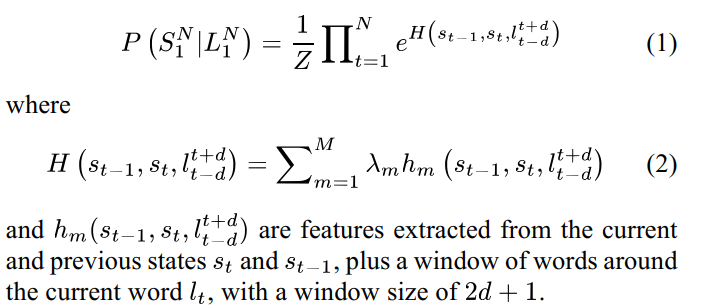
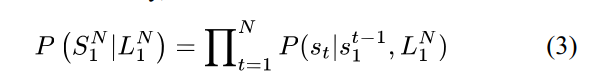
Paper 《Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding》

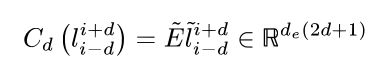
* Target: propose to use recurrent neural networks (RNNs) for this task, and present several novel architectures designed to efficiently model past and future temporal dependencies.
* Source：Slot filling is typically treated as a sequence classification problem in which contiguous sequences of words are assigned semantic class labels.
* Traditional methods：HMM/CFG composite models，hidden vector state (HVS) model，discriminative，conditional models such as conditional random fields (CRFs)，support vector machines。
* Paper contribution：we implemented and compared several important RNN architectures, including the Elman-type networks [16], Jordan-type networks [17] and their variations.
* Implement tools：common Theanoneural network toolkit [25] and evaluated them on the standard ATIS (Airline Travel Information Systems) benchmark。
* Traditional methods drawback：
  1. HMM/CFG composite models: assumes a deterministic one-to-one correspondence between model states and the segments, here is only one segment per state, and the order of the segments follows that of the states.
  2. Hidden Vector State model: the states in 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.
  3. conditional random field (CRF):



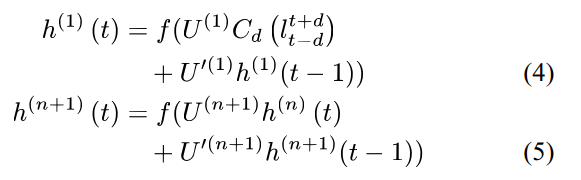
* 1. SVMs:



* 1. DBNs: it were built with a stack of Restricted Boltzmann Machines (RBMs) [12]. The RBM layers were pre-trained to initialize the weights. Then 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.
* Author work: 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.
* RNN method:
  1. Word Embedding. 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. Word embedding were initialized randomly in our experiments.
  2. 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. we construct the -context word window as the ordered concatenation of word embedding vectors.

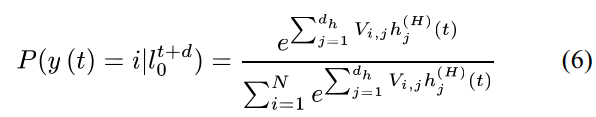


* 1. Traditional RNN.
     + **Elman model:** The hidden layer at time can be viewed as a state summarizing past inputs along with the current input.



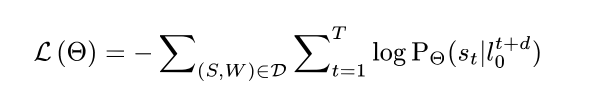
where we used the non-linear sigmoid function applied element wise for the hidden layer.

The posterior probabilities of the classifier for each class are then given by the softmax function applied to the hidden state: V is weight of softmax top layer.



Use stochastic gradient descent.

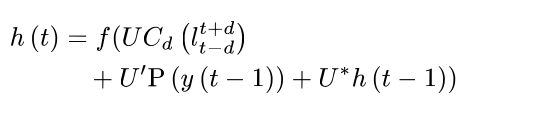
Loss function:



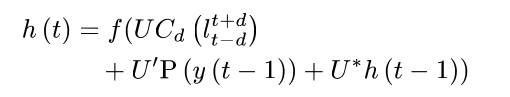
* + - **Jordan RNN:** recurrent connections take their input from the output posterior probabilities:



As pointed out in [15], three different options can be considered for the feedback connections: (a), (b) a one-hot vector with an active bit for or even (c) the ground truth label for training.

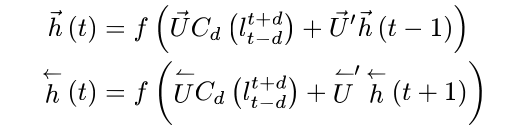


* + - **Combination：**

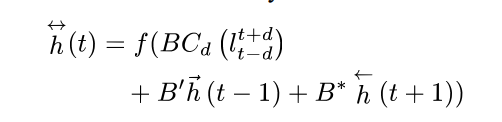


* Forward, Backward and Bidirectional Variants. A more appealing model would consider both past and future information at the same time: it corresponds to the bi-directional Elman or Jordan RNN.

Forward and the backward hidden layers:

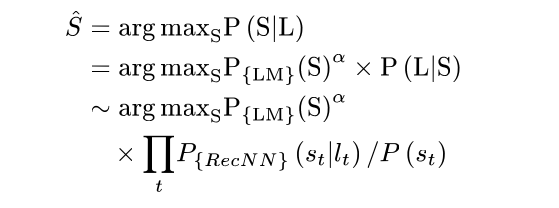


The bidirectional hidden layer then takes as input the forward and backward hidden layers:



* Sequence level optimization and decoder. Previous sequence-level objective: tag-by-tag likelihood. The RNNs produce a sequence of locally-normalized output distributions, thus, it can suffer from the same label bias problem. To ameliorate these problems, we propose two methods: Viterbi decoding with slot language models and recurrent CRF.
  1. Slot Language Models

As just mentioned, one advantage of CRF models over RNN model is that it is performing global sequence optimization using tag level features, using Viterbi [40] algorithm. To this end, a second order Markov model has been formed, using the slot tags, as states, where the state transition probabilities,  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.



* 1. Recurrent CRF. …..