IJCAI 2016

Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding

Xiaodong Zhang and **Houfeng Wang***
Institute of Computational Linguistics, Peking University
{zxdcs, wanghf}@pku.edu.cn

Abstract

Two major tasks in spoken language understanding (SLU) are intent determination (ID) and slot filling (SF). Recurrent neural networks (RNNs) have been proved effective in SF, while there is no prior work using RNNs in ID. Based on the idea that the intent and semantic slots of a sentence are correlative, we propose a joint model for both tasks. Gated recurrent unit (GRU) is used to learn the representation of each time step, by which the label of each slot is predicted. Meanwhile, a max-pooling layer is employed to capture global features of a sentence for intent classification. The representations are shared by two tasks and the model is trained by a united loss function. We conduct experiments on two datasets, and the experimental results demonstrate that our model outperforms the state-of-the-art approaches on both tasks.

1 Introduction

Spoken language understanding (SLU) in human/machine spoken dialog systems aims to automatically identify the intent of the user as expressed in natural language and extract associated arguments or slots towards achieving a goal [Tur *et al.*, 2011]. In recent years, with its widespread application in many areas, e.g. automatic customer service, automatic question answering, voice assistants, etc., SLU has become a hot point in research communities. Typically, a SLU system first transcribes users' voice into text by an automatic speech recognizer (ASR), or the input is directly text typed by users. Then the intent of the user and associated arguments is identified. The system can take the next proper action according to the extracted information to help the users achieve their demands.

An example sentence, "Show flights from Boston to New York today", is demonstrated in Table 1 with In/Out/Begin (IOB) representation. This sentence is derived from airline travel information system (ATIS) corpus [Price, 1990], the most widely used dataset in SLU area. The domain of the sentence is airline travel and the intent is to find a flight. The word "Boston" is labelled as the departure city and "New York" as arrival city. There are also named entity labels. The "Boston" and "New York" are labelled as cities, with which the slot label are easier to predict. The domain and intent determination are usually treated as a semantic utterance classification (SUC) problem and the slot filling as a sequence labelling problem. Since categories of intents are more fine-grained than domains, we focus on intent determination in this work.

Domain	Airline Travel	
Intent	Find Flight	
Sentence	**Slot Label**	**Named Entity**
show	O	O
flights	O	O
from	O	O
Boston	B-dept	B-city
to	O	O
New	B-arr	B-city
York	I-arr	I-city
today	O	O

Table 1: An example utterance from the ATIS dataset

In recent years, RNNs have demonstrated their effectiveness in language modelling [Mikolov *et al.*, 2011]. The state-of-the-art method for SF task is also based on RNNs [Yao *et al.*, 2014]. Nevertheless, RNNs have not been explored in the ID task, let alone a joint model for the two tasks. A joint model is worthwhile for two reasons. Firstly, the two tasks are usually both necessary in a SLU system. Secondly, the information of one task can be utilized in the other task to promote each other and a joint prediction can be made. For example, if the intent of a sentence is to find a flight, it is likely to contain the departure and arrival cities, and vice versa. Based on this idea, we propose a joint model for the two tasks. The input text can be viewed as a sequence. GRU is used to learn the representation of each time step in the sequence. On one hand, these representations are used for predicting slot labels. On the other hand, a global representation of the sequence for intent classification is learned by a max-pooling of these representations. Therefore, the representations learned by GRU are shared by two tasks, and with a joint loss function, the two tasks can interact with each other through the shared representations. Experimental results demonstrate that the joint

*Corresponding author

model outperforms separate models for each task. Furthermore, our model outperforms the state-of-the-art methods for ID and SF on two datasets. Another possible way of combining the two tasks is a pipeline scheme. It first classifies the intent of an utterance and then uses the extra intent information to help slot filling. This scheme is inferior to our joint model because the direction of information sharing is one-way and it suffers from error propagation problem.

2 Related Work

In the 1990s, the SLU research emerged from some call classification systems [Gorin *et al.*, 1997] and the ATIS project. Due to the informality of spoken language and recognition errors, the data-driven statistical approach has become the mainstream. For ID, word n-grams are typically used as features with generic entities, such as dates, locations. Because of the very large dimension of the input space, large margin classifiers such as SVM [Haffner *et al.*, 2003] and Adaboost [Schapire and Singer, 2000] were found to be effective for ID. For SF, a lot of work was based on CRF because of its strong ability on sequence labelling [Raymond and Riccardi, 2007]. Apart from lexical and named entity features, some researchers tried to use syntactic information. Hakkani-Tür *et al.* [2005] presented an approach populating heterogeneous features from syntactic and semantic graphs of utterance for call classification. Moschitti *et al.* [2007] employed syntactic features for slot filling via syntactic tree kernels with SVM. To solve the difficulty in processing complex sentences, Tur *et al.* [2011] proposed a dependency parsing based sentence simplification approach that extracts a set of keywords and uses those in addition to entire utterances for completing SLU tasks. Recently, there has been some work on joint ID and SF. Jeong and Geunbae Lee [2008] proposed triangular CRF, which coupled an additional random variable for intent on top of a standard CRF. Mairesse *et al.* [2009] presented an efficient technique that learned discriminative semantic concept classifiers whose output was used to recursively construct a semantic tree, resulting in both slot and intent labels. Although great successes have been made, these methods require good feature engineering and even additional semantic resources. A promising direction is deep learning, which integrates both feature design and classification into the learning procedure.

Recently, various deep learning models have been explored in SLU. The initial try is deep belief networks (DBNs), which have been used in call routing classification [Sarikaya *et al.*, 2011] and slot filling [Deoras and Sarikaya, 2013]. Tur *et al.* [2012] used deep convex networks (DCNs) for domain classification and produced higher accuracy than a boosting-based classifier. RNNs have shown excellent performance on the SF task [Mesnil *et al.*, 2013; 2015] and outperform traditional models, such as CRF. Yao *et al.* [2014] improved RNNs by using transition features and the sequence-level optimization criterion of CRF to explicitly model dependencies of output labels. As for joint work on ID and SF, Xu and Sarikaya [2013] improved the triangular CRF model by using convolutional neural networks (CNNs) to extract features automatically. Guo *et al.* [2014] adapted recursive neural networks (RecNNs) for joint training of ID and SF. To use RecNNs for sequence labelling, tri-path vector was proposed to capture contextual information. In this paper, we propose a RNNs based joint model for SLU. To the best of our knowledge, this is the first work using RNNs for joint ID and SF.

3 Model

The structure of our model is shown in Figure 1. The input of the network is text S of an utterance, which is a sequence of words $w_1, ..., w_T$, and T is the length of the utterance. The network consists of two kinds of output, i.e. the predicted slot label sequence $\hat{l^s}$ and predicted intent label $\hat{l^u}$. Next we give a detailed description of our model.

3.1 Embeddings

As an alternative to traditional representations, such as one-hot representation, word embeddings are suitable for serving as the input of neural networks. These embeddings are usually trained in an unsupervised way on a large corpus and then fine-tuned during supervised training process.

Mesnil *et al.* [2015] found that a context word window could improve the performance of RNNs on SF. Following their work, we also use a context window as the input of the recurrent layer. With each word mapped to an embedding vector, the d-context word window x_t^d, which considers the d previous words and d next words of the current word w_t, is defined as the ordered concatenation of the $2d + 1$ word embedding vectors. Formally,

$$x_t^d = [e(w_{t-d}), ..., e(w_t), ..., e(w_{t+d})] \tag{1}$$

where $e(w_t)$ is the embedding of w_t, and the size of the word window is $2d + 1$.

Named entity is an important kind of feature for SLU. To utilize these features, we associate each named entity (including a special label representing non-entity) with an embedding, which is initialized randomly according to a uniform distribution on the interval $[0, 1]$ and is fine-tuned during the training process. The context named entity window is defined like the word window. With named entity embeddings, the input of the recurrent layer at time step t is represented as:

$$\begin{aligned} x_t^d = [&e(w_{t-d}), ..., e(w_t), ..., e(w_{t+d}), \\ &e'(n_{t-c}), ..., e'(n_t), ..., e'(n_{t+c})] \end{aligned} \tag{2}$$

where $e'(n_t)$ is the embedding of the named entity n_t, and the size of named entity window is $2c + 1$.

3.2 Recurrent Hidden Layers

As an extension of conventional feed-forward neural networks, RNNs can handle the variable-length sequence by using a recurrent hidden state to take into account the influence of past states.

Traditionally, given a sequence $x = (x_1, x_2, ..., x_T)$, the recurrent hidden state is calculated by

$$h_t = \tanh(Wx_t + Uh_{t-1}) \tag{3}$$

where h_t is the hidden state at time t, W and U are respectively transformation matrices of the input and the previous hidden state.

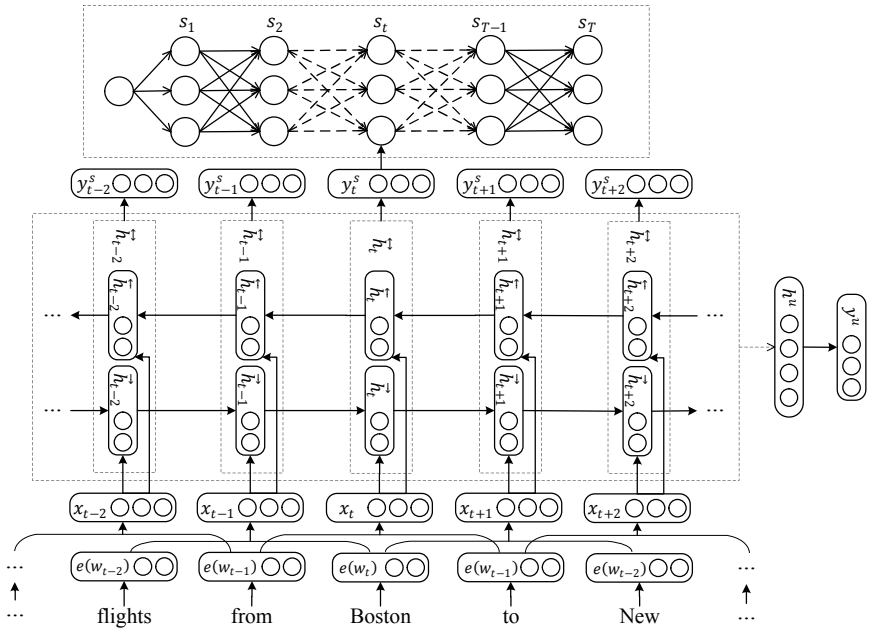


Figure 1: The structure of our model

It was difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish or explode. Therefore, some more sophisticated activation functions with gating units were designed. Two representative improvements are long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and recently proposed gated recurrent unit (GRU) [Cho *et al.*, 2014]. We use GRU in this work, because it performs comparably to LSTM in our experiments, but has less parameters.

GRU Definition: The hidden state h_t of GRU at time t is calculated by

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{4}$$

$$\widetilde{h}_t = \tanh(W x_t + r_t \odot (U h_{t-1})) \tag{5}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{6}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{7}$$

where x_t is the input at time t, r and z are reset gate and update gate respectively, σ is sigmoid function, W and U are transformation matrices, and $\odot$ denotes element-wise product of two vectors. For simplification, the above equations are abbreviated with $h_t = GRU(x_t, h_{t-1})$.

For sequence labelling, it is beneficial to consider both past and future information at the same time. Consequently, we use a bidirectional variant of GRU to learn the representations at each time step.

First, we define the forward $\overrightarrow{h}_t$ and the backward $\overleftarrow{h}_t$ hidden layers:

$$\overrightarrow{h}_t = \overrightarrow{GRU}(x_t, \overrightarrow{h}_{t-1}) \tag{8}$$

$$\overleftarrow{h}_t = \overleftarrow{GRU}(x_t, \overleftarrow{h}_{t+1}) \tag{9}$$

where $\rightarrow$ and $\leftarrow$ represent the forward and backward pass respectively.

The bidirectional hidden state $\overleftrightarrow{h}_t$ at time t is defined as the concatenation of the forward and backward hidden states.

$$\overleftrightarrow{h}_t = [\overrightarrow{h}_t, \overleftarrow{h}_t] \tag{10}$$

3.3 Task Specific Layers

The bidirectional hidden states are shared by two tasks. On one hand, the hidden states capture the features at each time step, so they are directly used for predicting slot labels. On the other hand, we use a max-pooling layer to acquire the representation of the whole sequence h^u. The global representation is defined as:

$$h^u = \max_{i=1}^{T} \overleftrightarrow{h}_t \tag{11}$$

where the $\max$ function is an element-wise function, and T is the number of words in the utterance. The pooling layer converts texts with variable length into a fixed-length vector, with which the information throughout the entire text can be captured.

The last part is the output layer. The softmax function is applied to the representations with linear transformation to give the probability distribution y_t^s over the t-th slot labels and the distribution y^u over the intent labels. Formally,

$$y_t^s = softmax(W^s \overleftrightarrow{h}_t + b^s) \tag{12}$$

$$y^u = softmax(W^u h^u + b^u) \tag{13}$$

where W^s and W^u are transformation matrices for SF and ID respectively, b^s and b^u are bias vectors.

The labels of slots are inferred at sentence level as in [Chen *et al.*, 2015]. A transition score A_{ij} is introduced to measure

the probability of jumping from label i to label j. For a label sequence $l_{1:T}$, a sentence-level score is given by the sum of label transition scores and predicted scores at each time step:

$$s(l_{1:T}, \theta) = \sum_{t=1}^{T} (A_{l_{t-1} l_t} + y_t^s(l_t)) \quad (14)$$

where $y_t^s(l_t)$ is the predicted probability of label l_t at time step t, which is calculated by Equation 12. A special circumstance is that when $t = 1$, l_0 is involved but not exists. Therefore we add one more label "BOS" to represent l_0. Finally, the predicted slot label sequence $\hat{l^s}$ is the one with the highest score in all possible label sequences L:

$$\hat{l^s} = \underset{l^s \in L}{\operatorname{argmax}}\, s(l^s, \theta) \quad (15)$$

3.4 Training

The parameters θ in our network include:

$$\theta = \{E, E', \overrightarrow{GRU_\theta}, \overleftarrow{GRU_\theta}, W^s, b^s, W^u, b^u, A\} \quad (16)$$

Specifically, $E \in \mathbb{R}^{|e| \times |V|}$, $E' \in \mathbb{R}^{|e'| \times |V'|}$ are respectively word and entity embeddings. $\overrightarrow{GRU_\theta}$ are parameters of the forward GRU, which include the transformation matrices $\overrightarrow{W}, \overrightarrow{W_r}, \overrightarrow{W_z} \in \mathbb{R}^{|x| \times |h|}$, $\overrightarrow{U}, \overrightarrow{U_r}, \overrightarrow{U_z} \in \mathbb{R}^{|h| \times |h|}$ and initial hidden states $\overrightarrow{h}_0 \in \mathbb{R}^{|h|}$. $\overleftarrow{GRU_\theta}$ are parameters of the backward GRU, which is similar to the forward GRU, and we use separate parameters for the forward and backward passes. $W^s \in \mathbb{R}^{2|h| \times L^s}$,$W^u \in \mathbb{R}^{2|h| \times L^u}$ are transformation matrices for two tasks and $b^s \in \mathbb{R}^{L^s}$,$b^u \in \mathbb{R}^{L^u}$ are the bias vectors. $A \in \mathbb{R}^{(L^s+1) \times (L^s+1)}$ is transition score between labels. $|V|, |V'|$ are respectively the number of words and named entities in the vocabulary, and $|e|$,$|e'|$ are dimensions of embeddings of word and named entity. $|h|$ is the dimension of the recurrent hidden units. L^s, L^u are respectively the number of labels of slot and intent. $|x|$ is the dimension of the input of recurrent layer, i.e. $|x| = (2d+1)|e|$ with only lexical features, and $|x| = (2d+1)|e| + (2c+1)|e'|$ with both lexical and named entity features.

Next we define loss function for our networks. We use S to denote the text of an utterance, l^s and l^u to denote the ground truth label of slot and intent.

The loss function for intent is a cross-entropy cost function.

$$\mathcal{L}^u(\theta) = -\log y^u(l^u) \quad (17)$$

The loss function for slot is calculated by a Max-Margin criterion. The structured margin loss $\Delta(l^s, \hat{l^s})$ for ground truth slot label sequence l^s and predicted sequence $\hat{l^s}$ is defined as:

$$\Delta(l^s, \hat{l^s}) = \sum_{t=1}^{T} 1\{l_t^s \neq \hat{l_t^s}\} \quad (18)$$

The loss function of a slot label sequence is defined as:

$$\mathcal{L}^s(\theta) = \max(0, s(\hat{l^s}, \theta) + \Delta(l^s, \hat{l^s}) - s(l^s, \theta)) \quad (19)$$

The training target of the network is minimizing a united loss function:

$$\mathcal{L}(\theta) = \Sigma_{(l^s, l^u, S) \in \mathcal{D}} \left(\alpha \mathcal{L}^s(\theta) + \mathcal{L}^u(\theta)\right) \quad (20)$$

where $\mathcal{D}$ is the dataset, α is a weight factor to adjust the attention paid to two tasks.

Through the united loss function, the shared representations learned by GRUs can consider two tasks jointly. Furthermore, the correlations of the two tasks can be learned and promote each other.

4 Experiments

4.1 Dataset

In order to evaluate our proposed model, we conducted experiments on two datasets. The first set is widely used ATIS dataset, and the second dataset consists of question collected from Baidu Knows[1], the most famous Chinese QA community. We refer to the second dataset as Chinese Question Understanding Dataset (CQUD).

ATIS dataset: The ATIS corpus [Price, 1990] is the most commonly used dataset for SLU research. There are some variants of the dataset. In this paper, we use the ATIS corpus used in [Tur *et al.*, 2010]. The dataset consists of sentences of people making flight reservations. There are 4978 sentences for training and 893 sentences for testing. The numbers of distinct intents and slots are 18 and 63 respectively.

As shown in Table 1, the ATIS dataset also has extra named entity features marked via table lookup, including entities such as city, airline, dates, etc., which nearly determine the slot label. Researchers have different opinions on whether to use these features or not. Some researchers think these features are hand-crafted and not generally available in open domain so that they only use lexical features [Guo *et al.*, 2014; Mesnil *et al.*, 2013]. To make a comprehensive comparison, we give our results with and without named entity features.

CQUD dataset: Although there are some other datasets for SLU, e.g. Cortana Data [Guo *et al.*, 2014] and Bing Query Understanding Dataset [Yao *et al.*, 2014], they are non-public and all English. We intend to conduct experiments on a dataset having some differences with the ATIS. Therefore, we collected questions from Baidu Knows and manually label them. Finally, 3286 questions were filtered and labelled. They are from four domains: Flights, Weather, Express, and Other. The "Other" domain consists of questions that do not belong to the previous three domains. There are totally 43 kinds of intents and 20 kinds of slots. Although questions in Baidu Knows are typed by users, they are far different from formal written language. The style of the questions is informal and filled with colloquialism. They are similar to spoken language except that they are not spoken but typed.

The CQUD dataset distinguishes from the ATIS dataset at three aspects. Firstly, the languages are different so that we can conduct experiments for different languages. Secondly, the domains of CQUD are more diverse than ATIS, and specifically CQUD includes sentences labelled with "Other", which is necessary for real-world applications. Thirdly, the intents and slots in CQUD are more diverse and informal than ATIS, which increases the difficulties for the two tasks. For example, slots of places in ATIS are basically limited to cities in America, while in CQUD the slots are more informal, like "the summit of Changbai Mountain".

[1] http://zhidao.baidu.com/

4.2 Evaluation Metrics

We use accuracy as evaluation metric for ID task. Some utterances in ATIS have more than one intent label. As in previous work [Tur *et al.*, 2010], an utterance is counted as a correct classification if any ground truth label is predicted.

F1-score is used as evaluation metric for SF task. A slot is considered to be correct if its range and type are correct. The F1-score is calculated using CoNLL evaluation script[2].

4.3 Baselines

We compare our model against the following baselines:

SVM: Raymond and Riccardi [2007] used heuristic combinations of forward-moving and backward-moving sequential SVMs classifiers for slot filling.

CRF: A CRF baseline was demonstrated in [Mesnil *et al.*, 2015]. The input is the n-grams in a context window.

RNN: Mesnil *et al.* [2013] employed RNNs for slot filling and the experimental results were updated in [Mesnil *et al.*, 2015].

R-CRF: Yao *et al.* [2014] proposed R-CRF as an improvement to RNNs. It is the state-of-the-art method for SF. Here we use the score reported in [Mesnil *et al.*, 2015] as we use the same dataset.

Boosting: Tur *et al.* [2010] employed AdaBoost.MH algorithm for intent determination. Word n-grams are used as features.

Sentence simplification: Tur *et al.* [2011] parsed the sentence to extract keywords, which served as additional information for SLU task. AdaBoost.MH is the classifier for ID, and CRF for SF. It is the state-of-the-art method for ID.

RecNN: Guo *et al.* [2014] adapted recursive neural networks for joint training of ID and SF. To improve the result on SF, the Viterbi algorithm was applied to optimize the sentence level tag sequence.

