur, D. Hakkani-Tür, and L. Heck, “What is Left to be Understood in ATIS”, in *IEEE SLT*, 2010.
[39] G. Tur, L. Deng, D. Hakkani-Tür, X. He, “Towards Deeper Understanding: Deep Convex Networks for Semantic Utterance Classification”, in *ICASSP*, 2012.
[40] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimal decoding algorithm,” in *IEEE Transactions on Information. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.
[41] K. Yao, B. Peng, and G. Zweig, D. Yu, X. Li and F. Gao, “Recurrent conditional random field for language understanding,” in *ICASSP*, pp. 4105-4009, 2014.
[42] K. Yao, B. Peng, and G. Zweig, D. Yu, X. Li and F. Gao, “Recurrent conditional random fields,” in *NIPS Deep Learning Workshop*, 2013.
[43] J. Peng, L. Bo, and J. Xu , “Conditional neural fields,” in *NIPS*, 2009.
[44] D. Yu, S. Wang and L. Deng, “Sequential labeling using deep-structured conditional random fileds,” in *Journal of Selected Topics in Signal Processing*, vol. 4, no. 6, pp. 965-973, 2010.
[45] P. Xu and R. Sarikaya, “Convolutional neural nteworks based triangular CRF for joint intent detection and slot filling,” in *ASRU*, 2013.
[46] K. Vesely, A. Ghoshal, L. Burget and D. Povey, “Sequence-discriminative training of deep neural networks,” in *INTERSPEECH*, 2013.
[47] 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.

[48] 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.
[49] T. Mikolov and G. Zweig. “Context Dependent Recurrent Neural Network Language Model” in *IEEE SLT*, 2012.
[50] Y. Dauphin, G.Tur, D. Hakkani-Tur and L. Heck. “Zero-shot learning and clustering for semantic utterance classification,” in *International Conference on Learning Representations (ICLR)*, 2013.
[51] J. Liu, S. Cyphers, P. Pasupat, I. McGraw, and J. Glass, “A conversational movie search system based on conditional random fields”, in *INTERSPEECH* 2012.
[52] T. Kudo and Y. Matsumoto, “Chunking with support vector machine”, in *ACL* 2001.
[53] M. Macherey, F. Och, and H. Ney, “Natural language understanding using statistical machine translation”, in *European Conference on Speech Communication and Technology*, pp. 2205-2208, 2001.
[54] M. Jeong and G. Lee, “Structures for spoken language understanding: a two-step approach”, in *ICASSP* 2007.
[55] V. Zue and J. Glass, “Conversational interface: advances and challenges”, in *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1166-1180, 2000.
[56] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory neural networks”, in *IEEE SLT*, 2014.
[57] Y. Bengio, *Learning deep architectures for AI*, Now Publishers, 2009.
[58] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *CIKM*, 2014
[59] R. Socher, B. Huval, C. Manning, and A. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *EMNLP-CoNLL*, pp. 1201-1211, 2012
[60] W. Yih, X. He, and C. Meek, “Semantic parsing for single-relation question answering,” in *ACL*, 2014
[61] M. Yu, T. Zhao, D. Dong, H. Tian, and D. Yu, "Compound embedding features for semi-supervised learning," in *NAACL-HLT* 2013, pp. 563–568, 2013
[62] D. Hakkani-Tur, L. Heck, and G. Tur, “Exploiting query click logs for utterance domain detection in spoken language understanding,” in *ICASSP*, 2011.
[63] L. Heck and D. Hakkani-Tur, “Exploiting the semantic web for unsupervised spoken language understanding,” in *IEEE-SLT*, 2012
[64] L. Heck and H. Huang, “Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs,” in IEEE Global Conference on Signal and Information Processing, 2014.

Grégoire Mesnil is a Ph.D. student in Computer Science at University of Montréal in Canada and University of Rouen in France. His main research interests lie within Artificial Intelligence, Machine Learning and Deep Neural Networks, creating solutions for large scale problems in Natural Language Processing and Computer Vision. He holds a M.Sc. in Machine Learning from École Normale Supérieure de Cachan and a B.Sc. in Applied Mathematics from University of Caen in France. He also received an Engineer degree in Applied Mathematics from the National Institute of Applied Sciences in Rouen.

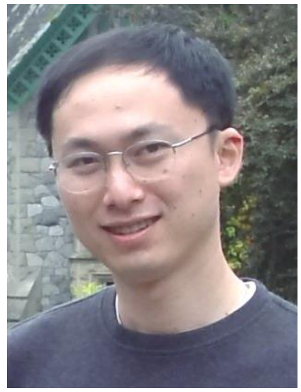
Yann Dauphin is a machine learning researcher and computer engineer. He is currently finishing my Ph.D. at University of Montréal on deep learning algorithms for large-scale problems. He received a M.S. degree in Computer Science from the University of Montréal in Canada in 2011, and a B. Eng. Degree in Computer Engineering from Ecole Polytechnique de Montreal, Canada, in 2010.