4.4 Training Details

We preprocess the datasets as follows. For ATIS, as in [Yao *et al.*, 2013], to deal with unseen words in the test set, we marked all words with only one occurrence in the training data as $\langle \mathrm{UNK} \rangle$ and used this label to represent those unseen words in the test set. We also converted sequences of numbers to the string DIGIT, e.g. "1990" is converted to "DIGIT*4". For CQUD, we use Chinese character rather than word as a basic unit for slot filling. It is because there is no separator between Chinese words and some wrong word segmentations bring inevitable errors for slot filling. Syntactical features are not employed for SF task on CQUD dataset because a parser needs word segmentation first, while for baselines that are set for ID and not joint, such as sentence simplification, we perform word segmentation and syntactical features can be used. Besides, named entity features are not employed on CQUD because we did not label named entities in the dataset.

For our joint model, the average score of the two tasks on the held-out data is used as target to select hyper-parameters. For ATIS, we took out 15% of the training data as hold-out data to determine the hyper-parameters. After optimizing these hyper-parameters, the model is trained on all of the training data and tested on the test data. For CQUD, we report the scores of 5-fold cross validation. GloVe [Pennington *et al.*, 2014] is used to train word embeddings as initializations. For ATIS, the training corpora are Wikipedia and Gigaword. For CQUD, the training corpus is 3 millions unlabelled question collected from Baidu Knows. The dimensions of word embeddings are both 200, and for ATIS the dimension of named entity embeddings is 40. For context window, we set $d = 1$ and $c = 1$. The dimensions of the forward and backward recurrent hidden states are both 300. Model parameters are updated using stochastic gradient descent (SGD). The training set is shuffled at each epoch. The learning rate is initialized to be 0.01 and is updated dynamically by AdaDelta method.

[2]http://www.cnts.ua.ac.be/conll2000/chunking/output.html

4.5 Results and Analysis

The results are demonstrated in Table 2. The second column lists the features used by each method. W, N and S denote lexical, named entity and syntactical features respectively except that W means Chinese character features on CQUD.

We can see that CRF outperforms SVM on SF, showing that with the sequence-level optimization, CRF is suitable for the sequence labelling task. Furthermore, RNN beats CRF because of the ability of capturing long-term dependencies. To model label transfer and acquire the global optimum of the whole sequence, R-CRF combined RNN and CRF, and achieved the state-of-the-art performance on SF task. In fact, the inference of slot labels at sentence level used in our model is similar to R-CRF. For ID task, the state-of-the-art approach is sentence simplification, which uses a dependency parser to extract keywords of a sentence. This method is not a joint work, two individual classifiers are used for the two tasks. RecNN uses the syntactical information in a deep learning scheme. However, the results are worse than the sentence simplification. We think this may be because the scale of the dataset is small such that human-written syntactical features perform better. Nevertheless, Guo *et al.* [2014] compared the joint model and separate models by RecNN and the results demonstrated the effectiveness of the joint model.

On ATIS dataset, our joint model outperforms the state-of-the-art of ID, yielding an absolute improvement of the F1-score of 1.34%, corresponding to a relative error reduction of 44%. As to SF, our model outperforms the state-of-the-art method by 0.43% (relative 12%). The absolute improvement may be not very high, because this dataset have been studied for more than ten years and the score of the state-of-the-art method is very high. Nevertheless, we still achieve obvious relative error reduction. We also observed that the named entity features help the ID only a little, but contribute to the SF a lot, because of the high relevancy of named entities and slot labels. Our method outperforms the previous joint work RecNN significantly, thanks to the powerful ability of recurrent neural networks for modelling sequence and the united loss function that can adjust the weights of two tasks.

The scores on CQUD dataset are lower than ATIS. It is likely because of the higher difficulty of CQUD. As mentioned in Section 4.1, the intents in CQUD are more diverse than ATIS, and the expression of slots are more informal. Besides, the input is a sequence of Chinese characters rather

Model	Features	ATIS		CQUD	
		Intent	Slot	Intent	Slot
SVM [Raymond and Riccardi, 2007]	W	—	89.76	—	81.32
CRF [Mesnil *et al.*, 2015]	W	—	92.94	—	83.40
CRF [Mesnil *et al.*, 2015]	W+N	—	95.16	—	—
RNN [Mesnil *et al.*, 2015]	W	—	95.06	—	85.63
RNN [Mesnil *et al.*, 2015]	W+N	—	96.24	—	—
R-CRF [Yao *et al.*, 2014]	W	—	—	—	**85.88**
R-CRF [Yao *et al.*, 2014]	W+N	—	**96.46**	—	—
Boosting [Tur *et al.*, 2010]	W	95.50	—	93.54	—
Sentence simplification [Tur *et al.*, 2011]	W+S	**96.98**	95.00	**94.46**	—
RecNN [Guo *et al.*, 2014]	W+S	95.40	93.22	—	—
RecNN+Viterbi [Guo *et al.*, 2014]	W+S	95.40	93.96	—	—
Our model	W	98.10	95.49	**96.05**	**87.12**
Our model	W+N	**98.32**	**96.89**	—	—

Table 2: Comparison with previous approaches

than words. Nevertheless, the improvements are consistent in CQUD. Our model outperforms the state-of-the-art methods by 1.59% for ID and 1.24% for SF.

4.6 Joint Model vs Separate Models

First, we give the definitions of joint model and separate models. The joint model is our proposed model in Figure 1. The separate model is similar to the joint model except that there is only one task. For ID, there is only the shared layers and ID specific layers without SF specific layers. It is in the same way for SF. We also implement a pipeline method. First a RNN is trained for ID, and then the predicted intent is used as addition feature to train another RNN for SF.

The speed advantage of the joint model is self-evident, because only one model is needed to train and test. The shared part of the model is only calculated one time for the two tasks. For quantitative analysis, we ran programs of the joint model and separate models using same parameter settings and hardware. On ATIS dataset, the time for training one epoch using joint model is 124 seconds, while the sum of time using separate models is 212 seconds.

Next we compare the performance of the joint model and separate models. In these experiments, only lexical features are used. Here a new concept, joint and with one task oriented, is introduced. In the joint model, we can pay different attentions to the two tasks. This is achieved by adjusting the weight factor α in Equation 20 and using score of one task as target to select hyper-parameters. Larger α means that more attention is paid to SF. The results are listed in Table 3.

The joint model outperforms separate models for two tasks, showing that the joint training is effective. The correlations of two tasks are learned by our joint model and contribute to the two tasks. Because of the two-way information sharing and supervision, our joint model outperforms the one-way pipeline method. Note that if we set one task as oriented in the joint model, higher performance can be acquired for it comparing to treating two tasks equally. This brings flexibility to have tendency to one task if high score is required for it or even only one task is needed in a real application.

In ATIS, α is set to 1.6 for equal model, 1.6 for ID oriented and 1.8 for SF oriented. In CQUD, α is set to 1.5, 2.0 and 1.8 respectively for three models. Intuitively, the performance of one task gets better with higher weight on it. It is not always true in our experiments, which may be because too large weight for one task leads to too quick convergence such that parameters are not well tuned for that task.

Model	ATIS		CQUD	
	Intent	Slot	Intent	Slot
ID only	97.53	—	95.34	—
SF only	—	95.14	—	85.78
Pipeline	97.53	95.41	95.34	86.96
Joint (equal)	98.10	95.49	96.05	87.12
Joint (ID oriented)	98.10	95.49	**96.35**	86.63
Joint (SF oriented)	97.87	**95.61**	95.93	**87.23**

Table 3: Comparison of joint model and separate model

5 Conclusion and Future Work

In this paper we have introduced recurrent neural networks for joint intent determination and slot filling, which are two major tasks in spoken language understanding. Bidirectional GRUs are used to learn the sequence representations shared by two tasks. A global representation is acquired by a max-pooling of the shared representations to predict the label of intent. The labels of slots are predicted by the shared representations and are further inferred at sequence level. Through a united loss function and shared representations, the correlations of the two tasks are learned so as to promote each other. We conducted experiments on two datasets. The joint model demonstrates advantages over separate models and outperforms the state-of-the-art approaches on both tasks.

In future works, we plan to improve our model by using syntactic information. Furthermore, our CQUD dataset is still small-scale for the application of deep learning methods. We would like increase the scale of our dataset, which can be useful for SLU and QA research.

Acknowledgements

Our work is supported by National High Technology Research and Development Program of China (863 Program) (No.2015AA015402), National Natural Science Foundation of China (No.61370117 & No.61433015). We thank Yang Liu for helping revising the paper.

References

[Chen *et al.*, 2015] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, 2015.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103, 2014.

[Deoras and Sarikaya, 2013] Anoop Deoras and Ruhi Sarikaya. Deep belief network based semantic taggers for spoken language understanding. In *INTERSPEECH*, pages 2713–2717, 2013.

[Gorin *et al.*, 1997] Allen L Gorin, Giuseppe Riccardi, and Jeremy H Wright. How may i help you? *Speech communication*, 23(1):113–127, 1997.

[Guo *et al.*, 2014] Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. Joint semantic utterance classification and slot filling with recursive neural networks. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 554–559. IEEE, 2014.

[Haffner *et al.*, 2003] Patrick Haffner, Gokhan Tur, and Jerry H Wright. Optimizing svms for complex call classification. In *ICASSP*, volume 1, pages I–632. IEEE, 2003.

[Hakkani-Tür *et al.*, 2005] D Hakkani-Tür, G Tur, and A Chotimongkol. Using syntactic and semantic graphs for call classification. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, 2005.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Jeong and Geunbae Lee, 2008] Minwoo Jeong and G Geunbae Lee. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302, 2008.

[Mairesse *et al.*, 2009] François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Spoken language understanding from unaligned data using discriminative classification models. In *ICASSP*, pages 4749–4752. IEEE, 2009.

[Mesnil *et al.*, 2013] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775, 2013.

[Mesnil *et al.*, 2015] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2015.

[Mikolov *et al.*, 2011] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE, 2011.

[Moschitti *et al.*, 2007] Alessandro Moschitti, Giuseppe Riccardi, and Christian Raymond. Spoken language understanding with kernels for syntactic/semantic structures. In *IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 183–188. IEEE, 2007.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *EMNLP*, 12:1532–1543, 2014.

[Price, 1990] Patti Price. Evaluation of spoken language systems: The atis domain. In *Proceedings of the Third DARPA Speech and Natural Language Workshop*, pages 91–95. Morgan Kaufmann, 1990.

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

[Sarikaya *et al.*, 2011] Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. Deep belief nets for natural language call-routing. In *ICASSP*, pages 5680–5683. IEEE, 2011.

[Schapire and Singer, 2000] Robert E Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2):135–168, 2000.

[Tur *et al.*, 2010] Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. What is left to be understood in atis? In *IEEE Spoken Language Technology Workshop (SLT)*, pages 19–24. IEEE, 2010.

[Tur *et al.*, 2011] Gokhan Tur, Dilek Hakkani-Tür, Larry Heck, and Sarangarajan Parthasarathy. Sentence simplification for spoken language understanding. In *ICASSP*, pages 5628–5631. IEEE, 2011.

[Tur *et al.*, 2012] Gokhan Tur, Li Deng, Dilek Hakkani-Tür, and Xiaodong He. Towards deeper understanding: deep convex networks for semantic utterance classification. In *ICASSP*, pages 5045–5048. IEEE, 2012.

[Xu and Sarikaya, 2013] Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *ASRU*, pages 78–83. IEEE, 2013.

[Yao *et al.*, 2013] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. Recurrent neural networks for language understanding. In *INTERSPEECH*, pages 2524–2528, 2013.

[Yao *et al.*, 2014] Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. Recurrent conditional random field for language understanding. In *ICASSP*, pages 4077–4081. IEEE, 2014.

Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Adversarial Active Learning for Sequence Labeling and Generation

Yue Deng[1], **KaWai Chen**[2], **Yilin Shen**[1], **Hongxia Jin**[1]
[1] AI Center, Samsung Research America, Mountain View, CA, USA
[2] Department of Electrical and Computer Engineering, University of California, San Diego
{y1.deng, yilin.shen, hongxia.jin}@samsung.com, w0chen@eng.ucsd.edu

Abstract

We introduce an active learning framework for general sequence learning tasks including sequence labeling and generation. Most existing active learning algorithms mainly rely on an uncertainty measure derived from the probabilistic classifier for query sample selection. However, such approaches suffer from two shortcomings in the context of sequence learning including 1) cold start problem and 2) label sampling dilemma. To overcome these shortcomings, we propose a deep-learning-based active learning framework to directly identify query samples from the perspective of adversarial learning. Our approach intends to offer labeling priorities for sequences whose information content are least covered by existing labeled data. We verify our sequence-based active learning approach on two tasks including sequence labeling and sequence generation.

1 Introduction

Active learning (AL) is a traditional approach to solve supervised learning problems without sufficient labels. While there have been many existing AL works proposed for classification problems [Settles, 2010; Scheffer *et al.*, 2001; Deng *et al.*, 2013], active learning algorithms for sequences are still not widely discussed. With the growing interests in AI research, many newly emerged problems are exactly defined in the scope of sequence learning including image captioning [Vinyals *et al.*, 2015], machine translation [Luong *et al.*,] and natural language understanding [Dong and Lapata, 2016]. Compared with classification tasks that only need one label for a sample, sequence learning tasks require a series of token-level labels for a whole sequence. Precise annotations for sequences are not only labor-consuming but may also require very specific domain knowledge [Dong and Lapata, 2016] that are not easily accomplished by crowd-sourcing workers. This apparent difficulty in sequence labeling exactly motivates our explorations of more effective active learning approaches for sequences.

Existing active learning strategies mainly rely on some uncertainty measures derived from a classifier for query sample selection [Cohn *et al.*, 1994; Settles, 2010]. These uncertainty measures can be defined from various perspectives including probabilistic confidence [Culotta and McCallum, 2005], margin value [Scheffer *et al.*, 2001], entropy [Deng *et al.*, 2016], fisher information [Sutton and McCallum, 2006; Bao *et al.*, 2017] and a score voted by several base models [Seung *et al.*, 1992; Deng *et al.*, 2017b]. While these active learning algorithms work well for data classification tasks, they are unfortunately not easily extended to solving sequence learning problems due to the complexity of the label space. Consider a label sequence with p tokens and each token can belong to k possible classes, then there are k^p possible combinations of the label sequence. This complexity can grow exponentially with the length of the output.

We consider two major challenges faced by existing active learning approaches in handling sequence learning tasks: 1) cold start problem and 2) label-sampling dilemma. The first cold-start challenge is mainly due to the complexity of the learning system for structured prediction. Unlike classification tasks that just need a simple probabilistic classifier, the predictor for sequences are configured within a complex recurrent structure, e.g. a LSTM. Training a structured predictor with very limited labeled sequence can easily lead to a biased estimation. If the predictor itself is seriously biased, how can we trust the uncertain measure derived from it? This cold start problem easily happens during the initial steps of active learning when there are only insufficient labeled samples in hand. The second label sampling dilemma is ascribed to the inability of the full enumeration of all possible sequence labels. In detail, when calculating an uncertainty score e.g., the entropy, for a sequence, all possible label combinations should be taken into account (see Eq. 3) that can become impossible when the output sequences are too long. Therefore, only approximated uncertainty measures can be used as a surrogate for sequence-based active learning.

To overcome the aforementioned limitations, we propose a new active learning framework for sequences inspired by adversarial learning. Our approach alleviates the demands on the structured predictor for query sample selection. The proposed adversarial active learning framework incorporates a neural network to explicitly assert each sample's informativeness with regard to labeled data. The easily-induced active score avoids heavy computations in sampling the whole label space and can improve the active learning efficiency by more than 100 times on some large datasets.

2 Preliminaries

2.1 Active Learning for Sequences

In this part, we review some existing active learning approaches for sequence learning [Settles and Craven, 2008]. The first widely used uncertainty measure defined from the probabilistic classifier is the least confidence (LC) score:

$$\psi_{LC}(x^U) = 1 - P(y^*|x^U), \tag{1}$$

where y^* is the most likely label sequence and x^U is an unlabeled sample. While this measure is intuitive, the calculation of the most likely label sequence is not easy. It requires the dynamic programming to find the Viterbi parse. Similarly, the margin term can also be used to define the uncertainty for an unknown sample:

$$\psi_M(x^U) = P((y_2^*|x^U)), -P(y_1^*|x^U), \tag{2}$$

where y_1^* and y_2^* are the first and second best label sequences, respectively.

The sequence entropy term is also widely used as an uncertainty score [Settles and Craven, 2008]:

$$\psi_{EN}(x^U) = -\sum_{y^p} P(y^p|x^U) \log P(y^p|x^U) \tag{3}$$

where y^p ranges over all possible label sequences for input x^U. We have also noted that the number of possible labeling grows exponentially with the desired output sequence length. For the ease of computation, we follow previous work to employ an approximated N-best sequence entropy (NSE) [Kim *et al.*, 2006],

$$\psi_{NSE}(x^U) = -\sum_{y^p \in \mathcal{N}} P(y^p|x^U) \log P(y^p|x^U) \tag{4}$$

$\mathcal{N} = \{(y^1)^p, ..., (y^N)^p\}$ corresponds to N most likely parses. These N most likely parses can obtained through beam search [Koehn, 2004]. The labeling priority should be given to samples with high entropy (corresponding to low confidence).

While the definition of these terms are different, they are all closely related to the structured predictor. The calculations of these uncertainty scores are also not trivial and may require some algorithmic explorations such as the dynamic programming or the beam search. When the candidate samples' quantity is large, the calculation of such complexity uncertainty measures can take a quite long while in scoring all individual samples from the data pool. These obvious shortcomings motivate us to design more efficient and advanced active learning strategies for sequence learning. In this work, we propose such a desired framework from adversarial learning [Deng *et al.*, 2017c]. The active learning strategy in our model is not related to the structured output predictor and hence can conduct query samples scoring in light on very large dataset.

2.2 Encoder-decoder Framework

Before going to the details about our active learning model, we will first review the prevalent encoder-decoder framework for sequence learning. This generic encoder-decoder model serves as the basic building block of our active learning system. We denote $(x^L, y^L) \sim (X^L, Y^L)$ as a pair of labeled sample, where x^L is the input data that can be of any type including images, speeches and texts depending on different learning tasks; and y^L is the targeted output sequence composed of p tokens $y^L = \{y_1^L ... y_p^L\}$. A feature encoder $M()$ is established to map the input x^L to a latent representation $z^L = M(x^L)$. $M()$ can be a convolution neural network for image data or a recurrent neural network for speeches and texts. Then, a decoder $C()$ adopts z^L as a conditional input and sequentially predicts each token in y^P:

$$\begin{aligned} &P(y^p = \{y_1^P ... y_q^P\}|x) \\ &= P(y_1^p|z^L = M(x^L)) \prod_{t=1}^{T} P(y_t^p|y_1^p ... y_{t-1}^p, z^L), \end{aligned} \tag{5}$$

The above generative probability can be well modeled by a recurrent neural network, e.g., a LSTM [Deng *et al.*, 2017a]. The encoded latent representation z^L is used as the 'starting key' at step zero. Then, it sequentially outputs each token y_t based on the tth step's input and the memory vector maintained by the recurrent neural network [Sutskever *et al.*, 2014]. The training loss of this sequence learning part is obtained by counting the differences between the predicted sequence y^P and the ground truth labels y^L:

$$\mathcal{L}_s(X^L, Y^L) = \sum_{(x^L, y^L) \sim (X^L, Y^L)} L(y^L, y^P) \tag{6}$$

We noted that both y^L is the labeled sequence; and predicted sequence y^P is generated by a function of x^L (see Eq.5) ; L can be arbitrary losses defined over two sequences such as the prevalent cross-entropy. Here, we just briefly introduced this encoder-decoder framework and interested readers are refereed to [Sutskever *et al.*, 2014; Xu *et al.*, 2015] for details.

3 Adversarial Active Learning for Sequences

3.1 ALISE Model

It is conceivable that sequence learning model requires a huge amount of labeled data for robust training. We hence consider developing an active learning algorithm to facilitate the whole labeling process. In our approach, we consider defining an active score based on the informativeness of an unlabeled sample x^U with respect to all labeled samples X^L:

$$s(x^U) = sim(x^U, X^L) \tag{7}$$

where X^L is the set containing all labeled samples and $sim(\cdot, \cdot)$ defines a similarity score between a point x^U and a training set X^L composed of labeled samples. The score in Eq.7 helps to rank unlabeled samples based on their inherent informativeness similarity to existing labeled data. A small similarity score implies the certain unlabeled sample is not related to any labeled samples in training set and vice versa. The labeling priority is offered to samples with low similarity scores.