Kaisheng Yao is a senior RSDE at Microsoft Research. He received his Ph.D. degree in Electrical Engineering in a joint program of Tsinghua University, China, and Hong Kong University of Science and Technology in 2000. From 2000 to 2002, he worked as an invited researcher at Advanced Telecommunication Research Lab in Japan. From 2002 to 2004, he was a post-doc researcher at Institute for Neural Computation at University of California at San Diego. From 2004 to 2008, he was with Texas Instruments. He joined Microsoft in 2008. He has been active in both research and development areas including natural language understanding, speech recognition, machine learning and speech signal processing. He has published more than 50 papers in these areas and is the inventor/co-inventor of more than 20 granted/pending patents. At Microsoft, he has helped in shipping products such as smart watch gesture control, Bing query understanding, Xbox, and voice search. His current research and development interests are in the areas of deep learning using recurrent neural networks and its applications to natural language processing, document understanding, speech recognition and speech processing. He is a senior member of IEEE and a member of ACL.

Yoshua Bengio (CS PhD, McGill University, 1991) was post-doc with Michael Jordan at MIT and worked at AT&T Bell Labs before becoming professor at U. Montreal. He wrote two books and around 200 papers, the most cited being in the areas of deep learning, recurrent neural networks, probabilistic learning, NLP and manifold learning. Among the most cited Canadian computer scientists and one of the scientists responsible for reviving neural networks research with deep learning in 2006, he sat on editorial boards of top ML journals and of the NIPS foundation, holds a Canada Research Chair and an NSERC chair, is a Senior Fellow and program director of CIFAR and has been program/general chair for NIPS. He is driven by his quest for AI through machine learning, involving fundamental questions on learning of deep representations, the geometry of generalization in high-dimension, manifold learning, biologically inspired learning, and challenging applications of ML.

Li Deng (M'89 – SM'92 – F'04) is Partner Research Manager at the Deep Learning Technology Center of Microsoft Research in Redmond. His current interest and work are focused on research, advanced development of deep learning and machine intelligence techniques applied to large-scale data analysis and to speech/image/text multimodal information processing. In several areas of computer science and electrical engineering, he has published over 300 refereed papers and authored or co-authored 5 books including the latest 2 books on Deep Learning during 2014. He is Fellow of IEEE, Fellow of the Acoustical Society of America, and Fellow of the International Speech Communication Association. He is granted over 70 patents in acoustics/audio, speech/language technology, large-scale data analysis, and machine learning.

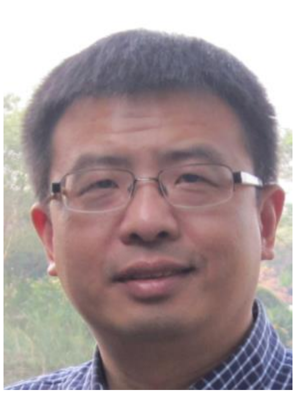
Dilek Hakkani-Tür (F'14) is a principal researcher at Microsoft Research. Prior to joining Microsoft, she was a senior researcher at the International Computer Science Institute (ICSI) speech group (2006-2010) and she was a senior technical staff member in the Voice Enabled Services Research Department at AT&T Labs-Research in Florham Park, NJ (2001-2005). She received her BSc degree from Middle East Technical University, in 1994, and MSc and PhD degrees from Bilkent University, Department of Computer Engineering, in 1996 and 2000, respectively. Her research interests include natural language and speech processing, spoken dialog systems, and machine learning for language processing. She has 38 patents that were granted and co-authored more than 150 papers in natural language and speech processing. She is the recipient of three best paper awards for her work on active learning, from IEEE Signal Processing Society, ISCA and EURASIP. She was an associate editor of IEEE Transactions on Audio, Speech and Language Processing (2005-2008), an elected member of the IEEE Speech and Language Technical Committee (2009-2012), an area editor for speech and language processing for Elsevier's Digital Signal Processing Journal and IEEE Signal Processing Letters (2011-2013). She was selected as a Fellow of the IEEE and ISCA in 2014.



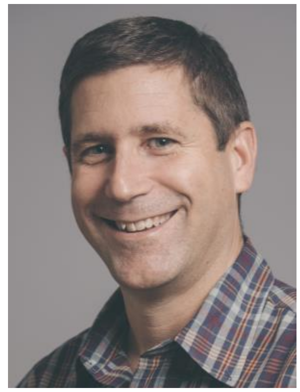
Xiaodong He (M'03 – SM'08) is a Researcher of Microsoft Research, Redmond, WA, USA. He is also an Affiliate Professor in Electrical Engineering at the University of Washington, Seattle, WA, USA. His research interests include deep learning, information retrieval, natural language understanding, machine translation, and speech recognition. Dr. He and his colleagues have developed entries that obtained No. 1 place in the 2008 NIST Machine Translation Evaluation (NIST MT) and the 2011 International Workshop on Spoken Language Translation Evaluation (IWSLT), both in Chinese-English translation, respectively. He serves as Associate Editor of IEEE Signal Processing Magazine and IEEE Signal Processing Letters, as Guest Editors of IEEE TASLP for the Special Issue on Continuous-space and related methods in natural language processing, and Area Chair of NAACL2015. He also served as GE for several IEEE Journals, and served in organizing committees and program committees of major speech and language processing conferences in the past. He is a senior member of IEEE and a member of ACL.