We take image captioning as an intuitive instance to explain the rational behind our active scoring approach. In the training set, most images and their corresponding descriptions are about human sports such as skating, running and swimming. Then, we have access to two extra unlabeled images that are respectively related to "swimming" and "a plate

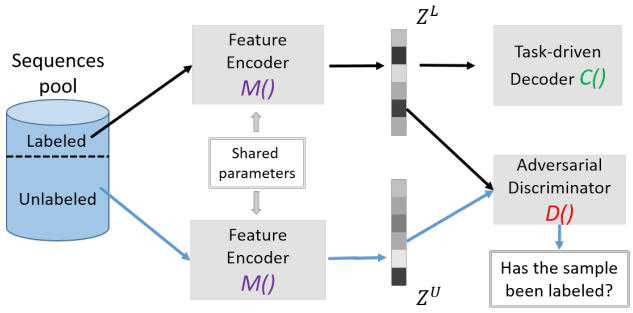


Figure 1: An overview of Adversarial Active Learning for sequences (ALISE). The black and blue arrows respectively indicate flows for labeled and unlabeled samples.

of food". With those annotated images in hand, which unlabeled image should be labeled first? There is no doubt that the image about 'food' should be sent out for captioning by human. This is because there are already some swimming images in the existing training pool and adding another similar image may not offer too much 'new' knowledge to the learning system. On the contrary, the image about food is not covered in existing training set and its captions can bring more valuable complementary information.

However, the problem still remains in how to quantitatively evaluate the informativeness similarity between an unlabeled sample with the labeled data pool. For sequence data, the similarity calculation itself is a difficult problem due to the variations in sequence lengths. Existing approaches mainly play kernel tricks to map the original sequences into a kernel space with fixed-length feature representations [Settles and Craven, 2008]. However, the selection of an appropriate kernel requires sophisticated domain knowledge and the best kernel can vary from task to task. For instance, the best kernels for Chinese and French sentences are obviously not the same.

In this work, we propose a new adversarial active learning model for sequences (ALISE). In our ALISE, we consider designing a discriminator network $D(\cdot)$ to directly outputs the informativeness similarity scores for unlabeled samples. In Fig. 1, we pass both a labeled sample x^L and an unlabeled sample x^u trough the same feature encoder M (shared parameters), then we get $z^L = M(x^L)$ (latent representation for labeled data) and $z^U = M(x^U)$ (latent representation for unlabeled data). These two latent representations are further fed into the discriminator network (D), which is trained to classify whether the certain data is sampled from labeled or unlabeled data pool. The output of the D is a sigmoid function that indicates how likely the certain sample is from the labeled pool.

The learning objectives of M and D are involved in an adversarial process. From the aspect of encoder M, it intends to map all data to a latent space where both labeled and unlabeled data can follow very similar probabilistic distributions. In the most ideal scenario that if z^L and z^U follow exactly the same generative probability, then the decoder C trained with z^L should also seamlessly work on latent representations z^U obtained from unlabeled sample x^U. Therefore, the encoder $M()$ intends to fool the discriminator to regard all latent representations (z^L and z^U) as already labeled. Mathematically, it encourages the discriminator D to output a score 1 for both z^L and z^U. The corresponding loss is modeled by the cross-entropy in the first two terms of the following equation:

$$\begin{aligned}\min \mathcal{L}_M = &-\mathbb{E}_{x^L \sim X^L}[(\log D(M(x^L)] \\ &-\mathbb{E}_{x^U \sim X^U}[\log D(M(x^U))] + \lambda \mathcal{L}_s(X^L, Y^L),\end{aligned} \tag{8}$$

In addition to the cross-entropy loss defined on the discriminator side, the above equation also takes the supervised loss in Eq.6 into consideration, i.e., in the third term. In all, the learning objectives of the feature encoder M are concluded as two-fold: 1) fool the discriminator and 2) improve the fitting quality on labeled data. These two learning objectives are balanced by a hyper-parameter λ.

The learning objective of the discriminator D goes against to the objective in Eq. 8. The discriminator is trained to correctly assign $z^L = M(x^L)$ to labeled category ($D(Z^L) = 1$) and $z^U = M(x^U)$ to unlabeled class ($D(Z^U) = 0$). The corresponding learning objective of D is also defined by the cross-entropy:

$$\begin{aligned}\min \mathcal{L}_D = &-\mathbb{E}_{x^L \sim X^L}[(\log D(M(x^L)] \\ &-\mathbb{E}_{x^U \sim X^U}[\log(1 - D(M(x^U)))]\end{aligned} \tag{9}$$

This adversarial discriminator D exactly serves the purpose of distribution comparisons between two set of samples. In GAN work [Deng *et al.*, 2017c], it is indicated that the adversarial discriminator implicitly compares the generative distributions between real data and fake data. Here, we borrowed the same adversarial learning idea to compare the distributions between labeled and unlabeled samples. In GAN (resp. our ALISE model), the discriminator outputs low scores for those fake (resp. unlabeled) samples that are mostly not similar to real images (resp. labeled data). Therefore, the score from this discriminator already serves as an informativeness similarity score that could be directly used for Eq.7. The feature encoder M, sequence decoder C and adversarial discriminator D can all be trained in an alternative manner by iteratively optimizing the objectives in Eq.8 and Eq.9. We have detailed the learning steps in Algorithm 1.

3.2 Active Scoring

After well training, we can pass all unlabeled samples through M and D to get their corresponding score by ALISE framework, i.e.,

$$s(x^U) = D(M(x^U)) \in (0, 1), \forall x^U \in X^U \tag{10}$$

The score $s = 1$ (resp. $s = 0$) means the information content of the certain unlabeled sample is most (resp. least) covered by the existing labeled data. Apparently, those samples with lowest scores should be sent out for labeling because they carry most valuable information in complementary to the current labeled data.

It is noted that our ALISE approach does not rely on the structured predictor (i.e. the decoder C) for uncertainty measure calculation. However, we can still consider incorporating existing predictor-dependent uncertainty scores into our

Algorithm 1: ALISE Learning

Input : A data pool composed of labeled and unlabeled data $X = \{X^L, X^U\}$; X^L are paired with sequence label $Y^L = \{y_1^L ... y_p^L\}$;

Initialization: Initialize parameters in encoder network M and decoder network C by training an encoder-decoder framework with available training samples (X^L, Y^L); Initialize parameters in the discriminator network D randomly.

for *epoch=1...K* **do**
  for *all mini-batches* $(x^L, y^L) \sim (X^L, Y^L)$ *and* $x^U \sim X^U$ **do**
    Minimize the loss $\mathcal{L}_M$ in Eq.8 to update parameters in the encoder network M and decoder network C
    Minimize the loss $\mathcal{L}_D$ in Eq.9 and update parameters in discriminator network D
  end
end

Output : The well trained M, C and D

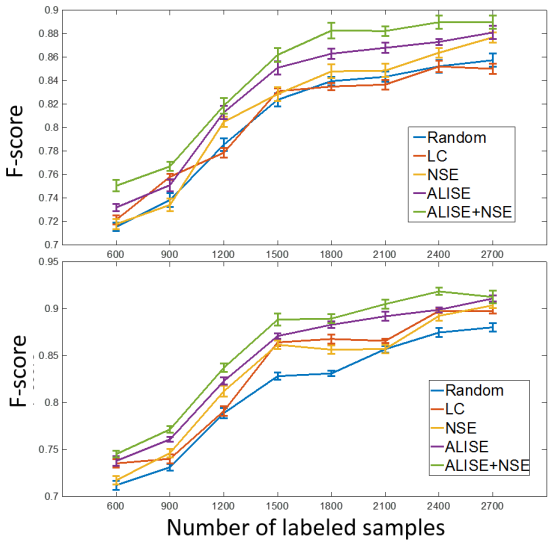


Figure 2: Slot filling F-score of different active learning approaches.

framework. Because the ALISE framework has already been built with a probabilistic decoder $C()$, then the calculations of uncertainty measures from it is natural and convenient. In such a combinational setting, we can first select K top samples selected by the adversarial discriminator. Then, within these K samples, we further calculate their sequence-based uncertainty scores $\psi(x^U)$ (e.g. the sequence entropy) as introduced in Section 2.1. The top k samples with highest uncertainty scores are selected as query samples for labeling. These candidate query samples are mainly determined by the adversarial discriminator and the probabilistic decoder only provides auxiliary information for fine-grained selection. Moreover, the complexity for sequence-based uncertainty measure computations have also been reduced. This is because the uncertainty measure is only required to be computed on K candidate samples selected by ALISE rather than the whole pool of unlabeled samples.

While there are some early works that also use the 'buzzwords' adversarial active learning, they are totally different from our ALISE. First, the work in [Zhu and Bento, 2017] used the GAN model to generate fake images and then labeling those fake images to augment training set. ALISE does not generate any fake sample and just borrows the adversarial learning objective for sample scoring. The work in [Miller *et al.*, 2014] is totally none related to adversarial learning. It just uses traditional active learning approach to solve the adversarial attract problem in security domain.

4 Experiments

In this part, we investigate the performances of ALISE on two sequence learning tasks including slot filling and image captioning.

4.1 Slot Filling

Slot filling is a basic component of spoken language understanding. It can be viewed as a sequence labeling problem, where both the input and output label sequences are of the same length. This part of experiments were mainly conducted on the ATIS (Airline Travel Information Systems) dataset [Hemphill *et al.*, 1990]. We obtained ATIS text corpus that was used in [Liu and Lane, 2016] and [Deoras and Sarikaya, 2013] for active learning. For instance, an input sentence in ATIS $x^L = \{$business, class, fare,from, SF, to, LA$\}$ can be parsed as a label sequence $y^L = \{$B-class-type, I-class-type, O, O, B-from-loc, B-to-loc$\}$. This studied dataset contains 5138 utterances with annotated slot labels.

We follow the same implementation in [Liu and Lane, 2016] to use a bi-directional LSTM as the encoder network M in Fig.1. This bidirectional LSTM read the input sentence in both forward and backward directions and their hidden states at each step were concatenated as the long vector. We choose 128 for word embedding layer and 64 hidden states for the encoder LSTM. To this end, we have obtained 128 dimensions for the latent representation z. The decoder C in Fig.1 is implemented by either a standard LSTM decoder [Sutskever *et al.*, 2014] or a more advanced attention model [Liu and Lane, 2016]. Both of them are widely used in existing literatures. The adversarial network D is configured by three dense-connected layers with 128 (input layer), 64 (intermediate layer) and 1 (output layer) units, respectively. The output layer is further connected with a sigmoid function for probabilistic conversion. We use $relu$ activation among all other layers. Each token of the output sequence is coded as a one-hot vector with the hot entry indicating the underlying cateogory of the token. The whole deep learning system was trained by ADAM [Kingma and Ba, 2014]. Among all labeled training samples, we further randomly select 10% of them as validation samples. The whole training process is terminated when the loss on the validation set does not decrease or when the optimization reaches 100 epochs.

We consider comparing our ALISE approach with existing sequence-based active learning algorithms. The competitors include random sampling, least confidence score (see Eq.2),

N-best sequence entropy (NSE, see Eq.4). Moreover, we further consider the combinational scoring approach as introduced in Section 3.2 that we combine both ALISE scores and NSE scores for query sample selection. To make fair comparisons, the number of optimal decoding parses (N) is chosen as five for both NSE approach and our ALISE+NSE approach. In active sequence learning, we randomly select 2130 sequences as testing samples. The remaining 3000 sequences are used for model training and active labeling.

In detail, among these 3000 data, $p = 300$ samples are randomly chosen as initial labeled data. Then, we train the ALISE model with these $p = 300$ samples and conduct active learning based on the remaining $3000 - p$ non-testing samples. $k = 300$ top samples returned by different active learning methods are selected for label query. After labeling, these k samples will be merged with the existing p labeled samples as the new labeled pool. The ALISE and other active learning models will be trained with this new labeled set and the trained model will be used to select another k unlabeled samples for the next round. Such query sample selection, labeling approach and model retraining processes will be iteratively conducted. We only report results until we have get 2700 training samples because it is the limitation for active selection. When all 3000 points are used, there are no distinctions among different active learning algorithms. The active learning results with different training sample sizes are reported in Fig.2, where both the LSTM and attention model are respectively used as the decoder C for sequence label prediction. In the figure, we do the random splitting process for five times and the average F-score with standard deviations are reported. Here, we choose the F-score as the accuracy indicator because it is widely used in existing works [Liu and Lane, 2016][Deoras and Sarikaya, 2013]

From the results, we have observed that our ALISE model and its combinational extension (ALISE+NSE) both outperform existing sequence learning approaches. When the labeled number size is small, the improvements of two ALISE models are more significant. The ALISE+NSE model further improves the performances of ALISE. However, when the number of sample sizes is relatively large, the differences between ALISE and ALISE+NSE are minor. However, these two ALISE methods are still better than other sequence learning approaches. Meanwhile, we have observed that using attention model as the sequence decoder is much better than the LSTM model.

4.2 Image Captioning

We further apply ALISE model for the sequence generation task of image captioning. In this task, the input data is an image and the corresponding label is a caption sentence describing the content of the input image. We follow the same configuration and parameter settings in the work [Xu *et al.*, 2015] to implement the encoder-decoder learning framework. The structure of the adversarial discriminator in ALISE is kept the same as in the slot filling experiment. This part of active learning experiments are mainly conducted on MSCOCO dataset [Lin *et al.*, 2014], which consists of 82,783 images for training, 40,504 for validation, and 40,775 for testing. We noted that each image in MSCOCO dataset is paired with 5

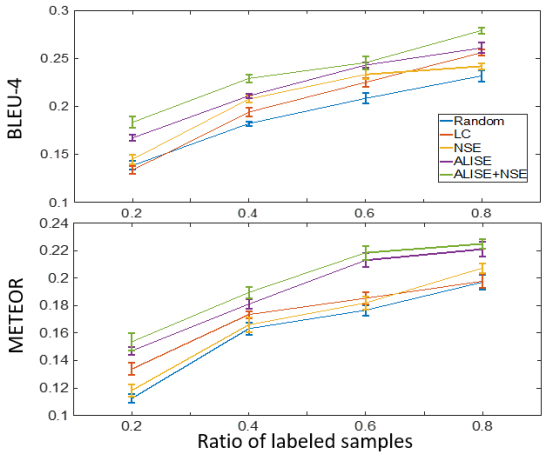


Figure 3: Image captioning results by active learning.

.

ground truth captions. In our active learning setting, the query sample selection is mainly conducted at the image level. It means that if one image has been selected for labeling, its corresponding five ground-truth captions are all accessible. We follow Karpathy et al.[Karpathy and Fei-Fei, 2015] to preporcess the sentences, where all the words are converted to lower-case, and all non-alphanumeric characters are discards. We discarded all words that appear less than twice in all captions.

We consider all 82,783 training set as the basic data pool for active learning and query selection. We increase the labeled samples' rate from 0.2 to 0.8 with 0.2 as an incremental step. Among the first $0.2\times$ 82,783 samples, half of them are randomly chosen as the initial labeled set and the remaining are selected by different active learning algorithms. The active selection and learning processes are iteratively conducted by adding k =$0.2\times$ 82,783 new labeled samples to the labeled pool in each round. These extra k samples are selected by different active learning algorithms. The performances of ALISE are compared with other active learning approaches in Fig.3. For result evaluations, we follow existing works to report BLEU-4 and METEOR as the accuracy indicator. These two accuracy measures can be easily calculated by the MSCOCO API. We repeat the aforementioned active learning process for 5 times with average and standard deviation reported in Fig.3. We have observed from quantitative evaluation that ALISE models (the original ALISE and ALISE+NSE) beat all existing active learning models based on these two scores. Meanwhile, the performance of ALISE can be further enhanced by combining NSE score as auxiliary indicator (ALISE+NSE).

To better understand differences among various active learning approaches, we provide some captioning results as intuitive instances in Fig.4. All these image captioning models are trained with 80% data points from the training set. Nevertheless, these same amount of training samples are selected by different active learning methods. In the figure, we provide the captioning results by NSE, ALISE and

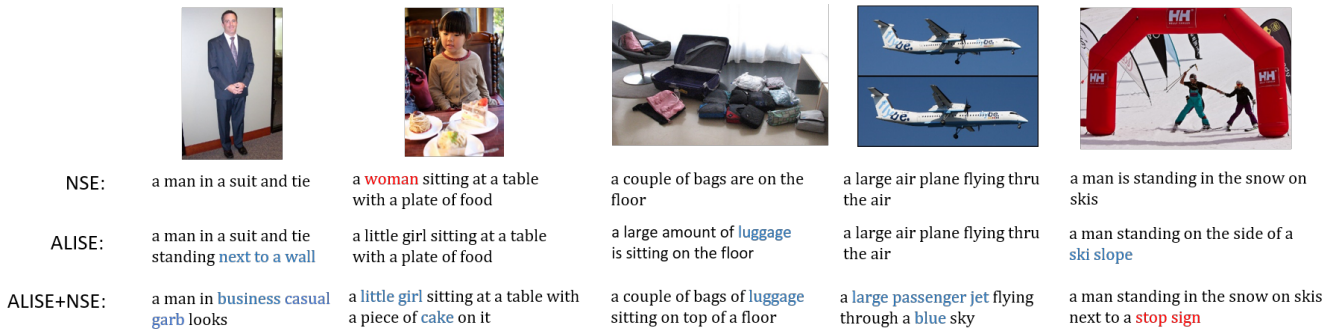


Figure 4: Image captioning results in the active learning setting by ALISE, ALISE+NSE and NSE-based approaches. The novel plausible descriptions are annotated with blue color while wrong descriptions are colored in red.

.

ALISE+NSE because these three algorithms outperform others in Fig.3. From these intuitive results, we have observed that the two ALISE models tends to use more complex sentence structures for image descriptions. Moreover, these two ALISE models can cover more details about the visual information. It is mainly because the ALISE approach can actively build up a training set covering diverse information. However, there is still small chance that the ALISE model over-explains the image with wrongly recognized objects. As shown in the rightest sub-figure in Fig.4, ALISE+NSE model mistakenly describes the red finishing line as a stop sign. However, the captioning results of ALISE model are still much better than the NSE approach in producing precise captioning sentences in a more natural manner.

4.3 Computational Complexity

We also reported the computational costs of ALISE. We have observed that the computational costs of ALISE are almost the same as the original baseline model. In detail, we respectively report the training costs on the slot filling and image captioning tasks for references. The ALISE training in slot filling task (with 2700 samples) can be accomplished in just 74 seconds with 16 GPUs (Tesla K80) parallelized in optimization. The original Attention-based encoder-decoder slot-filling model costs 53 seconds when trained with the same amount of data [Liu and Lane, 2016]. For the image captioning task, the baseline attention model [Xu *et al.*, 2015] and ALISE respectively spends an average of 2.7 hours and 3.2 hours in total on 66,000 images. From these two tasks, the training cost of ALISE is not that different than the corresponding baseline encoder-decoder model. This is because ALISE has only introduced an auxiliary adversarial discriminator in the model and this discriminator neural network exhibits very simple structures (just a multi-layer neural network with a 64 nodes intermediate layer and a 1 node output layer.).

However, the active learning complexity of different methods can vary significantly, especially when the candidate unlabeled pool size is large. We report the query sample selection costs on the aforementioned two datasets, that include 2,400 (i.e., the first data point in Fig.2) and 66,000 (i.e., the first data point in Fig.3) candidate samples on the slot filling and image captioning datasets, respectively. The corresponding costs of different algorithms are reported in **??**. Here, we omit the complexity of random sampling because it can be finished in real time.

	Slot Filling	Captioning
LC	173s	2182s
NSE	245s	3956 s
ALISE	1.7s	6.9 s
ALISE+NSE	11.3s	67.4 s

Table 1: The active selection costs for different algorithms

We have found that ALISE methods are much faster than existing sequence-based active learning approaches. This is because the calculation of the LC and NSE scores require the Viterbi parsing and beam search over the whole output space. Therefore, their costs are significant higher when the sample size is large (as in the image captioning dataset). However, the scoring mechanism in ALISE method just requires passing all samples through a trained neural network (i.e. the adversarial discriminator D in Fig.1). Therefore, the corresponding active scoring cost can be minor. The ALISE+NSE can also be efficiently implemented because it just performs N-best sequence entropy calculations on a selected number of samples filtered by ALISE model. Therefore, its computational costs are a bit higher than ALISE but are still far more less than other approaches.

5 Discussions

We introduced a sequence-based active learning model ALISE from the perspective of adversarial learning. It conducts query sample selections based on a well trained discriminator. Therefore, ALISE is much more efficient than existing predictor-dependent active learning approaches. Moreover, our model accomplishes both the tasks of active learning and sequence learning into a joint framework that is end-to-end trainable. Therefore, it is seamlessly applied to diverse learning tasks across different domains. Experimental verifications show that ALISE can greatly improve the performances and speed of existing models in the early active

learning stages with insufficient training samples.

References

[Bao *et al.*, 2017] Feng Bao, Yue Deng, Mulong Du, Zhiquan Ren, Qingzhao Zhang, Yanyu Zhao, Jinli Suo, Zhengdong Zhang, Meilin Wang, and Qionghai Dai. Probabilistic natural mapping of gene-level tests for genome-wide association studies. *Briefings in bioinformatics*, page bbx002, 2017.

[Cohn *et al.*, 1994] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[Culotta and McCallum, 2005] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[Deng *et al.*, 2013] Yue Deng, Qionghai Dai, Risheng Liu, Zengke Zhang, and Sanqing Hu. Low-rank structure learning via nonconvex heuristic recovery. *IEEE TNNLS*, 24(3):383–396, 2013.

[Deng *et al.*, 2016] Yue Deng, Feng Bao, Xuesong Deng, Ruiping Wang, Youyong Kong, and Qionghai Dai. Deep and structured robust information theoretic learning for image analysis. *IEEE TIP*, 25(9):4209–4221, 2016.

[Deng *et al.*, 2017a] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE TNNLS*, 28(3):653–664, 2017.

[Deng *et al.*, 2017b] Yue Deng, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai. A hierarchical fused fuzzy deep neural network for data classification. *IEEE TFS*, 25(4):1006–1012, 2017.

[Deng *et al.*, 2017c] Yue Deng, Yilin Shen, and Hongxia Jin. Disguise adversarial networks for click-through rate prediction. In *IJCAI*, pages 1589–1595. AAAI Press, 2017.

[Deoras and Sarikaya, 2013] Anoop Deoras and Ruhi Sarikaya. Deep belief network based semantic taggers for spoken language understanding. In *Interspeech*, pages 2713–2717, 2013.

[Dong and Lapata, 2016] Li Dong and Mirella Lapata. Language to logical form with neural attention. *ACL*, 2016.

[Hemphill *et al.*, 1990] Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language Workshop*, 1990.

[Karpathy and Fei-Fei, 2015] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.

[Kim *et al.*, 2006] Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. Mmr-based active machine learning for bio named entity recognition. In *NAACL*, pages 69–72. Association for Computational Linguistics, 2006.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[Koehn, 2004] Philipp Koehn. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Association for Machine Translation in the Americas*, pages 115–124. Springer, 2004.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014.

[Liu and Lane, 2016] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech*, 2016.

[Luong *et al.*,] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation.

[Miller *et al.*, 2014] Brad Miller, Alex Kantchelian, Sadia Afroz, Rekha Bachwani, Edwin Dauber, Ling Huang, Michael Carl Tschantz, Anthony D Joseph, and J Doug Tygar. Adversarial active learning. In *Artificial Intelligent and Security Workshop*, pages 3–14. ACM, 2014.

[Scheffer *et al.*, 2001] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[Settles and Craven, 2008] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[Settles, 2010] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[Seung *et al.*, 1992] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[Sutton and McCallum, 2006] Charles Sutton and Andrew McCallum. *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press, 2006.

[Vinyals *et al.*, 2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[Zhu and Bento, 2017] Jia-Jie Zhu and Jose Bento. Generative adversarial active learning. *arXiv*, 2017.

Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling

Bing Liu[1], *Ian Lane*[1,2]

[1]Electrical and Computer Engineering, Carnegie Mellon University
[2]Language Technologies Institute, Carnegie Mellon University
liubing@cmu.edu, lane@cmu.edu

Abstract

Attention-based encoder-decoder neural network models have recently shown promising results in machine translation and speech recognition. In this work, we propose an attention-based neural network model for joint intent detection and slot filling, both of which are critical steps for many speech understanding and dialog systems. Unlike in machine translation and speech recognition, alignment is explicit in slot filling. We explore different strategies in incorporating this alignment information to the encoder-decoder framework. Learning from the attention mechanism in encoder-decoder model, we further propose introducing attention to the alignment-based RNN models. Such attentions provide additional information to the intent classification and slot label prediction. Our independent task models achieve state-of-the-art intent detection error rate and slot filling F1 score on the benchmark ATIS task. Our joint training model further obtains 0.56% absolute (23.8% relative) error reduction on intent detection and 0.23% absolute gain on slot filling over the independent task models.

Index Terms: Spoken Language Understanding, Slot Filling, Intent Detection, Recurrent Neural Networks, Attention Model

1. Introduction

Spoken language understanding (SLU) system is a critical component in spoken dialogue systems. SLU system typically involves identifying speaker's intent and extracting semantic constituents from the natural language query, two tasks that are often referred to as intent detection and slot filling.

Intent detection and slot filling are usually processed separately. Intent detection can be treated as a semantic utterance classification problem, and popular classifiers like support vector machines (SVMs) [1] and deep neural network methods [2] can be applied. Slot filling can be treated as a sequence labeling task. Popular approaches to solving sequence labeling problems include maximum entropy Markov models (MEMMs) [3], conditional random fields (CRFs) [4], and recurrent neural networks (RNNs) [5, 6, 7]. Joint model for intent detection and slot filling has also been proposed in literature [8, 9]. Such joint model simplifies the SLU system, as only one model needs to be trained and fine-tuned for the two tasks.

Recently, encoder-decoder neural network models have been successfully applied in many sequence learning problems such as machine translation [10] and speech recognition [11]. The main idea behind the encoder-decoder model is to encode input sequence into a dense vector, and then use this vector to generate corresponding output sequence. The attention mechanism introduced in [12] enables the encoder-decoder architecture to learn to align and decode simultaneously.

In this work, we investigate how an SLU model can benefit from the strong modeling capacity of the sequence models. Attention-based encoder-decoder model is capable of mapping sequences that are of different lengths when no alignment information is given. In slot filling, however, alignment is explicit, and thus alignment-based RNN models typically work well. We would like to investigate the combination of the attention-based and alignment-based methods. Specifically, we want to explore how the alignment information in slot filling can be best utilized in the encoder-decoder models, and on the other hand, whether the alignment-based RNN slot filling models can be further improved with the attention mechanism that introduced from the encoder-decoder architecture. Moreover, we want to investigate how slot filling and intent detection can be jointly modeled under such schemes.

The remainder of the paper is organized as follows. In section 2, we introduce the background on using RNN for slot filling and using encoder-decoder models for sequence learning. In section 3, we describe two approaches for jointly modeling intent and slot filling. Section 4 discusses the experiment setup and results on ATIS benchmarking task. Section 5 concludes the work.

2. Background

2.1. RNN for Slot Filling

Slot filling can be treated as a sequence labeling problem, where we have training examples of $\left\{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}) : n = 1, ..., N\right\}$ and we want to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an input sequence $\mathbf{x}$ to the corresponding label sequence $\mathbf{y}$. In slot filling, the input sequence and label sequence are of the same length, and thus there is explicit alignment.

Sentence	first	class	fares	from	boston	to	denver
Slots	B-class_type	I-class_type	O	O	B-fromloc	O	B-toloc
Intent	airfare						

Figure 1: *ATIS corpus sample with intent and slot annotation.*

RNNs have been widely used in many sequence modeling problems [6, 13]. At each time step of slot filling, RNN reads a word as input and predicts its corresponding slot label considering all available information from the input and the emitted output sequences. The model is trained to find the best parameter set θ that maximizes the likelihood:

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

where $\mathbf{x}$ represents the input word sequence, y_1^{t-1} represents the output label sequence prior to time step t. During inference, we want to find the best label sequence $\mathbf{y}$ given an input sequence $\mathbf{x}$ such that:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{2}$$

2.2. RNN Encoder-Decoder

The RNN encoder-decoder framework is firstly introduced in [10] and [14]. The encoder and decoder are two separate RNNs. The encoder reads a sequence of input $(x_1, ..., x_T)$ to a vector c. This vector encodes information of the whole source sequence, and is used in decoder to generate the target output sequence. The decoder defines the probability of the output sequence as:

$$P(\mathbf{y}) = \prod_{t=1}^{T} P(y_t|y_1^{t-1}, c) \tag{3}$$

where y_1^{t-1} represents the predicted output sequence prior to time step t. Comparing to an RNN model for sequence labeling, the RNN encoder-decoder model is capable of mapping sequence to sequence with different lengths. There is no explicit alignment between source and target sequences. The attention mechanism later introduced in [12] enables the encoder-decoder model to learn a soft alignment and to decode at the same time.

3. Proposed Methods

In this section, we first describe our approach on integrating alignment information to the encoder-decoder architecture for slot filling and intent detection. Following that, we describe the proposed method on introducing attention mechanism from the encoder-decoder architecture to the alignment-based RNN models.

3.1. Encoder-Decoder Model with Aligned Inputs

The encoder-decoder model for joint intent detection and slot filling is illustrated in Figure 2. On encoder side, we use a bidirectional RNN. Bidirectional RNN has been successfully applied in speech recognition [15] and spoken language understanding [6]. We use LSTM [16] as the basic recurrent network unit for its ability to better model long-term dependencies comparing to simple RNN.

In slot filling, we want to map a word sequence $\mathbf{x} = (x_1, ..., x_T)$ to its corresponding slot label sequence $\mathbf{y} = (y_1, ..., y_T)$. The bidirectional RNN encoder reads the source word sequence forward and backward. The forward RNN reads the word sequence in its original order and generates a hidden state fh_i at each time step. Similarly, the backward RNN reads the word sequence in its reverse order and generate a sequence of hidden states $(bh_T, ..., bh_1)$. The final encoder hidden state h_i at each time step i is a concatenation of the forward state fh_i and backward state bh_i, i.e. $h_i = [fh_i, bh_i]$.

The last state of the forward and backward encoder RNN carries information of the entire source sequence. We use the last state of the backward encoder RNN to compute the initial decoder hidden state following the approach in [12]. The decoder is a unidirectional RNN. Again, we use an LSTM cell as the basic RNN unit. At each decoding step i, the decoder state s_i is calculated as a function of the previous decoder state s_{i-1}, the previous emitted label y_{i-1}, the aligned encoder hidden state h_i, and the context vector c_i:

$$s_i = f(s_{i-1}, y_{i-1}, h_i, c_i) \tag{4}$$

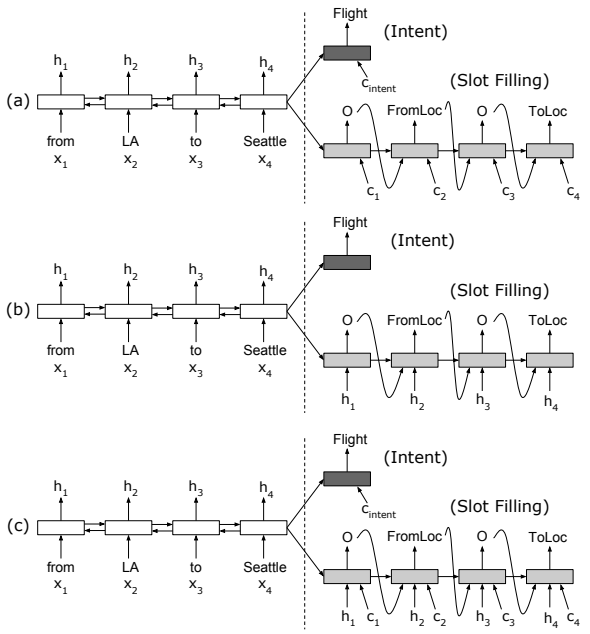


Figure 2: *Encoder-decoder model for joint intent detection and slot filling. (a) with no aligned inputs. (b) with aligned inputs. (c) with aligned inputs and attention. Encoder is a bidirectional RNN. The last hidden state of the backward encoder RNN is used to initialize the decoder RNN state.*

where the context vector c_i is computed as a weighted sum of the encoder states $\mathbf{h} = (h_1, ..., h_T)$ [12]:

$$c_i = \sum_{j=1}^{T} \alpha_{i,j} h_j \tag{5}$$

and

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{T} \exp(e_{i,k})}$$
$$e_{i,k} = g(s_{i-1}, h_k) \tag{6}$$

g a feed-forward neural network. At each decoding step, the explicit aligned input is the encoder state h_i. The context vector c_i provides additional information to the decoder and can be seen as a continuous bag of weighted features $(h_1, ..., h_T)$.

For joint modeling of intent detection and slot filling, we add an additional decoder for intent detection (or intent classification) task that shares the same encoder with slot filling decoder. During model training, costs from both decoders are back-propagated to the encoder. The intent decoder generates only one single output which is the intent class distribution of the sentence, and thus alignment is not required. The intent decoder state is a function of the shared initial decoder state s_0, which encodes information of the entire source sequence, and the context vector c_{intent}, which indicates part of the source sequence that the intent decoder pays attention to.

3.2. Attention-Based RNN Model

The attention-based RNN model for joint intent detection and slot filling is illustrated in Figure 3. The idea of introducing attention to the alignment-based RNN sequence labeling model

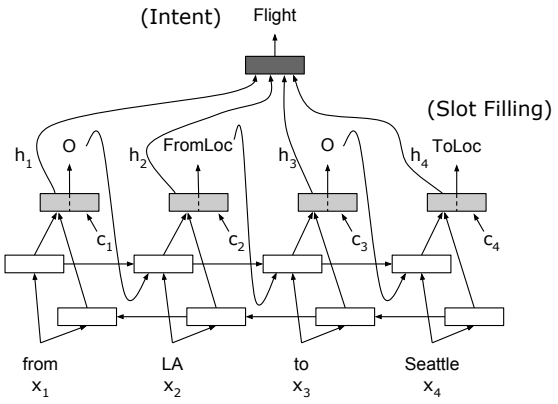


Figure 3: *Attention-based RNN model for joint intent detection and slot filling. The bidirectional RNN reads the source sequence forward and backward. Slot label dependency is modeled in the forward RNN. At each time step, the concatenated forward and backward hidden states is used to predict the slot label. If attention is enabled, the context vector c_i provides information from parts of the input sequence that is used together with the time aligned hidden state h_i for slot label prediction.*

is motivated by the use of attention mechanism in encoder-decoder models. In bidirectional RNN for sequence labeling, the hidden state at each time step carries information of the whole sequence, but information may gradually lose along the forward and backward propagation. Thus, when making slot label prediction, instead of only utilizing the aligned hidden state h_i at each step, we would like to see whether the use of context vector c_i gives us any additional supporting information, especially those require longer term dependencies that is not being fully captured by the hidden state.

In the proposed model, a bidirectional RNN (BiRNN) reads the source sequence in both forward and backward directions. We use LSTM cell for the basic RNN unit. Slot label dependencies are modeled in the forward RNN. Similar to the encoder module in the above described encoder-decoder architecture, the hidden state h_i at each step is a concatenation of the forward state fh_i and backward state bh_i, $h_i = [fh_i, bh_i]$. Each hidden state h_i contains information of the whole input word sequence, with strong focus on the parts surrounding the word at step i. This hidden state h_i is then combined with the context vector c_i to produce the label distribution, where the context vector c_i is calculated as a weighted average of the RNN hidden states $\mathbf{h} = (h_1, ..., h_T)$.

For joint modeling of intent detection and slot filling, we reuse the pre-computed hidden states $\mathbf{h}$ of the bidirectional RNN to produce intent class distribution. If attention is not used, we apply mean-pooling [17] over time on the hidden states $\mathbf{h}$ followed by logistic regression to perform the intent classification. If attention is enabled, we instead take the weighted average of the hidden states $\mathbf{h}$ over time.

Comparing to the attention-based encoder-decoder model that utilizes explicit aligned inputs, the attention-based RNN model is more computational efficient. During model training, the encoder-decoder slot filling model reads through the input sequence twice, while the attention-based RNN model reads through the input sequence only once.

4. Experiments

4.1. Data

ATIS (Airline Travel Information Systems) data set [18] is widely used in SLU research. The data set contains audio recordings of people making flight reservations. In this work, we follow the ATIS corpus[1] setup used in [6, 7, 9, 19]. The training set contains 4978 utterances from the ATIS-2 and ATIS-3 corpora, and the test set contains 893 utterances from the ATIS-3 NOV93 and DEC94 data sets. There are in total 127 distinct slot labels and 18 different intent types. We evaluate the system performance on slot filling using F1 score, and the performance on intent detection using classification error rate.

We obtained another ATIS text corpus that was used in [9] and [20] for SLU evaluation. This corpus contains 5138 utterances with both intent and slot labels annotated. In total there are 110 different slot labels and 21 intent types. We use the same 10-fold cross validation setup as in [9] and [20].

4.2. Training Procedure

LSTM cell is used as the basic RNN unit in the experiments. Our LSTM implementation follows the design in [21]. Given the size the data set, we set the number of units in LSTM cell as 128. The default forget gate bias is set to 1 [22]. We use only one layer of LSTM in the proposed models, and deeper models by stacking the LSTM layers are to be explored in future work.

Word embeddings of size 128 are randomly initialized and fine-tuned during mini-batch training with batch size of 16. Dropout rate 0.5 is applied to the non-recurrent connections [21] during model training for regularization. Maximum norm for gradient clipping is set to 5. We use Adam optimization method following the suggested parameter setup in [23].

4.3. Independent Training Model Results: Slot Filling

We first report the results on our independent task training models. Table 1 shows the slot filling F1 scores using our proposed architectures. Table 2 compares our proposed model performance on slot filling to previously reported results.

Table 1: *Independent training model results on ATIS slot filling.*

Model	F1 Score	Average
(a) Encoder-decoder NN with no aligned inputs	81.64	79.66 ± 1.59
(b) Encoder-decoder NN with aligned inputs	95.72	95.38 ± 0.18
(c) Encoder-decoder NN with aligned inputs & attention	**95.78**	95.47 ± 0.22
BiRNN no attention	95.71	95.37 ± 0.19
BiRNN with attention	**95.75**	95.42 ± 0.18

In Table 1, the first set of results are for variations of encoder-decoder models described in section 3.1. Not to our surprise, the pure attention-based slot filling model that does not utilize explicit alignment information performs poorly. Letting the model to learn the alignment from training data does not seem to be appropriate for slot filling task. Line 2 and line 3 show the F1 scores of the non-attention and attention-based encode-decoder models that utilize the aligned inputs. The

[1] We thank Gokhan Tur and Puyang Xu for sharing the ATIS data set.

attention-based model gives slightly better F1 score than the non-attention-based one, on both the average and best scores. By investigating the attention learned by the model, we find that the attention weights are more likely to be evenly distributed across words in the source sequence. There are a few cases where we observe insightful attention (Figure 4) that the decoder pays to the input sequence, and that might partly explain the observed performance gain when attention is enabled.

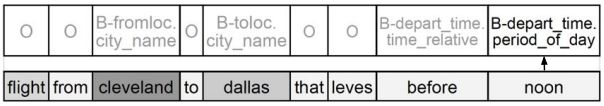


Figure 4: *Illustration of the inferred attention when predicting the slot label for the last word "noon" in the given sentence. Darker shades indicate higher attention weights. When word "noon" is fed to the model as the aligned input, the attention mechanism tries to find other supporting information from the input word sequence for the slot label prediction.*

The second set of results in Table 1 are for bidirectional RNN models described in section 3.2. Similar to the previous set of results, we observe slightly improved F1 score on the model that uses attentions. The contribution from the context vector for slot filling is not very obvious. It seems that for sequence length at such level (average sentence length is 11 for this ATIS corpus), the hidden state h_i that produced by the bidirectional RNN is capable of encoding most of the information that is needed to make the slot label prediction.

Table 2 compares our slot filling models to previous approaches. Results from both of our model architectures advance the best F1 scores reported previously.

Table 2: *Comparison to previous approaches. Independent training model results on ATIS slot filling.*

Model	F1 Score
CNN-CRF [9]	94.35
RNN with Label Sampling [7]	94.89
Hybrid RNN [6]	95.06
Deep LSTM [5]	95.08
RNN-EM [24]	95.25
Encoder-labeler Deep LSTM [25]	95.66
Attention Encoder-Decoder NN (with aligned inputs)	**95.78**
Attention BiRNN	**95.75**

4.4. Independent Training Model Results: Intent Detection

Table 3 compares intent classification error rate between our intent models and previous approaches. Intent error rate of our proposed models outperform the state-of-the-art results by a large margin. The attention-based encoder-decoder intent model advances the bidirectional RNN model. This might be attributed to the sequence level information passed from the encoder and additional layer of non-linearity in the decoder RNN.

4.5. Joint Model Results

Table 4 shows our joint training model performance on intent detection and slot filling comparing to previous reported results. As shown in this table, the joint training model using encoder-decoder architecture achieves 0.09% absolute gain on slot filling and 0.45% absolute gain (22.2% relative improvement) on intent detection over the independent training model. For the attention-based bidirectional RNN architecture, the join training model achieves 0.23% absolute gain on slot filling and 0.56% absolute gain (23.8% relative improvement) on intent detection over the independent training models. The attention-based RNN model seems to benefit more from the joint training. Results from both of our joint training approaches outperform the best reported joint modeling results.

Table 3: *Comparison to previous approaches. Independent training model results on ATIS intent detection.*

Model	Error (%)
Recursive NN [8]	4.60
Boosting [19]	4.38
Boosting + Simplified sentences [26]	3.02
Attention Encoder-Decoder NN	**2.02**
Attention BiRNN	**2.35**

Table 4: *Comparison to previous approaches. Joint training model results on ATIS slot filling and intent detection.*

Model	F1 Score	Intent Error (%)
RecNN [8]	93.22	4.60
RecNN+Viterbi [8]	93.96	4.60
Attention Encoder-Decoder NN (with aligned inputs)	**95.87**	**1.57**
Attention BiRNN	**95.98**	**1.79**

To further verify the performance of our joint training models, we apply the proposed models on the additional ATIS data set and evaluate them with 10-fold cross validation same as in [9] and [20]. Both the encoder-decoder and attention-based RNN methods achieve promising results.