Larry Heck joined the Google Machine Intelligence group in 2014. From 2009-2014, he was with Microsoft. In 2009, he started the personal assistant effort in Microsoft's Speech Group. The effort established the early conversational understanding (CU) scientific foundations for Cortana, Microsoft's personal assistant launched on Windows Phone in 2014. From 2005 to 2009, he was Vice President of Search & Advertising Sciences at Yahoo!, responsible for the creation, development, and deployment of the algorithms powering Yahoo! Search, Yahoo! Sponsored Search, Yahoo! Content Match, and Yahoo! display advertising. From 1998 to 2005, he was with Nuance Communications and served as Vice President of R&D, responsible for natural language processing, speech recognition, voice authentication, and text-to-speech synthesis technology. He began his career as a researcher at the Stanford Research Institute (1992-1998), initially in the field of acoustics and later in speech research with the Speech Technology and Research (STAR) Laboratory. Dr. Heck received the PhD in Electrical Engineering from the Georgia Institute of Technology in 1991. He has published over 80 scientific papers and has granted/filed for 47 patents.

Gokhan Tur is an established computer scientist working on spoken language understanding (SLU), mainly for conversational systems. He received the Ph.D. degree in Computer Science from Bilkent University, Turkey in 2000. Between 1997 and 1999, he was a visiting scholar at the Language Technologies Institute, CMU, then the Johns Hopkins University, MD, and the Speech Technology and Research (STAR) Lab of SRI, CA. He worked at AT&T Labs - Research, NJ (2001-2006), working on pioneering conversational systems such as "How May I Help You?". He worked for the DARPA GALE and CALO projects at the STAR Lab of SRI, CA (2006-2010). He worked on building conversational understanding systems like Cortana in Microsoft (2010-2014). He is currently with the Apple Siri. Dr. Tur co-authored more than 150 papers published in journals or books and presented at conferences. He is the editor of the book entitled "Spoken Language Understanding - Systems for Extracting Semantic Information from Speech" by Wiley in 2011. Dr. Tur is also the recipient of the Speech Communication Journal Best Paper awards by ISCA for 2004-2006 and by EURASIP for 2005-2006. Dr. Tur is the organizer of the HLT-NAACL 2007 Workshop on Spoken Dialog Technologies, and the HLT-NAACL 2004 and AAAI 2005 Workshops on SLU, and the editor of the Speech Communication Issue on SLU in 2006. He is also the spoken language processing area chair for IEEE ICASSP 2007, 2008, and 2009 conferences and IEEE ASRU 2005 workshop, spoken dialog area chair for HLT-NAACL 2007 conference, and organizer of SLT 2010 workshop. Dr. Tur is a senior member of IEEE, ACL, and ISCA, was an elected member of IEEE Speech and Language Technical Committee (SLTC) for 2006-2008, and an associate editor for the IEEE Transactions on Audio, Speech, and Language Processing journal for 2010-2014, and is currently an associate editor for the IEEE Transactions on Multimedia Processing journal and member of the IEEE SPS Industrial Relations Committee.



Dong Yu (M'97, SM'06) is a principal researcher at the Microsoft speech and dialog research group. His current research interests include speech processing, robust speech recognition, discriminative training, and machine learning. He has published two books and over 140 papers in these areas and is the co-inventor of more than 50 granted/pending patents. His work on context-dependent deep neural network hidden Markov model (CD-DNN-HMM) has helped to shape the new direction on large vocabulary speech recognition research and was recognized by the IEEE SPS 2013 best paper award. Dr. Dong Yu is currently serving as a member of the IEEE Speech and Language Processing Technical Committee (2013-) and an associate editor of IEEE transactions on audio, speech, and language processing (2011-). He has served as an associate editor of IEEE signal processing magazine (2008-2011) and the lead guest editor of IEEE transactions on audio, speech, and language processing - special issue on deep learning for speech and language processing (2010-2011).



Geoffrey Zweig (F'13) is a Principal Researcher, and Manager of the Speech & Dialog Group at Microsoft Research. His research interests lie in improved algorithms for acoustic and language modeling for speech recognition, and language processing for downstream applications. Recent work has included the development of methods for conditioning recurrent neural networks on side-information for applications such as machine translation, and the use of recurrent neural network language models in first pass speech recognition. Prior to Microsoft, Dr. Zweig managed the Advanced Large Vocabulary Continuous Speech Recognition Group at IBM Research, with a focus on the DARPA EARS and GALE programs. Dr. Zweig received his PhD from the University of California at Berkeley. He is the author of over 80 papers, numerous patents, an Associate Editor of Computers Speech & Language, and is a Fellow of the IEEE.