Table 5: *Joint training model results on the additional ATIS corpus using 10-fold cross validation.*

Model	F1 Score	Intent Error (%)
TriCRF [20]	94.42	6.93
CNN TriCRF [9]	95.42	5.91
Attention Encoder-Decoder NN (with aligned inputs)	**95.62**	**5.86**
Attention BiRNN	**95.78**	**5.60**

5. Conclusions

In this paper, we explored strategies in utilizing explicit alignment information in the attention-based encoder-decoder neural network models. We further proposed an attention-based bidirectional RNN model for joint intent detection and slot filling. Using a joint model for the two SLU tasks simplifies the dialog system, as only one model needs to be trained and deployed. Our independent training models achieved state-of-the-art performance for both intent detection and slot filling on the benchmark ATIS task. The proposed joint training models improved the intent detection accuracy and slot filling F1 score further over the independent training models.

6. References

[1] P. Haffner, G. Tur, and J. H. Wright, "Optimizing svms for complex call classification," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 1. IEEE, 2003, pp. I–632.

[2] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5680–5683.

[3] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation." in *ICML*, vol. 17, 2000, pp. 591–598.

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

[5] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 189–194.

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

[7] B. Liu and I. Lane, "Recurrent neural network structured output prediction for spoken language understanding," in *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*, 2015.

[8] D. Guo, G. Tur, W.-t. Yih, and G. Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 554–559.

[9] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 78–83.

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

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

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

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

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[15] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

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

[19] G. Tur, D. Hakkani-Tur, and L. Heck, "What is left to be understood in atis?" in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 19–24.

[20] M. Jeong and G. Geunbae Lee, "Triangular-chain conditional random fields," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1287–1302, 2008.

[21] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[22] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] B. Peng and K. Yao, "Recurrent neural networks with external memory for language understanding," *arXiv preprint arXiv:1506.00195*, 2015.

[25] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder lstm for natural language understanding," *arXiv preprint arXiv:1601.01530*, 2016.

[26] G. Tur, D. Hakkani-Tür, L. Heck, and S. Parthasarathy, "Sentence simplification for spoken language understanding," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5628–5631.

FRAMES: A CORPUS FOR ADDING MEMORY TO GOAL-ORIENTED DIALOGUE SYSTEMS

Layla El Asri* **Hannes Schulz*** **Shikhar Sharma*** **Jeremie Zumer***
Justin Harris **Emery Fine** **Rahul Mehrotra** **Kaheer Suleman**

Maluuba Research
Montreal, Canada
first.last@maluuba.com
***these authors contributed equally**

ABSTRACT

This paper presents the *Frames* dataset[1], a corpus of 1369 human-human dialogues with an average of 15 turns per dialogue. We developed this dataset to study the role of memory in goal-oriented dialogue systems. Based on Frames, we introduce a task called *frame tracking*, which extends state tracking to a setting where several states are tracked simultaneously. We propose a baseline model for this task. We show that Frames can also be used to study memory in dialogue management and information presentation through natural language generation.

1 INTRODUCTION

Goal-oriented, information-retrieving dialogue systems have traditionally been designed to help users find items in a database given a certain set of constraints (El Asri et al., 2014; Laroche et al., 2011; Raux et al., 2003; Singh et al., 2002). For instance, the *LET'S GO* dialogue system finds a bus schedule given a bus number and a location (Raux et al., 2003).

These systems model dialogue as a sequential process: the system asks for constraints until it can query the database and return a few results to the user. Then, the user can ask for more information about a given result or ask for other possibilities. If the user wants to know about database items corresponding to a different set of constraints (*e.g.*, another bus line), then these constraints simply overwrite the previous ones. As a consequence, users can neither compare results corresponding to different constraints, nor go back-and-forth between results.

We can assume users in the bus domain know exactly what they want. In contrast, user studies in e-commerce have shown that several information-seeking behaviours are encountered: users may come with a very well defined item in mind, but they may also visit an e-commerce website with the intent to compare items and explore different possibilities (Moe and Fader, 2001). Supporting this kind of decision-making process in conversational systems implies adding *memory*. Memory is necessary to track different items or preferences set by the user during the dialogue. For instance, consider product comparisons. If a user wants to compare different items using a dialogue system, then this system should be able to separately recall properties pertaining to each item.

This paper presents the *Frames* dataset, which comprises dialogues that require this type of memory. Frames is a corpus of 1369 human-human dialogues collected in a Wizard-of-Oz (WOz) setting – *i.e.*, users were connected to humans, whom we refer to as *the wizards*, who were assuming the role of the dialogue system. The wizards had access to a database of vacation packages containing round-trip flights and a hotel. The users were asked to find packages based on a few constraints such as a destination and a budget.

In order to test the memory capabilities of conversational agents, we formalize a new task called *frame tracking*. In frame tracking, the agent must simultaneously track multiple semantic frames

[1]Frames is available at http://datasets.maluuba.com/Frames

throughout the dialogue. For example, two frames would be constructed and recalled while comparing two products – each containing the properties of a specific item. Frame tracking is a generalization of the state tracking task (Henderson, 2015). In state tracking, all the information summarizing a dialogue history is compressed into one semantic frame. In contrast, several frames are kept in memory during frame tracking, with each frame corresponding to a particular context, *e.g.*, one or more vacation packages.

Another important property of human dialogue that we want to study with the Frames dataset is how to provide the user with information on the database. When a set of user constraints leads to no results, users would benefit from knowing that relaxing a given constraint (*e.g.*, increasing the budget by a reasonable amount) would lead to results instead of navigating the database blindly. This recommendation behaviour is in accordance with Grice's cooperative principle (Grice, 1989): "Make your contribution as informative as is required (for the current purposes of the exchange)". We study this by including suggestions when the database returns no results for a given user query.

This paper describes the Frames dataset in detail, formally defines the frame tracking task, and provides a baseline model for frame tracking. The next section discusses motivation for the Frames dataset. **??** explains the data collection process and **??** describes the dataset in detail. We describe the annotation scheme in **??**. In **??**, we identify the main research topics of the corpus and formalize the frame tracking task. The dialogue data format is described in **??**. **??** proposes a baseline model for frame tracking. Finally, we conclude in **??** and suggest directions for future work.

2 MOTIVATION

Much work has focused on spoken dialogue (Lemon and Pietquin, 2007; Walker et al., 1998; Williams and Zweig, 2016), since spoken dialogue systems are useful in many settings, including in hands-free environments such as cars (Lemon et al., 2006). A generation of voice assistants – such as SIRI, Cortana, and Google Voice – have popularized spoken dialogue systems. More recently, users have become familiar with *chatbots*. Many platforms for deploying chatbots are now available, such as Facebook Messenger, Slack, or Kik. Text offers advantages over voice such as privacy and the ability to avoid bad speech recognition in noisy environments. Chatbots provide a welcome alternative to downloading and installing applications, and make a lot of sense for everyday services such as ordering a cab or knowing the weather. Chatbots have been proposed for tasks that one would traditionally perform through Graphical User Interfaces (GUIs). For instance, many chatbots for booking a flight are now entering the market.

In most cases, as with current voice-based assistants, the conversation with a chatbot is very limited: asking for the weather and ordering a cab are accomplished with simple, sequential slot-filling. These tasks have in common the fact that in both cases the user knows exactly what she wants, *i.e.*, the destination for the cab or the city for the weather. Booking a flight is a bit different. Flight booking requires specifying many parameters, and these are usually determined during the search process[2]. Technically, finding a weather forecast is only about reading a database: the task is to form a complete database query and then to verbalise the result to the user. The user might start with very few constraints and then refine her query given the database results. In the case of booking a flight, there is a decision-making process requiring comparison and backtracking.

GUIs are not optimal on many levels when it comes to helping users through this decision-making process. A first point of friction is the limited visual space. Consider the example of e-commerce websites. A user is very likely to compare different options before picking an item to buy. This often results in a large number of open browser tabs among which the user must navigate. In order to avoid this situation, some websites provide a comparator that can be used to display several items on a single page. However, this option is not optimal for hierarchical objects such as vacation packages because these objects have global properties (dates of the trips) but are also composed of different modules (flights and hotel) which have different properties (*e.g.*, seat class and hotel category). Optimally, the user should be able to define the properties for the different modules while being able to compare items corresponding to each set of properties. A text interface could complement a GUI by offering this flexibility while remembering the properties mentioned by the user and displaying comparisons when asked.

[2]One can easily imagine a user changing from economy to business class if the price difference is small.

We propose the Frames dataset to support work on text-based conversational agents which help a user make a decision. The decision-making process is tightly coupled with the notion of memory. Indeed, if a user intends to compare different options in the course of defining the options, the system should follow the user's path and remember every option. In this paper, we formalize this aspect of conversation in the *frame tracking* task. This task is the main challenge of the Frames corpus, and **??** describes it in detail.

3 Data Collection

We collected data over a period of 20 days and with 12 participants. To increase variation in the dialogues, 8 participants were hired for only one week each, and one participant was hired for one day. The three remaining participants participated in the entire data collection. The participants were paired up for each dialogue and interacted through a chat interface.

3.1 Wizard-Of-Oz Data Collection

Data collection for goal-oriented dialogue is challenging. To control the data such that specific aspects of the problem can be studied, it is common to collect dialogues using an automated system. This requires, *e.g.*, a natural language understanding module that already performs well on the task, which implies possession of in-domain data or the ability to generate it (Henderson et al., 2014; Raux et al., 2003). Another possibility, which permits even greater control, is to generate dialogues using a rule-based system (Bordes and Weston, 2016). These approaches are useful for studying specific modules and analysing the behaviour of different architectures. However, it is costly to generate new dialogues for each experiment and skills acquired on artificial data are not directly usable in real settings because of natural language understanding noise (Bordes and Weston, 2016).

The Wizard-of-Oz (WOz) approach offers a powerful alternative (Kelley, 1984; Rieser et al., 2005; Wen et al., 2016). In WOz data collection, one participant (the wizard) plays the role of the dialogue system. The wizard has access to a search interface connected to the database. She receives the user's input in text form and decides what to say next. This does not require preexisting dialogue system components, except potentially automatic speech recognition for transcribing the user's inputs. Dialogues collected in WOz settings can be used for studying and developing every part of a dialogue system, from language understanding to language generation. They are also essential for offline training of end-to-end dialogue systems (Bordes and Weston, 2016; Wen et al., 2016) on different domains, which may reduce costs from handcrafting new systems for each domain.

WOz dialogues also have the considerable advantage of exhibiting realistic behaviours that cannot be supported by current (end-to-end or not) architectures. Since there is no dialogue system that incorporates the type of memory that we want to study with this dataset, we need to work directly on human-human dialogues. Our setting is a bit different from the usual WOz setting because, in our case, the users did not think they were interacting with a dialogue system but instead knew that they were talking to a human-being. We made the choice not to give templated answers to the wizards because, apart from studying memory, we also want to study information presentation and dialogue management. We have chosen to work on text-based dialogues because this allows a more controlled wizard behaviour, obviates handling time-sensitive turn-taking and speech recognition noise, and allows studying more complex dialogue flows.

3.2 Task Templates and Instructions

Dialogues were performed on Slack[3]. We deployed a Slack bot named *wozbot* to pair up participants and record conversations. The participants in the user role indicated when they were available for a new dialogue through this bot. They were then assigned to an available wizard and received a new task. The tasks were built from templates such as the following:

> "Find a vacation between `[START_DATE]` and `[END_DATE]` for `[NUM_ADULTS]` adults and `[NUM_CHILDREN]` kids. You leave from

[3] www.slack.com

[ORIGIN_CITY]. You are travelling on a budget and you would like to spend at most $[BUDGET]."

Each template had a probability of success. The tasks were generated by drawing values (*e.g.*, BUDGET) from the database. The generated tasks were then added to a pool. The values were drawn in order to comply with the template's probability of success. For example, if 20 tasks were generated at probability 0.5, about 10 tasks would be generated with successful database queries and the other 10 would be generated so the database returned no results for the constraints. This mechanism allowed us to emulate cases when a user would not find anything meeting her constraints. If a task was unsuccessful, the user either ended the dialogue or got an alternate task such as:

"If nothing matches your constraints, try increasing your budget by $200."

We wrote 38 templates. 14 templates were generic such as the one presented above and the other 24 were written to encourage more role-playing from users. One example is:

"Pokemon are now a worldwide currency. You are the best Pokemon hunter in the world. You have caught them all except for one Pokemon worth a fortune: Mewtwo. You heard it was spotted somewhere in [DESTINATION_CITY] and [DESTINATION_CITY]. You want to visit one of these cities, leaving from [ORIGIN_CITY] and starting on or after [START_DATE]. You are leaving with your hunting team and you will be a total of [NUM_ADULTS] adults. You have a budget of [PRICE_MAX]. You want to compare the packages between the different cities and book one, the one that will take you to your destiny."

These templates were meant to add variety to the dialogues. The generic templates were also important for the users to create their own character and personality. We found that the combination of the two types of templates prevented the task from becoming too repetitive. Notably, we distributed the role-playing templates throughout the data collection process to bring some novelty and surprise. We also asked the participants to write templates (13 of them) to keep them engaged in the task.

To control data collection, we gave a set of instructions to the participants. The users received the following instructions:

- Do not use uncommon slang terms, but feel free to use colloquialisms.
- Make up personalities.
- Feel free to end the conversation at any time.
- Try to spell things correctly.
- You do not necessarily have to choose an option.
- Try to determine what you can get for your money.

These instructions were meant to encourage a variety of behaviours from the users. As for the wizards, they were asked to only talk about the database results and the task-at-hand. This is necessary if one wants to build a dialogue system that emulates the wizards' behaviour in this corpus. The wizard instructions were as follows:

- Be polite, and do not jump in on the role play of the users.
- Vary the way you answer the user, sometimes you can say something that would be right at another point in a dialogue.
- Your knowledge of the world is limited by your database.
- Try to spell things correctly.

We asked the wizards to sometimes act badly (second point in the list). It is interesting from a dialogue management point of view to have examples of bad behaviour and of how it impacts user satisfaction. At the end of each dialogue, the user was asked to provide a wizard cooperativity rating on a scale of 1 to 5. The wizard, on the other hand, was shown the user's task and was asked whether she thought the user had accomplished it.

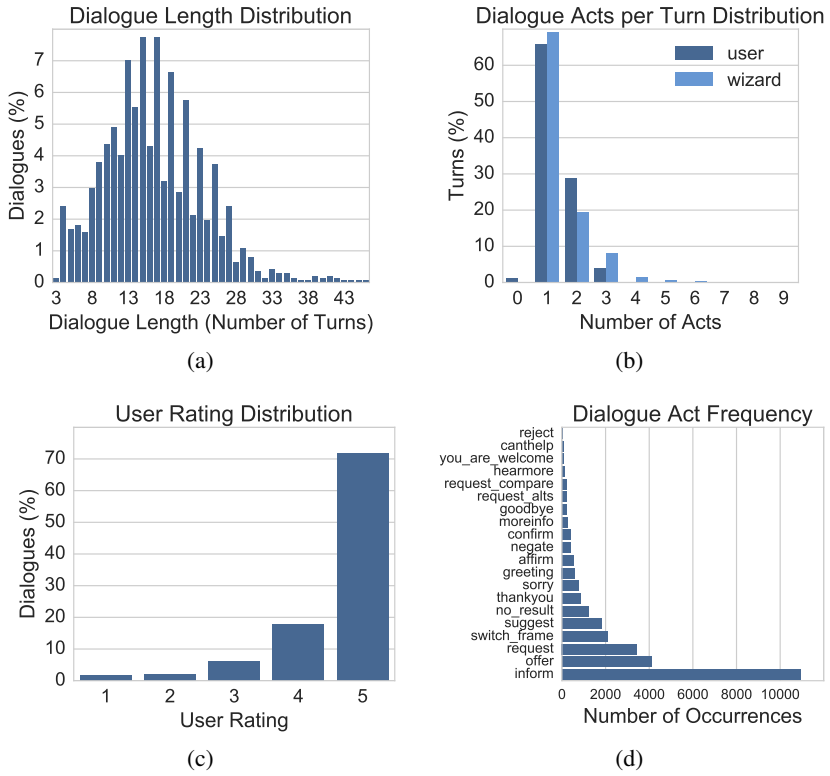


Figure 1: Overview of the Frames corpus

3.3 Database Search Interface

Wizards received a link to a search interface every time a user was connected with them. The search interface was a simple GUI with all the searchable fields in the database (see **??**). For every search in the database, up to 10 results were displayed. These results were sorted by increasing price.

3.4 Suggestions

When a database query returned no results, suggestions were sometimes displayed to the wizards. Suggestions were packages obtained by relaxing one or more constraints. It was up to the wizard to decide whether or not to use suggestions. Our goal with suggestions is not to learn a recommender system, but to learn the timing of recommendation, hence the randomness of the mechanism.

4 Statistics of the Corpus

We used the data collection process described in the previous section to collect *1369 dialogues*. **??** shows the distribution of dialogue length in the corpus. The average number of turns is 15, for a total of *19986 turns* in the dataset. A turn is defined as a Slack message sent by either a user or a wizard. A user turn is always followed by a wizard turn and vice versa.

?? shows the number of acts per dialogue turn. About 25% of the dialogue turns have more than one dialogue act. The turns with 0 dialogue act are turns where the user asked for something that the wizard could not provide, *e.g.*, because it was not part of the database. An example in the dataset is: "Would my room have a view of the city? How much would it cost to upgrade to a room with a view?". We left such user turns unannotated and they are usually followed up by the wizard saying she cannot provide the required information.

?? shows the distribution of user ratings. More than 70% of the dialogues have the maximum rating of 5. **??** shows the occurrences of dialogue acts in the corpus. The dialogue acts are described in **??** in **??**. We present the annotation scheme in the following section.

5 Dialogue Annotation Scheme

We annotated the Frames dataset with three types of labels:

1. Dialogue acts, slot types, slot values, and references to other frames for each utterance.
2. The ID of the currently active frame.
3. Frame labels which were automatically computed based on the previous two sets of labels.

5.1 Dialogue Acts, Slot Types, and Slot Values

Most of the dialogue acts used for annotation are acts which are usually encountered in the goal-oriented setting such as `inform` and `offer`. We also introduced dialogue acts which are specific to our frame tracking setting such as `switch_frame` and `request_compare`. The dialogue acts are listed in **??**.

Our annotation uses three sets of slot types. The first set, listed in **????**, corresponds to the fields of the database. The second set is listed in **??** and contains the slot types which we defined in order to describe specific aspects of the dialogue such as `intent` and `action`. An `intent` indicates whether or not the user wants to book a package, whereas an `action` indicates whether or not the wizard should, or did, book it. We also introduced several `count` slot types which were used most often by wizards to summarize information in the database, *e.g.*, "I have 2 hotels in Marseille". In this case, the wizard informs that the count for hotels is 2.

The remaining slot types in **??** were introduced to describe frames and cross-references between them. Before we discuss these slot types, we define frames more formally in the following section.

5.2 Frames

5.2.1 Definition

A semantic frame is defined by the following four parts:

- User comparison requests.
- User requests.
- User binary questions.
- Constraints.

The first three parts are designed to keep track of user questions. We distinguish three types of questions. The first type, comparison requests, corresponds to the `request_compare` dialogue act. This dialogue act is used to annotate turns when a user asks to compare different results, for instance: "*Could you tell me which of these resorts offers free wifi?*". These questions relate to several frames. The second type of question is user requests, corresponding to the `request` act. These are questions related to one specific frame, for instance "*how much will it cost?*". Binary questions are questions with slot types and slot values, *e.g.*, "*Is this hotel in the downtown area of the city?*" (`request` act), or "*Is the trip to Marseille cheaper than to Naples?*" (`request_compare` act), as well as all `confirm` acts.

The constraints are all the slots which have been set to a particular value by the user or the wizard. Any field in the database (see **????** in **??**) can be set by the user or the wizard. For user-created frames, the constraints are the preferences set by the user (*e.g.*, budget or city). Wizards create a frame by making an offer or a suggestion. The constraints are then the properties of the offer or the suggestion.

5.2.2 Creation and Annotation

Each dialogue starts in frame 1. New frames are introduced when the wizard offers or suggests something, or when the user modifies pre-established slots. Frames can be seen as checkpoints in a dialogue, to which the user can return. The frames contain the slot values set by the user, or by information given by the wizard. An example is given in **??**. In this example, the frame number is changed when the user changes several slot values: the destination city, the number of adults for the trip, and the budget. Though frames are created for each offer or suggestion made by the wizard, the *active* frame can only be changed by the user. If the user asks for more information about a specific offer or suggestion, the active frame is changed to the frame introduced with that offer or suggestion. This change of frame is indicated by a `switch_frame` act (see **??**). The rules for creating and switching to frames are summarized in **??**.

Table 1: Dialogue excerpt with active frame annotation

Author	**Utterance**	**Frame**
User	I'd like to book a trip to Atlantis from Caprica on Saturday, August 13, 2016 for 8 adults. I have a tight budget of 1700.	1
Wizard	Hi...I checked a few options for you, and unfortunately, we do not currently have any trips that meet this criteria. Would you like to book an alternate travel option?	1
User	Yes, how about going to Neverland from Caprica on August 13, 2016 for 5 adults. For this trip, my budget would be 1900.	2
Wizard	I checked the availability for those dates and there were no trips available. Would you like to select some alternate dates?	2

We introduced specific slot types for recording the creation and modification of frames. These slot types are `id`, `ref`, `read`, and `write` (see **??** in **??**). The frame id is defined when the frame is created and is used to switch to this frame when the user decides to do so.

The other slot types – `ref`, `read`, and `write` – are used to annotate cross-references between frames, which are a crucial component of the recorded dialogues. A reference has two parts: the number of the frame it is referring to and the slots and values that are used to refer to that frame (if any). For instance, `ref[1{name=Tropic}]` means that frame 1 is being referred to by the hotel name *Tropic*. If anaphora is used to refer to a frame, we annotated this with the slot `ref_anaphora` (*e.g.*, "This is too long" – `inform(duration=too long,ref_anaphora=this)`). Inside an `offer` dialogue act, a `ref` means that the frame corresponding to the offer is derived from another frame. For example, here is an utterance from the corpus, written by a wizard:

> "Here are a couple of options. The first option is a 3.0 star hotel (the Tropic), with a guest rating of 4.77/10 and a business class flight. The cost is 1002.27 USD. Or, if you prefer, you could choose the same 3.0 star hotel with a guest rating of 4.77/10 (the Tropic) and an economy flight, for 812.69."

Table 2: Frequency of frame creation and switching events

Rule Type	**Author**	**Rule Description**	**Frequency**	
			Relative	**Absolute**
Creation	User	Changing the value of a slot	31 %	2092
	Wizard	Making an offer or a suggestion	69 %	4762
Switching	User	Changing the value of a slot (it causes the dialogue to switch to that frame)	50 %	2092
		Considering a wizard offer or suggestion	39 %	1635
		Switching to an earlier frame by mentioning its slot values	11 %	458

Table 3: Annotation example with the `write` and `read` slot types

Author	Utterance	Frame	Annotation
Wizard	I am only able to find hotels with a 2.5 star rating in Punta Cana for that time.	6	inform(*read*=[7{dst_city=Punta Cana, category=2.5}])
User	2.5 stars will do. Can you offer any additional activities?	11	inform(category=2.5)
Wizard	Unfortunately I am not able to provide this information.	11	sorry, canthelp
User	How about breakfast?	11	request(breakfast)
Wizard	El Mar does not provide breakfast.	11	inform(breakfast=False, *write*=[7{name=El Mar}])

This utterance is annotated with the following dialogue acts:

- `offer(category=3.0,name=Tropic,gst_rating=4.77/10,id=6);`
- `offer(ref=[6],seat=business,price=1002.27 USD,id=7);`
- and `offer(ref=[6],seat=economy,price=812.69,id=8).`

Here, the frames corresponding to the last two offers are derived from the first one by inheriting all values.

The slot types `read` and `write` only occur inside a wizard's `inform` act and are used by the wizards to provide relations between offers or suggestions: `read` is used to indicate which frame the values are coming from (and which slots are used to refer to this frame, if any), while `write` indicates the frame where the slot values are to be written (and which slot values are used to refer to this frame, if any). If there is a `read` without a `write`, the current frame is assumed as the storage for the slot values. A slot type without a value indicates that the value is the same as in the referenced frame, but was not mentioned explicitly *i.e.*, "for the same price".

?? gives an example of how these slot types are used in practice: `inform(` *`read`*`=[7{dst_city=Punta Cana, category=2.5}]` means that the values *2.5* and *Punta Cana* are to be read from frame 7, and to be written in the current frame. At this turn of the dialogue, the wizard repeats information from frame 7.

The annotation `inform(breakfast=False,` *`write`*`=[7{name=El Mar}])` means that the value *False* for breakfast is written in frame 7 and that frame 7 was identified in this utterance by the name of the hotel *El Mar*.

The average number of frames created per dialogue is 6.71 and the average number of frame switches is 3.58. **??** shows boxplots for the number of frame creations and the number of frame changes in the corpus.

6 Research Topics

Frames can be used to research many aspects of goal-oriented dialogue, from Natural Language Understanding (NLU) to natural language generation. In this section, we propose three topics that we believe are new and representative of this dataset.

6.1 Frame Tracking

6.1.1 Definition

Frame tracking extends state tracking (Henderson, 2015) to a setting where several semantic frames are tracked simultaneously. In state tracking, every new slot value overwrites the previous one. In frame tracking, a new value creates a new semantic frame. The frame tracking task is significantly

harder as it requires, for each user utterance, identifying the active frame as well as all the frames which are modified by the utterance.

Definition 1 (Frame Tracking)**.** At each user turn t, we assume access to the full dialogue history $H = \{f_1, ..., f_{n_{t-1}}\}$, where f_i is a frame and n_{t-1} is the number of frames created so far in the dialogue. For a user utterance u_t at time t, we provide the following NLU labels: dialogue acts, slot types, and slot values. The purpose of frame tracking is to predict if a new frame is created and to predict for each dialogue act the `ref` labels (possibly none) and the ids of the referred frames.

Predicting the frame referred to by a dialogue act requires detecting if a new frame is created and recognizing a previous frame by the values being mentioned by the user (potentially a synonym, *e.g.*, NYC and New York) or by using the user utterance directly. It is necessary in many cases to use the user utterance directly because users do not always use slot values to refer to previous frames. An example in the corpus is a user asking: "Which package has the soonest departure?". In this case, the user refers to several frames (the packages) without ever explicitly describing which ones. This phenomenon is quite common for dialogue acts such as `switch_frame` (979 occurrences in the corpus) and `request_compare` (455 occurrences in the corpus). These cases can only be resolved by working on the text directly and solving anaphora.

6.1.2 Evaluation Metrics

We define two metrics: frame identification and frame creation. For frame identification, for each dialogue act, we compare the ground truth pair (`key-value`, `frame`) to the one predicted by the frame tracker. We compute performance as the number of correct predictions over the number of pairs. A prediction is correct if and only if the `frame`, `key`, and `value` are the same in the ground truth and the prediction. The `frame` is the id of the referred frame. The `key` and `value` are respectively the type and the value of the slot used to refer to the frame (as said previously, these can be null).

For frame creation, we compute the number of times the frame tracker predicts that a frame is created over the number of dialogue turns.

6.2 Dialogue Management

One of the notable aspects of this dataset is that memory is not only a matter of frame tracking. Most of the time, the wizard would speak about the current frame to ask or answer questions. However, sometimes, the wizard would talk about previous frames. We can see it as appealing to memories in a conversation. An example is given in **??**. In the bold utterance in this dialogue, even though the active frame is frame 4, the wizard mentions a previous frame (frame 3). In order to reproduce this

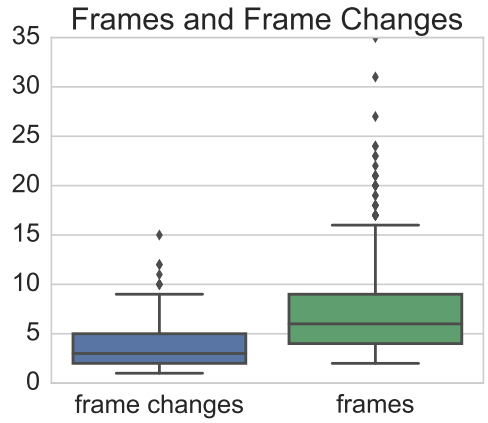


Figure 2: Number of frames and frame switches in the corpus

Table 4: Dialogue excerpt where the wizard talks about a frame other than the active frame.

Author	Utterance	Frame
User	i need a vacation	1
Wizard	How can I help?	1
User	i've got a few days off from august 26-august 31. im not flexible on this, but i still want to somehow treat myself with an 8 day trip (??) im leaving Dallas and i wanna check out mannheim	1
Wizard	would a 5 day trip suffice?	1
User	sure dude	2
Wizard	A 5 star hotel called the Regal Resort, it has free wifi and a spa.	2
User	dates?	3
Wizard	Starts on august 27th until the 30th	3
User	ok that could work. I would like to see my options in Santos as well	4
Wizard	*regal resort goes for $2800 or there is the Hotel Globetrotter in Santos it has 3 stars and comes with breakfast and wifi, it leaves on the 25th and returns on the 30th! all for $2000*	*4*
User	ahh I can't leave until august 26 though	4
Wizard	**then i guess you might have to choose the Regal resort**	**4**
User	yeah. I will book it	3
Wizard	Thank you!	3

kind of behaviour, a dialogue manager would need to be able to identify potentially relevant frames for the current turn and to output actions for these frames.

?? also illustrates another novelty. In the utterance in italic, the wizard actually performs two actions. The first action consists of informing the user about the price of the *regal resort* and the second action consists of proposing another option, *Hotel Globetrotter*. Performing more than one action per turn is a challenge when using reinforcement learning (Fatemi et al., 2016; Gašić et al., 2012; Pietquin et al., 2011) and, to our knowledge, this has only been done in a simulated setting (Laroche et al., 2009).

6.3 Natural Language Generation

An interesting behaviour observed in our dataset is that wizards often tended to summarize database results. An example is the wizard saying: *"The cheapest available flight is 1947.14USD."* In this case, the wizard informs the user that the database has no cheaper result than the one she is proposing. To imitate this behaviour, a dialogue system would need to reason over the database and decide how to present the results to the user.

7 Dataset Format

7.1 Dialogues

We provide the Frames dialogues in JSON format. Each dialogue has five main fields: `turns`, `labels`, `user_id`, `wizard_id`, and `id`. The ids are unique for each dialogue (`id`), each user (`user_id`), and each wizard (`wizard_id`).

The `labels` have two fields:

- `userSurveyRating` user rating of wizard cooperativity on a scale of 1 to 5 (see Section **??**).
- `wizardSurveyTaskSuccessful` wizard's perceived task completion (see Section **??**).

The `turns` have the following fields:

- `author` "user" or "wizard".

- `text` the author's utterance.
- `labels` the id of the currently active frame (`active_frame`) as well as a list of dialogue acts (`acts`) each with a `name`, and `args` (key-value pairs), and a list of dialogue acts without `ref` tags (`acts_without_refs`) for frame tracking.
- `timestamp` timestamp for the message.
- `frames` List of all frames after integrating the current turn. Each frame has the following labels:
  - `frame_id` id of the frame.
  - `frame_parent_id` id of the parent frame.
  - `requests, binary_questions, compare_requests` user questions (see **??**).
  - `info` properties of the frame (see **??**) Note that each slot can have multiple values, which accumulate as long as the frame does not change. For example, price can be both "1000 USD" and "cheapest". Each value has a boolean property "negated", expressing whether the user negated the value of the corresponding slot, for instance *"I don't want to stay 3 days"* (`negate(duration=3)`), or negated an explicit confirmation question. When a user switches to a frame, we assume the user accepts all information provided by the wizard for that frame as "constraints". We drop these additional constraints when a constraint is modified by the user, or the user requests alternatives. Our motivation for this scheme is to make frames more distinguishable and encourage methods which correctly identify frame switches. Additionally to slots and their values, we added the following fields to keep track of specific aspects of the dialogue:
    - `REJECTED` a boolean value expressing if the user negated or affirmed an offer made by the wizard (corresponds to a `negate` act that does not follow a question).
    - `MOREINFO` a boolean value expressing whether the user wants to know more about this frame, which happens if the wizard withholds detail information (see `moreinfo` act).
- `db` (wizard turns only) list of search queries made by the wizard with the associated search results/suggestions.

7.2 Hotels

The vacation packages were generated randomly. A database of packages can be created by using the search results in the JSON files containing the dialogues. The hotels in these search results have all the fields listed in the *Hotel Properties* section of **??** in **??**. Note that amenities or points of interest in the vicinity of the hotel are only listed in a hotel's description if they are true. For instance, the field *breakfast* is only present for hotels proposing free breakfast. **??** shows statistics for these boolean values.

8 Baselines

8.1 Natural Language Understanding

We define the NLU task as dialogue act prediction and IOB (Inside, Outside, Beginning) tagging. The IOB tag format (Inside, Outside, Beginning) is a common word-level annotation format for natural language understanding. A word tagged with *O* means that this word is not part of any slot value. The *B* and *I* tags are used for slot values. Every word which is the beginning of a slot value is tagged with *B* and the *I* tag is used for subsequent words until the end of the slot value. For instance, "I need to go to New York for business" would be tagged as "I (O) need (O) to (O) go (O) to (O) New (B) York (I) for (O) business (O)". We generated these word-level tags by matching the slot values in the manual annotations with the corresponding textual utterances. The act tags were also generated at the word level: for a given dialogue act with slot values, each word between the slot value that occurred first in the text and the one that occurred last in the text was tagged with the corresponding act. The other words were tagged with *O*.

The NLU model is illustrated in **??**. The IOB tagging part operates on character trigrams and is based on the robust named entity recognition model (Arnold et al., 2016). We predict, for each word

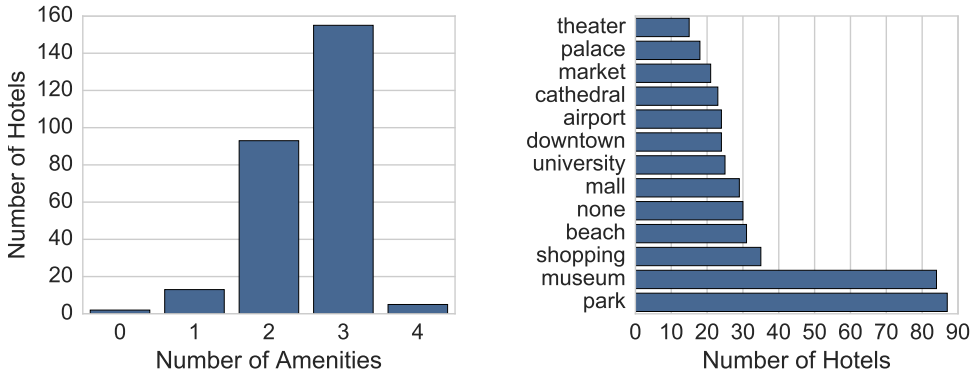


Figure 3: Left – number of amenities per hotel. Right – number of hotels in the vicinity of points of interest (none if the hotel is not close to any point of interest).

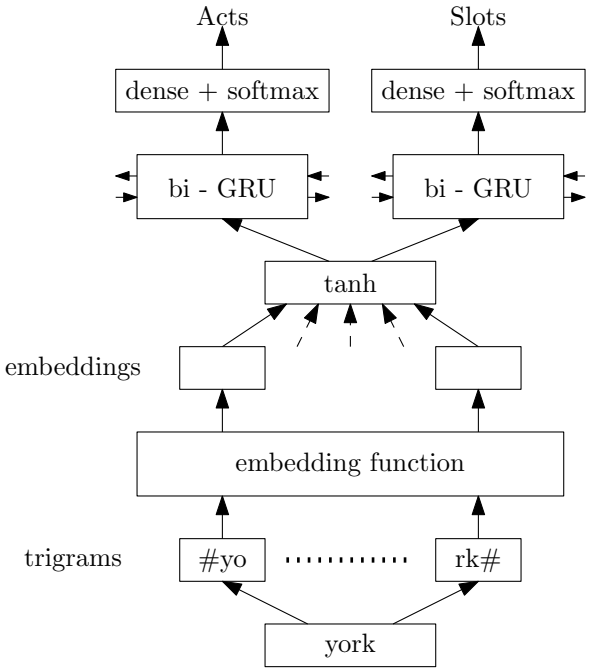


Figure 4: Illustration of the NLU model for slots and acts prediction, taking input words and outputting labels for slots and acts. The model splits into slots-specific and acts-specific predictors after the word embedding layer, which computes a non-linearity on top of the per-word sum of character trigram embeddings.

of the utterance, a pair of tags – one for the act and one for the slot. The model splits into two parts: one part is trained to predict dialogue acts and the other part is trained to predict slot types (at this stage, we predict either a slot type or an *O* tag). These two parts share an embedding matrix for the input character trigrams. Note that the model only predicts IOB tags for slots whose values can be found in the text. Therefore, the prediction for slots such as `intent` or vicinities and amenities is not evaluated for this simple baseline.

The two parts of the model are trained simultaneously, using a modified categorical crossentropy loss for either set of outputs. We modify the loss to ignore *O* labels that are already predicted correctly by the model. We introduce this modification because *O* labels are far more frequent than

Table 5: F1 scores for the NLU baseline (mean and standard deviation).

Dialogue Acts	Slots
0.78 ± 0.05	0.79 ± 0.04

Table 6: Performance of the Frame Tracking Baselines (mean and standard deviation).

	Rule-Based	Random
Frame Creation	0.49 ± 0.03	0.47 ± 0.02
Frame Identification	0.24 ± 0.02	0.18 ± 0.02

other labels, and not limiting their contribution to the loss causes the model to get stuck into a mode where it predicts *O* labels for every word. The loss for the two parts of the model are added together, and the combined objective is optimized using the ADAM optimizer (Kingma and Ba, 2014).

We provide F1 scores for acts and slots for this model in Table **??**. We report average and standard deviation over ten leave-one-user-out splits of the Frames dataset. We had a total of 11 participants in the user role during data collection. Two participants performed significantly fewer dialogues than the others. We merged the dialogues generated by these two participants (ids U21E41CQP and U23KPC9QV). For each of the resulting 10 users, we randomly split the combined dialogues of the nine others into training (80%) and validation (20%), and then tested on the dialogues from the held-out user.

8.2 Frame Tracking

The rule-based frame tracker takes as input the `acts_without_refs` annotation and the values set in the existing frames. We write $f[k]$ to denote the value of slot k in frame f. According to hand-designed rules, the frame tracker predicts the `ref` tags (for frame identification, see **??**) and frame creations. For an act $a(k{=}v)$ in frame f, the following rules are used:

- *Create and switch to a new frame* if a is `inform` and $f[k]$ is set, but v does not match $f[k]$.
- *Switch to frame g* if a is `switch_frame` and $g[k]$ matches v. If no match is found, switch to the most recently created frame.[4]
- *Assign `ref` to frame g* if a can have a `ref` tag, and $g[k]$ matches v. The most recently created frame is used in ambiguous cases. If no match is found, assign `ref` to the current frame.

We compare this baseline to random performance. For random performance, for each (dialogue act, slot type) combination, we computed priors on the corpus for each time the user would refer to the current frame *vs* a previous one. We sampled whether each slot was referring to the current frame or another one based on that prior, and if it referred to another frame, the frame number for that other frame was sampled uniformly from the list of frames created so far.

?? presents results for these baselines. We report results over 10 runs following the same method as for the NLU model. **??** shows that the rule-based baseline only performs slightly better than random on frame identification and performs similarly on frame creation. In general, these results suggest that simple rules are far from adequate for frame tracking and require more in-depth analysis of the user text.

9 Conclusion and Future Work

In this paper, we introduced the Frames dataset: a corpus of human-human dialogues for researching the role of memory in goal-oriented dialogue systems. We propose this dataset to study memory in goal-oriented dialogue systems. We formalized the frame tracking task, which extends the state

[4]This is a reasonable assumption since this case often happens when a wizard makes an offer and the user talks about this offer

tracking task to a setting where several semantic frames are simultaneously tracked throughout the dialogue. We proposed a baseline for this task and we showed that there is a lot of room for improvement. Finally, we showed that Frames can be used to research other interesting aspects of dialogue such as the use of memory for dialogue management and information presentation through natural language generation. We propose adding memory as a first milestone towards goal-oriented dialogue systems that support more complex dialogue flows. Future work will consist of proposing models for frame tracking as well as proposing a methodology to scale up data collection and annotation.

REFERENCES

Arnold, S., F. A. Gers, T. Kilias, and A. Löser (2016). "Robust Named Entity Recognition in Idiosyncratic Domains". In: arXiv: 1608.06757 [cs.CL].

Bordes, Antoine and Jason Weston (2016). "Learning End-to-End Goal-Oriented Dialog". In: arXiv: 1605.07683 [cs.CL].

El Asri, Layla, Rémi Lemonnier, Romain Laroche, Olivier Pietquin, and Hatim Khouzaimi (2014). "NASTIA: Negotiating Appointment Setting Interface". In: *Proc. of LREC*.

Fatemi, Mehdi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman (2016). "Policy Networks with Two-Stage Training for Dialogue Systems". In: *Proc. of SIGDIAL*.

Gašić, M., M. Henderson, B. Thomson, P. Tsiakoulis, and S. Young (2012). "Policy optimisation of POMDP-based dialogue systems without state space compression". In: *Proc. of SLT*.

Grice, Paul (1989). "Studies in the Way of Words". In: Harvard University Press, Cambridge MA. Chap. Logic and Conversation.

Henderson, Matthew (2015). "Machine Learning for Dialog State Tracking: A Review". In: *First International Workshop on Machine Learning in Spoken Language Processing*.

Henderson, Matthew, Blaise Thomson, and Jason D. Williams (2014). "The Second Dialog State Tracking Challenge". In: *Proc. of SIGDIAL*.

Kelley, John F. (1984). "An iterative design methodology for user-friendly natural language office information applications". In: *ACM Transaction on Information Systems*.

Kingma, D. and J. Ba (2014). "Adam: A Method for Stochastic Optimization". In: arXiv: 1412.6980 [cs.LG].

Laroche, Romain, Ghislain Putois, Philippe Bretier, and Bernadette Bouchon-Meunier (2009). "Hybridisation of expertise and reinforcement learning in dialogue systems." In: *Proc. of Interspeech*.

Laroche, Romain, Ghislain Putois, Philippe Bretier, Martin Aranguren, Julia Velkovska, Helen Hastie, Simon Keizer, Kai Yu, Filip Jurčíček, Oliver Lemon, and Steve Young (2011). *Report D6.4 : Final Evaluation of CLASSIC TownInfo and Appointment Scheduling systems*. Tech. rep. CLASSIC Project.

Lemon, Oliver and Olivier Pietquin (2007). "Machine Learning for Spoken Dialogue Systems". In: *Proc. of Interspeech*, pp. 2685–2688.

Lemon, Oliver, Kallirroi Georgila, James Henderson, and Matthew Stuttle (2006). "An ISU Dialogue System Exhibiting Reinforcement Learning of Dialogue Policies: Generic Slot-filling in the TALK In-car System". In: *Proc. of EACL*.

Moe, Wendy W. and Peter S. Fader (2001). "Uncovering Patterns in Cybershopping". In: *California Management Review*.

Pietquin, Olivier, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet (2011). "Sample-efficient batch reinforcement learning for dialogue management optimization". In: *ACM Transaction on Speech and Language Processing* 7.3, pp. 1–21.

Raux, Antoine, Brian Langner, Allan Black, and Maxine Eskenazi (2003). "LET'S GO: Improving Spoken Dialog Systems for the Elderly and Non-natives". In: *Proc. of Eurospeech*.

Rieser, Verena, Ivana Kruijff-Korbayov, and Oliver Lemon (2005). "A corpus collection and annotation framework for learning multimodal clarification strategies". In: *Proc. of SIGDIAL*.

Singh, Satinder, Michael Kearns, Diane Litman, and Marilyn Walker (2002). "Optimizing dialogue management with reinforcement learning: experiments with the NJFun System". In: *Journal of Artificial Intelligence Research* 16, pp. 105–133.

Walker, Marilyn A., Jeanne C. Fromer, and Shrikanth Narayanan (1998). "Learning Optimal Dialogue Strategies: A Case Study of a Spoken Dialogue Agent for Email". In: *Proc. of COLING/ACL*, pp. 1345–1352.

Wen, Tsung-Hsien, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve J. Young (2016). "A Network-based End-to-End Trainable Task-oriented Dialogue System". In: arXiv: 1604.04562 [cs.CL].

Williams, Jason D. and Geoffrey Zweig (2016). "End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning". In: arXiv: 1606.01269 [cs.CL].

A Database Overview

Table 7: Searchable fields in the database of packages

Field	Description
PRICE_MAX	Maximum price the user is willing to pay
PRICE_MIN	Minimum price defined by the user
DESTINATION_CITY	Destination city
MAX_DURATION	Maximum number of days for the trip
NUM_ADULTS	Number of adults
NUM_CHILDREN	Number of children
START_DATE	Start date for the trip
END_DATE	End date for the trip
ARE_DATES_FLEXIBLE	Boolean value indicating whether or not the user's dates are flexible. If True, then the search is broadened to 2 days before START_DATE and 2 days after END_DATE.
ORIGIN_CITY	Origin city

Table 8: Non-searchable fields in the database of packages

Field	Description
Global Properties	
PRICE	Price of the trip including flights and hotel
DURATION	Duration of the trip
Hotel Properties	
NAME	Name of the hotel
COUNTRY	Country where the hotel is located
CATEGORY	Rating of the hotel (in number of stars)
CITY	City where the hotel is located
GUEST_RATING	Rating of the hotel by guests (in number of stars)
BREAKFAST, PARKING, WIFI, GYM, SPA	Boolean value indicating whether or not the hotel offers this amenity.
PARK, MUSEUM, BEACH, SHOPPING, MARKET, AIRPORT, UNIVERSITY, MALL, CATHEDRAL, DOWNTOWN, PALACE, THEATRE	Boolean value indicating whether or not the hotel is in the vicinity of one of these.
Flights Properties	
SEAT	Seat type (economy or business)
DEPARTURE_DATE_DEP	Date of departure to destination
DEPARTURE_DATE_ARR	Date of return flight
DEPARTURE_TIME_DEP	Time of departure to destination
ARRIVAL_TIME_DEP	Time of arrival to destination
DEPARTURE_TIME_ARR	Time of departure from destination
ARRIVAL_TIME_ARR	Time of arrival to origin city
DURATION_DEP	Duration of flight to destination
DURATION_ARR	Duration of return flight

B Dialogue Acts and Slot Types

Table 9: List of dialogue acts in the annotation of Frames

Dialogue Act	Speaker	Description
inform	User/Wizard	Inform a slot value
offer	Wizard	Offer a package to the user
request	User/Wizard	Ask for the value of a particular slot
switch_frame	User	Switch to a frame
suggest	Wizard	Suggest a slot value or package that does not match the user's constraints
no_result	Wizard	Tell the user that the database returned no results
thankyou	User/Wizard	Thank the other speaker
sorry	Wizard	Apologize to the user
greeting	User/Wizard	Greet the other speaker
affirm	User/Wizard	Affirm something said by the other speaker
negate	User/Wizard	Negate something said by the other speaker
confirm	User/Wizard	Ask the other speaker to confirm a given slot value
moreinfo	User	Ask for more information on a given set of results
goodbye	User/Wizard	Say goodbye to the other speaker
request_alts	User	Ask for other possibilities
request_compare	User	Ask the wizard to compare packages
hearmore	Wizard	Ask the user if she'd like to hear more about a given package
you_are_welcome	Wizard	Tell the user she is welcome
canthelp	Wizard	Tell the user you cannot answer her request
reject	Wizard	Tell the user you did not understand what she meant

Table 10: List of slot types not present in the database

Slot Type	Description
count	Number of different packages
count_amenities	Number of amenities
count_name	Number of different hotels
count_dst_city	Number of destination cities
count_seat	Number of seat options (for flights)
count_category	Number of star ratings
id	Id of the frame created (for offers and suggestions)
vicinity	Vicinity of the hotel
amenities	Amenities of the hotel
ref_anaphora	Words used to refer to a frame *e.g.*, "the second package'
impl_anaphora	Used when a slot type is not specifically mentionned *e.g.*, "What is the price for Rio?"..."And for Cleveland?"
ref	Id of the frame that the speaker is referring to
read	Reads slot values specified in another frame and writes them in the current frame
write	Writes slot values in a given frame
intent	User intent (*e.g.*, book)
action	Wizard action (*e.g.*, book)

Label-dependency coding in Simple Recurrent Networks for Spoken Language Understanding

Marco Dinarelli, Vedran Vukotic, Christian Raymond

Label-dependency coding in Simple Recurrent Networks for Spoken Language Understanding

Marco Dinarelli [1], *Vedran Vukotic*[2,3], *Christian Raymond*[2,3]

[1]Lattice, CNRS, ENS Paris, Université Sorbonne Nouvelle - Paris 3
PSL Research University, USPC (Université Sorbonne Paris Cité)
[2]INSA Rennes, France
[3]INRIA/IRISA, Rennes, France

marco.dinarelli@ens.fr,{vedran.vukotic,christian.raymond}@irisa.fr

Abstract

Modelling target label dependencies is important for sequence labelling tasks. This may become crucial in the case of Spoken Language Understanding (SLU) applications, especially for the slot-filling task where models have to deal often with a high number of target labels. Conditional Random Fields (CRF) were previously considered as the most efficient algorithm in these conditions. More recently, different architectures of Recurrent Neural Networks (RNNs) have been proposed for the SLU slot-filling task. Most of them, however, have been successfully evaluated on the simple ATIS database, on which it is difficult to draw significant conclusions. In this paper we propose new variants of RNNs able to learn efficiently and effectively label dependencies by integrating label embeddings. We show first that modeling label dependencies is useless on the (simple) ATIS database and unstructured models can produce state-of-the-art results on this benchmark. On ATIS our new variants achieve the same results as state-of-the-art models, while being much simpler. On the other hand, on the MEDIA benchmark, we show that the modification introduced in the proposed RNN outperforms traditional RNNs and CRF models.
Index Terms: recurrent neural networks, label dependencies,spoken language understanding, slot filling, ATIS, MEDIA

1. Introduction

In classical Spoken Language Understanding (SLU) systems, one of the key tasks is to label words with lexical semantics. For example, in the sentence "I want a Chinese restaurant near Tour-Eiffel", the word "Chinese" should be labeled as the food-type of a restaurant, and "Tour-Eiffel" as a relative place in Paris. Many algorithms have been investigated for slot tagging: SVM [1], HVS [2], Machine translation models, Finite State Transducers and Conditional Random Fields [3]. Recently, also Neural Networks have been investigated [4, 5, 6]. Neural networks have the advantage to come together with new text representations. Discrete items in the text are mapped into vectors, named often embeddings, using popular word embedding methods [7, 8]. This representation has several advantages, the most salient one is to make words that are syntactically or semantically related, close to each-other in the representation space. This ability is particularly useful for several tasks, but not particularly for SLU on the ATIS task because the database of this task already provides important clusters (*e.g.* city names, airline names, places, *etc.*). Anyway, this representation appears to be noise robust [6]. Neural networks are not dedicated sequence-labelling algorithms and many efforts have been made to improve their ability to process sequences. Recurrent Neural Network (RNN) architectures like LSTM [9] have been investigated to better model long range dependencies in the observations. RNNs like Jordan architectures have been proposed to better model target label sequences [5]. In this work we focus on modeling the target label dependencies. We propose a modification of the Jordan architecture by introducing an embedding of the previous predicted target labels. This simple modification results in a RNN very effective at learning label dependencies, and allows improvements over the other RNNs proposed in the literature as well as state-of-the-art CRF models.

Unfortunately, the public widely used benchmark ATIS [10, 11] is not very challenging and a wide variety of methods provides similar (very good) results, including methods that are not specifically designed for sequence labeling. These last methods fail [12, 3, 6] when evaluated on MEDIA [13], another public SLU database where modeling label-dependencies is crucial to obtain good results. This indicates that conclusions formulated from results obtained on the ATIS database are not particularly strong. We will provide in this work results from experiments conducted on both databases.

Results on the MEDIA task, in particular, provide evidence to conclude that: i) the proposed variant of RNN using label embeddings outperforms by a large margin the standard Jordan RNN, and thus also the Elman RNN which is less effective than the Jordan model [6]; ii) by simply using label embeddings, RNNs can model label dependencies more effectively compared to RNNs using a CRF neural layer like the one used in [5, 14, 15]; iii) the proposed variant of RNN provides the new state-of-the-art results on both the ATIS and the MEDIA tasks.

We particularly stress on results obtained on the MEDIA task because, as it has been shown and as we will show in this paper, only models keeping label-dependencies into account obtain good results on this task, which means in turn that these models are the most suited for sequence labeling.

2. Datasets

In our experiments we used two datasets: ATIS and MEDIA. ATIS is a publicly available corpus used in the early nineties for SLU evaluation. MEDIA has been collected in the last decade and is available through ELRA since 2008.

2.1. ATIS

The Air Travel Information System (ATIS) task [10] is dedicated to provide flight information. The semantic representation used is frame based. The SLU goal is to find the good frame and fill the corresponding slots.

The training set consists of 4978 utterances selected from the Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora while the ATIS test set contains both the ATIS-3 NOV93 and DEC94 datasets. Please see [10] for more details.

2.2. MEDIA

The research project MEDIA [13] evaluates different SLU models of spoken dialogue systems dedicated to provide tourist information. A corpus made of 1250 French dialogues has been collected by ELDA, following a Wizard of Oz protocol: 250 speakers have followed 5 hotel reservation scenarios. This corpus has been transcribed manually and annotated with concepts from a rich semantic ontology. The representation is based on the definition of concepts that can be associated with 3 kinds of information. First a concept is defined by a label and a value; for example with the concept date, the value 2006/04/02 can be associated. Second, a specifier can be attached to a concept in order to link the concept, and to go from flat concept/value representations to hierarchical ones; for example, the concept date can be associated with the specifiers *reservation* and *begin* to specify that this date is the beginning date of a hotel reservation. Third, modal information is added to each concept (positive, affirmative, interrogative or optional). Table 1 shows an example of dialogue turn from the MEDIA corpus with only concept-value information. The first column contains the segment identifier in the message, the second column shows the chunks W^c supporting the concept c of the third column. In the fourth column the value of the concept c in the chunk W^c is displayed. The MEDIA semantic dictionary contains 83 concept labels, 19 specifiers and 4 types of modal information. In this study we will focus only on concept extraction. No specifiers, values or modal information are considered, so the tag-set consists of 83 labels. The MEDIA corpus is split into 3 parts. The first part (720 dialogues, 12K messages) is used for training the models, the second (79 dialogues, 1.3K message) is used for selecting the best system, and the third part (200 dialogues, 3.4K message) is used as test.

3. Simple Recurrent Networks

3.1. Elman network

Elman networks have been proposed in [16] and are defined as:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned}$$

n	W^c	c	**value**
1	yes	answer	yes
2	the	RefLink	singular
3	hotel	BDObject	hotel
4	which	null	
5	price	object	payment-amount
6	is below	comparative-payment	below
7	fifty five	payment-amount-int	55
8	euros	payment-currency	euro

Table 1: *Example of message with concept+value information. The original French transcription is: "oui l'hôtel dont le prix est inférieur à cinquante cinq euros"*

where x_t is the input vector, h_t the hidden layer vector, y_t the output vector, W, U and b are parameter matrices and vector, σ_h and σ_y are activation functions.

Elman RNNs use the previous hidden state as contextual information (h_{t-1}), but they don't use any information about previous predicted labels. For this reason, while they have shown good results as any other model on the ATIS task, they are among the least effective neural models on the MEDIA task [6].

3.2. Jordan network

Jordan networks have been proposed in [17] and are defined as follows:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned}$$

This model uses previous predicted labels as contextual information to predict the current label. However previous labels are provided as input to the hidden layer either as raw network outputs, or as *one-hot* representations[1]. Raw network outputs are the output of the softmax output layer [18], which computes a probability distribution over all possible labels defined in the task. One-hot representations can be computed from raw outputs putting 1 at the position corresponding to the maximum probability, and zero anywhere else.

3.3. eJordan

The improved RNN proposed in this paper is based on a similar idea as the one described in [19, 20].

In this variant predicted labels are mapped into embeddings, the same way as words. Word embeddings are stored in a matrix $E_w \in \mathbb{R}^{|D_w| \times N}$, where $|D_w|$ is the size of the word dictionary, N is the size chosen for the embeddings. In the same way label embeddings are stored in a matrix $E_l \in \mathbb{R}^{|D_l| \times N}$, where $|D_l|$ is the size of the label dictionary, which is also the size of the network output y_t.

In order to keep notation lighter, we indicate with y_t both the raw output of the network (computed by the softmax) and the one-hot representation of the label. The latter can be seen in turn as the index of the corresponding label. With this formalism, the input of the hidden layer is $x_t = E_w(w_t)$, like in the other RNNs and w_t is the word to be labeled at position t in a sequence, and $z_t = E_l(y_{t-1})$, which is the embedding of the previous predicted label y_{t-1}. The hidden and output layers are then computed as:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h z_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned}$$

As we can see thus, the only difference between the proposed variant and a Jordan RNN is that in our variant the label used as contextual information is provided as an embedding. For this reason we name our variant *eJordan*, for *embedded Jordan* RNN.

3.4. Bi-eJordan

Since a couple of years, RNNs are provided as bidirectional models [21, 5]. These models allow to keep into account both past and future information to predict the current label.

[1] *one-hot* representations are sparse vectors representing dictionary entries. The entry having index i in a dictionary V of size $|V|$, is represented with a vector of size $|V|$ which is zero everywhere, except at position i where it has value 1.

We provide also our eJordan variant as bidirectional model. As described in [21], in this variant we use first a backward model to predict labels in backward direction, that is from the end to the begin of a sentence. Such labels are then used as future predicted labels by a bidirectional model, which processes sentences in forward direction and computes its final output as the geometric mean of the forward and backward decisions:

$$y_t = \sqrt{y_t^f \odot y_t^b}$$

where y_t^f is the output of the forward model, y_t^b is the output of the backward model, and $\odot$ is the element-wise product.

4. Experimental protocol

We will compare our eJordan model against several competitor architectures:

- the basic Multi-Layer Perceptron (MLP) with softmax output layer (no recurrence), also named Feed-Forward Neural Network (FFNN) in the literature, in order to show the importance of modeling target label dependencies. This is called MLP+SOFT in later.
- the Jordan RNN: in order to compare the difference between one-hot and fine tuned embedded representation
- a MLP with CRF layer on top instead of the softmax (and obviously applied on sequences), called later MLP+CRF

A boosting based [22] system is also presented. This system is a local classification model not designed at all for sequence labeling like MLP+SOFT and uses only symbolic features (no embedding). This model and the MLP+SOFT are used to illustrate the difference in results that can be obtained when modeling label dependencies and sequences is important, like in MEDIA, with respect to the ATIS task, where any of the described models reaches state-of-the-art results.

4.1. Features and configuration

One of the objective of this paper is to fairly compare systems and their ability to model target label dependencies. So, usual and reasonable configurations previously published are used and fixed for all neural systems. Thus they are evaluated and compared to each other in the same conditions, which are:

- observation: word or class if the word belongs to a semantic class (*e.g.* CITY_NAME)
- size of observation window: 7 for MEDIA and 11 for ATIS
- hidden layer: 200 for MEDIA and 100 for ATIS
- size of the embedding: 200 for MEDIA, 100 for ATIS

All systems have been ran 10 times for 30 epochs. Averages of 10 results will be provided in terms of:

- F1 measure computed by the script conlleval[2]: this measure tends to show how good is the segmentation of concepts over surface forms.
- Concept Error Rate (CER) measure computed by sclite[3] on the target label level: this measure tends to show how good is the concept recognition in the perspective of using SLU in spoken dialog systems.

[2] `http://www.cnts.ua.ac.be/conll2000/chunking/output.html`

[3] `http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm`

Algo	Parameters	conlleval F1	CER sclite
ATIS			
boosting	i=2500 d=2	95.69	5.0
MLP+SOFT	151,488	95.67 (0.07)	5.00 (0.09)
MLP+CRF	157,606	95.45 (0.11)	5.28 (0.12)
Bi-Jordan	338,376	95.69 (0.07)	4.97 (0.07)
Bi-eJordan	340,576	95.74 (0.02)	4.91 (0.03)
MEDIA			
boosting	i=3500 d=3	77.11	18.2
MLP+SOFT	642,135	83.60 (0.16)	12.71 (0.21)
MLP+CRF	660 360	86.34 (0.19)	10.96 (0.14)
Bi-Jordan	1,399,882	86.15 (0.09)	11.12 (0.11)
Bi-eJordan	1,743,082	86.97 (0.12)	10.34 (0.19)
BiGru+CRF	2,328,360	86.69 (0.13)	10.13 (0.21)

Table 2: *Performances of the various algorithms on ATIS and MEDIA. Averaged (over 10 runs) F1 measure (%), Concept Error Rate (%) and their respective standard deviations (in parenthesis).*

Boosting systems boost bonsai trees [22], number of iterations (i) and depth (d) of the trees are fixed arbitrarily to some values that provide good results on the used benchmarks (they are used to set our expected low baseline since they don't use neither embeddings nor sequence dedicated mechanisms). This system is doing automatically feature selection, thus a larger context (observation window of 20) is provided and allows improvements.

The MLP+SOFT, Jordan and eJordan models have been implemented in *Octave*[4]. The other neural models have been implemented in *Keras*[5].

5. Results

The first remarkable result in table 2 is the performance of the boosting system: on ATIS, this system is performing very well, as well as all other algorithms despite the fact that it is not using any embedding and has no knowledge about target label dependencies. On the opposite, on MEDIA, this system looks largely ineffective in comparison to the others. These results illustrate clearly the fact that ATIS is not a challenging task. It illustrates also the fact that it is not possible to draw strong conclusions about the fact that an algorithm is better or not than another one: almost every algorithm is able to provide outstanding results on ATIS, and noise may be a better explanation for the slight difference between algorithms than the effectiveness of the algorithm itself [23, 6, 24].

As shown in the example in table 1, MEDIA is a much more challenging task: first, the semantic annotation is richer; second, labels are segmented over multiple words, which can create relatively long label dependencies and increases in practice the number of labels to be recognized to 135 (using the *BIO* segmentation formalism); third, though it is not specifically addressed in this paper, coreference phenomena are annotated in the MEDIA task, making annotation decisions depend on long past contexts. An idea of the difficulty of this task with respect to ATIS is given also by the absolute magnitude of results in table 2 (9-12 F1 points lower than results on ATIS).

Comparing results in table 2 on MEDIA among the neural models, provides evidence of interesting outcomes.

[4] https://www.gnu.org/software/octave/; The code is described at http://www.marcodinarelli.it/software.php and available upon request

[5] https://keras.io

First we note that models integrating increasingly rich information on label dependencies provide increasingly good results: the MLP+SOFT model, which has no label information, is the less effective; the traditional Jordan RNN integrates the previous label as one-hot representation, outperforming by a large margin the MLP model; the MLP+CRF further improves results, showing that it can integrate longer range label information thanks to the global-level probability normalization of CRF; very interestingly, the most effective model on MEDIA is the proposed eJordan as bidirectional variant (Bi-eJordan), which uses label embeddings. Since this variant uses a local decision function (the softmax), from these results we can deduct that the use of label embeddings, together with their combination with word embeddings at the hidden layer, allows RNNs to model more fine label dependency features and word-label interactions than a CRF neural layer, overcoming in fact the limitation of using a local decision function.

Second, eJordan achieves a CER of 10.34 on average, and 10.32 according to more accurate model on the development data on the 10 runs. These results can be compared to [3] on only *attributes* extraction. To the best of our knowledge this is the best result achieved on this task with an individual model[6].

Finally, we give more insights on the behavior of neural models when integrating in different ways and at different degrees contextual label information. For this purpose we simply analyze results in terms of accuracy on void concepts (O) compared to accuracy on all the other concepts ($\neg O$). Indeed the ratio of void concepts is very different on the two tasks addressed in this paper: $35,623$ out of $52,170$ (68,28%) on ATIS, and $33,186$ out of $95,851$ (34,62%) on MEDIA. Also, while in the ATIS corpus there is no segmentation of concepts over multiple words (each concept is instantiated by one token), in the MEDIA task, on the opposite, concepts are segmented over relatively long lexical chunks.

As consequence we expect models not aware of label dependencies to be somehow naive, predicting correctly a larger amount of void concepts (to minimize the risk). This is the consequence of the large representation of this category in the training data, combined to the fact that these models cannot "trade" the decision conducted from word-level information with the one conducted from label-level information. In contrast, the more sophisticated the representation of label context information in the neural model, the more we expect the model to be effective in predicting labels other than the void concept. In these models the bias toward predicting the over-represented class of void concepts can be possibly in contrast with the constraints introduced by label dependencies.

This simple analysis is depicted in table 3. We can see once again that all models perform astonishingly well on ATIS, and even more astonishingly close: all models achieve accuracy close or higher than 99 on void concepts, and higher than 97 on the other concepts.

The same analysis on MEDIA is much more interesting. As expected, the MLP+SOFT model, which is the only one without any contextual label information, achieves a relatively high accuracy on void concepts (the second best), while it performs the worst on the other concepts, and by more than 1 point from the second best (Jordan). We can consider the other 3 neural models addressed in this analysis, as more and more sophisticated in integrating contextual label information, the order being Bi-Jordan, Bi-eJordan and MLP+CRF. The accuracy on non-void concepts reflects indeed this ranking. The Bi-eJordan variant however reaches a better compromise between accuracy on void and non-void concepts, and it is thus the most effective among these 4 neural models in terms of F1 measure and CER[7] (table 2).

Model	Accuracy on O	Accuracy on ¬O
ATIS		
MLP+SOFT	99.01	97.16
MLP+CRF	98.99	97.16
Bi-Jordan	99.01	97.21
Bi-eJordan	99.01	97.27
MEDIA		
MLP+SOFT	95.89	87.98
MLP+CRF	93.73	89.04
Bi-Jordan	96.46	88.01
Bi-eJordan	94.68	88.60

Table 3: *Comparative results of the different neural models described in the paper in terms of accuracy on void concepts (*O*) and all the other concepts (*¬O*). Models with less label-level contextual information are those with higher accuracy, but lower F1 and CER.*

We would like to point out that eJordan and the CRF mechanism must not be considered in mutual exclusion. In [14, 15] we can see actually that the CRF neural layer used so far is somehow complementary to eJordan, in the sense that it does not represent labels as embeddings. The combination of these two models may thus lead to even more sophisticated models.

However the goal of this work is not to produce the best result on the addressed benchmarks, but to propose and compare some label-dependencies aware methods for SLU in a fair way. Of course, better architectures may be easily proposed: for example, using LSTM as hidden layer to better encode long context input may allow further improvements. We started investigating also these more complex models, in particular a bidirectional GRU [25] with a CRF neural layer as output layer. Preliminary results are given in table 2 with the name *BiGRU+CRF*. Also richer inputs may be provided to the networks, *e.g.* word embeddings externally trained on huge amount of data, character-level convolution like in [14, 15], and so on.

6. Conclusion

We proposed in this paper a recurrent neural network architecture to better model target label dependencies in sequence labeling problems for SLU. This architecture, named eJordan, is a slight modification of the Jordan network where the label predicted at time $t-1$ is embedded and injected as input to the network at time t. A bidirectional eJordan network is fairly compared and outperforms traditional competitors, Jordan and MLP+CRF on the MEDIA task. As usual, every methods tends to perform similarly (and very well) on the ATIS dataset.

7. Acknowledgment

This work has been partially funded by the French ANR project Democrat ANR-15-CE38-0008[8]. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GTX Titan X GPU used for this research.

[6]The best absolute result in [3] is 10.2, but it is obtained with a combination of 6 individual models

[7]We recall the reader that F1 measure and CER don't take void concepts into account.

[8]http://www.agence-nationale-recherche.fr/?Projet=ANR-15-CE38-0008

8. References

[1] T. Kudo and Y. Matsumoto, "Chunking with Support Vector Machines," in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, ser. NAACL '01. Stroudsburg, PA, USA: Association for Computational Linguistics, 2001, pp. 1–8. [Online]. Available: http://dx.doi.org/10.3115/1073336.1073361

[2] Y. He and S. Young, "Semantic Processing using the Hidden Vector State Model," *Computer Speech and Language*, vol. 19, pp. 85–106, 2005.

[3] S. Hahn, M. Dinarelli, C. Raymond, F. Lefèvre, P. Lehnen, R. De Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing Stochastic Approaches to Spoken Language Understanding in Multiple Languages," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 6, pp. 1569–1583, August 2011.

[4] 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, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, 2013, pp. 3771–3775. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2013/i13_3771.html

[5] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 23, no. 3, pp. 530–539, 2015.

[6] V. Vukotic, C. Raymond, and G. Gravier, "Is it time to switch to word embedding and recurrent neural networks for spoken language understanding?" in *InterSpeech*, Dresde, Germany, September 2015.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *International Conference on Learning Representations*, 2013.

[8] R. Lebret and R. Collobert, "Word Embeddings through Hellinger PCA."

[9] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 189–194.

[10] D. A. 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 *HLT*, 1994, pp. 43–48.

[11] C. Raymond and G. Riccardi, "Generative and Discriminative Algorithms for Spoken Language Understanding," in *InterSpeech*, Antwerp, Belgium, August 2007, pp. 1605–1608.

[12] S. Hahn, P. Lehnen, C. Raymond, and H. Ney, "A comparison of various methods for concept tagging for spoken language understanding," in *Proceedings of the Language Resources and Evaluation Conference*, Marrakech, Morocco, May 2008.

[13] H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, and D. Mostefa, "Semantic Annotation of the French Media Dialog Corpus," in *InterSpeech*, Lisbon, September 2005.

[14] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016.

[15] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[16] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[17] M. I. Jordan, "Serial order: A parallel, distributed processing approach," in *Advances in Connectionist Theory: Speech*, J. L. Elman and D. E. Rumelhart, Eds. Hillsdale, NJ: Erlbaum, 1989.

[18] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012. [Online]. Available: http://arxiv.org/abs/1206.5533

[19] M. Dinarelli and I. Tellier, "New recurrent neural network variants for sequence labeling," in *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics*. Konya, Turkey: Lecture Notes in Computer Science (Springer), Avril 2016.

[20] D. Bonadiman, A. Severyn, and A. Moschitti, "Recurrent context window networks for italian named entity recognizer," *Italian Journal of Computational Linguistics*, vol. 2, 2016. [Online]. Available: http://disi.unitn.it/moschitti/since2013/2016_IJCoL_Moschitti_NER-CNNs-IT.pdf

[21] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Trans. Sig. Proc.*, vol. 45, no. 11, pp. 2673–2681, nov 1997. [Online]. Available: http://dx.doi.org/10.1109/78.650093

[22] A. Laurent, N. Camelin, and C. Raymond, "Boosting bonsai trees for efficient features combination : application to speaker role identification," in *InterSpeech*, Singapour, September 2014. [Online]. Available: http://bonzaiboost.gforge.inria.fr

[23] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent Neural Networks for Language Understanding," in *InterSpeech*. Interspeech, August 2013. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=200236

[24] G. Tur, D. Hakkani-Tur, and L. Heck, "What is left to be understood in ATIS?" in *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 2010, pp. 19–24.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, 2014.

Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding

Grégoire Mesnil [1,3], *Xiaodong He*[2], *Li Deng*,[2] *and Yoshua Bengio* [1]

[1] University of Montréal, Québec, Canada
[2] Microsoft Research, Redmond, WA, USA
[3] University of Rouen, France

{gregoire.mesnil|yoshua.bengio}@umontreal.ca, {xiaohe|deng}@microsoft.com

Abstract

One of the key problems in spoken language understanding (SLU) is the task of slot filling. In light of the recent success of applying deep neural network technologies in domain detection and intent identification, we carried out an in-depth investigation on the use of recurrent neural networks for the more difficult task of slot filling involving sequence discrimination. In this work, we implemented and compared several important recurrent-neural-network architectures, including the Elman-type and Jordan-type recurrent networks and their variants. To make the results easy to reproduce and compare, we implemented these networks on the common Theano neural network toolkit, and evaluated them on the ATIS benchmark. We also compared our results to a conditional random fields (CRF) baseline. Our results show that on this task, both types of recurrent 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, outperforming a CRF-based baseline by 14% in relative error reduction.

Index Terms: spoken language understanding, word embeddings, recurrent neural network, slot filling

1. Introduction

A major task in speech understanding or spoken language understanding (SLU) is to automatically extract semantic concept, 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. Despite many years of research, the slot filling task in SLU is still a challenging problem, in parallel with the intent determination task [23][27].

Until fairly recently, the main technical approaches to solving the slot filling problem in SLU included generative modeling, such as HMM/CFG composite models [25], and discriminative or conditional modeling such as conditional random fields (CRF) [13][26]. A few years ago, a new approach emerged for advancing speech recognition, based on deep learning, which involves many layers of nonlinear information processing in deep neural networks [11][6]. Subsequently these techniques have been applied to intent determination or semantic utterance classification tasks of SLU [24][7]. Deep learning has also been successfully applied to a number of other human language technology areas including language modeling [16][21], especially with the use of the naturally deep architecture of recurrent neural networks [15]. Among these progresses, one important advance is the invention of word embeddings [2], successfully projecting very-high-dimensional, sparse vector for word representations into a low-dimensional, dense vector representation for a variety of natural language tasks [3][4][15].

In light of the recent success of these methods, especially the success of recurrent neural networks for language modeling [15], we carried out an in-depth investigation of recurrent neural networks for the slot filling task of SLU. In this work, we implemented and compared several important recurrent neural network architectures, e.g., the Elman-type networks [8][15] and Jordan-type networks [12] and their variations. To make the results easy to reproduce and rigorously comparable, we implemented these models using the common Theano neural network toolkit [1], and evaluated 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 of both past and future dependencies among slots works best.

2. The Slot Filling Task

Semantic parsing of input utterances in SLU typically consists of three tasks: domain detection, intent determination, and slot filling. Originating from call routing systems, domain detection or intent determination tasks are typically treated as semantic utterance classification. Slot filling is typically treated as a sequence classification problem after semantic templates for concept classes or "slots" are defined.

An example sentence is provided in Table 1, 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*							

Table 1. ATIS utterance example IOB representation

For the slot filling task, the input is the sentence consisting of a sequence of words, and the output is a sequence of slot/concept IDs, one for each word. Traditionally, one of the most successful approaches for slot filling is the conditional random field (CRF) [13] and its variants. I.e., 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:

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

where

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

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 $2d + 1$.

3. Using RNNs for Slot Filling

3.1. Word embeddings

As an alternative to N-gram models, researchers came up with several different techniques based on learning Euclidean space structures for words. A real-valued embedding vector is associated with each word, and these embeddings are usually trained in an unsupervised way on a large corpus of natural language, e.g. Wikipedia. The architecture of these models can vary from shallow neural nets (NN) [19] or convolutional nets (SENNA) [4] to recurrent neural nets (RNN) [15]. The learned word embedding shows good generalization properties across many common natural language processing (NLP) tasks [4]. The neural network architectures evaluated in this paper are based on such word embeddings.

3.2. Short-term dependencies captured using a word context window

Without considering a temporal feedback, the neural network architecture corresponds to a simple feed-forward multi-layer perceptron (MLP), e.g., with a hidden layer and sigmoid activations. To capture short-term temporal dependencies in this setting, one can use a word-context window. With each word mapped to an embedding vector, the word-context window is the ordered concatenation of word embedding vectors. Here is an example of constructing the input vector with the word context window of size 3:

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

$$'from' \rightarrow x_{from} \in \mathbb{R}^d$$

$$w(t) \rightarrow x(t) = [x_{flights}, x_{\boldsymbol{from}}, x_{Boston}] \in \mathbb{R}^{3d}$$

In the example, $w(t)$ is the 3-word context window around the *t*-th word 'from', x_{from} is the embedding vector of the word 'from', and d is the dimension of the embedding vector. Correspondingly, $x(t)$ is the ordered concatenated word embeddings vector for the words in $w(t)$.

In the feed-forward NN [11], the raw input vector *x* is first fed into the hidden layer *h*. After a non-linear transformation, the output of the hidden layer *h* is then fed into the output layer to generate the final output *y*.

3.3. Two types of RNN architectures

At the highest level of complexity for slot filling, one has to take into account the slot/concept dependencies (sequences of labels) beyond the words surrounding the word of interest (the context word window that captures short-term dependencies). Here we first describe two variants of RNNs for modeling slot sequences: the Elman-type RNN [8] and the Jordan-type RNN [12]. Using RNNs to model long-term dependencies will be presented in the next sub-section. In contrast to a feed-forward NN, in the Elman-type RNN, the output from the hidden layer at time *t*-1 is be kept and fed back to the hidden layer at time *t*, together with the raw input $x(t)$ for time *t*. This can be interpreted as having an additional set of virtual "context nodes", where there are connections from the hidden layer to these context nodes fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion, and then a parameter updating rule is applied, taking into account the influence of past states through the recurrent connections. In this way, the context nodes always maintain a copy of the previous values of the hidden nodes, since these propagate through the recurrent connections from time *t*-1, before the updating rule is applied at time *t*. Thus the network can maintain and learn a sort of state summarizing past inputs, allowing it to perform tasks such as sequence-prediction that are beyond the power of a standard feed-forward NN. Precisely, dynamics of the Elman-type RNN can be represented mathematically by

$$h(t) = f\big(Ux(t) + Vh(t-1)\big)$$

where we use the sigmoid function at the hidden layer:

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

and a softmax function at the output layer:

$$y(t) = g\big(Wh(t)\big), \qquad where \;\; g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

where *U* and *V* are weight matrices between the raw input and the hidden nodes, and between the context nodes and the hidden nodes, respectively, while *W* is the output weight matrix. The input vector $x(t)$ has as dimensionality that of the word embeddings d multiplied by the number of words in the context window. $h(t)$ corresponds to the hidden layer, and $y(t)$ has as dimensionality the number of classes; i.e., 127 for the ATIS slot filling task.

Jordan-type RNNs are similar to Elman-type networks, except that the context nodes are fed from the output layer instead of from the hidden layer. The context nodes in a Jordan-type network are also referred to as the state layer. The difference between Elman and Jordan-type networks appears only in the hidden layer input:

$$h(t) = f(Ux(t) + Vy(t-1))$$

3.4. Long-term dependencies captured using a RNN

To capture dependencies beyond the input window, we need to exploit the time-connection feedback, giving rise to the RNN architectures. Learning long-term dependencies with RNNs raises an optimization problem known as the vanishing gradient problem. Indeed, capturing longer time dependencies correspond to training a deeper model when the RNN is unfolded in time (i.e., each time instance of a layer being a separate layer of a deep net). Rather than training classical RNNs in this way, we can directly provide some of the past information from different time steps. Instead of relying only on learning through one-step recurrences to capture context, one can combine recurrence with the idea of input window. This is achieved by feeding the network with concatenation of the *T* previous time steps vectors (from the output layer as in the Jordan-type network or the hidden layer as in the Elman-type network) in addition to the use of word context windows. This also provides a way to obtain the most accurate number of time steps to consider for a particular task. In the case of Elman-type networks, the feedback from the output layer leads to the following backward model (predicting from the future to the past) and forward model (predicting from the past to the future):

$$h_{backward}(t) = f(Ux(t) + \sum_{k=1}^{T} V_k h(t+k))$$

and

$$h_{forward}(t) = f(Ux(t) + \sum_{k=1}^{T} G_k h(t-k))$$

Further, since having only backward or forward time dependencies uses only partial information available, it would be helpful to consider both past and future dependencies together. Bi-directional Elman-type RNNs have been studied in the past [18] and in this paper, we consider variants of bi-directional Jordan-type RNNs:

$$h_{bidir}(t) = f(Ux(t) + \sum_{k=1}^{T} V_k y_b(t+k) + \sum_{k=1}^{T} G_k y_f(t-k))$$

where y_b and y_f denote the output of a backward model and a forward model in the Jordan-type RNN, respectively.

3.5. Learning methods

3.5.1. Fine-tuning word embedding

Once the word embeddings have been learned in an unsupervised fashion [3][15], it is possible to fine-tune them during supervised training on the task of interest. Actually, this is double-edged: the model could fit the task better but the risk of overfitting may arise. We compare both cases experimentally in Section 4.

3.5.2. Sentence-level and word-level gradient descents

The average length of a sentence in the ATIS data set is about 15 words. With such relatively long inputs, training RNN models could be tricky if updates are done online (i.e., after each word). This is because the predictions at the current word have been made with model parameters that are no longer current, and the sequence of predictions does not correspond to the one that could be performed with a fixed parameter set. For instance, if we want to predict or perform an update at the 17th slot of a sentence with a forward RNN model, we would have to re-compute all the values from the beginning of the sentence in order to get "correct" predictions consistent with the current model parameters.

For training the Elman-type RNN, one option to prevent the above problem is to perform mini-batch gradient descent with exactly one sentence per mini-batch. For a given sentence, we perform one pass that computes the mean loss for this sentence and then perform a gradient update for the whole sentence. This approach performs well even if the mini-batch size varies for the sentences with different lengths.

A similar learning technique has also been applied for training the Jordan-type RNN, which corresponds to performing parameter updates after each word. Like in the mini-batch case, we compute all slot values for the sentence in one pass. Then, we keep this history of values as an approximation to the exact values and perform one gradient step for each word. In the experiments, we have observed fast convergence between the exact and approximate slot values.

3.5.3. Dropout regularization

We found that training bi-directional RNNs on the slot/concept predictions of previously trained RNNs gave us poor generalization results due to overfitting. In order to address this issue, we implemented a recently introduced regularization technique called dropouts [10] that omits a given proportion of the hidden nodes for each training sample as well as parts of the input vector. Experimental results show that it allows us to improve the performance of the bi-directional RNN over regular RNNs.

4. Experimental Evaluation

We use the ATIS corpus as used extensively by the SLU community, e.g. [9][17][22]. The training set contains 4978 utterances selected from Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora, while the test set contains 893 utterances from the ATIS-3 Nov93 and Dec94 datasets. In the evaluation, we only use lexical features in the experiments.

4.1. Corpus for learning word embeddings

The methods and data used for learning word embeddings might impact performance on the slot filling task we are interested in. In order to evaluate that impact, different procedures for learning word embeddings have been considered, including SENNA [4] and RNNs [14]. For SENNA, we directly download the embeddings pre-trained on the Wikipedia corpus. RNN word embeddings were obtained by running the RNNLM toolkit available online [14]. We also took into account the dimension *d* of the embedding as a factor of variation. For data, we consider three semantically different corpora for embedding training: Wikipedia, Gutenberg, and Broadcast news. A Wikipedia snapshot is downloaded from [20]. Gutenberg corresponds to digitized books with expired copyrights which we downloaded and built ourselves. For the Broadcast news corpus, we directly downloaded the word embeddings provided by [14] on the RNNLM website.

4.2. Results on word embeddings

To evaluate the impact on SLU of learning methods and data for word embedding training, we test different types of word embeddings trained on different corpora and by different methods. We first consider a frame-level MLP setting (e.g., a feed-forward MLP with inputs of only word embedding features within a word context window), and we compare results of using the embeddings as is versus fine-tuning them during training. Results are also compared with randomly initialized word embeddings of various dimensions $\{50, 80, 100\}$ which are fine-tuned during training. In the experiment, an MLP with 1000 hidden nodes is trained. 1000 sentences of the original ATIS training set are held out as a validation set for choosing the best word context window size and hyper-parameters. The model is then re-trained with the best set of hyper-parameters on the whole ATIS training set.

method	Corpus	embedding's dimension	w/o fine-tuning	w/ fine-tuning
SENNA	Wikipedia	50	92.01	**92.38**
RNNLM	Wikipedia	100	90.51	90.61
	Gutenberg	100	90.20	90.31
	Broadcast	80	90.14	90.58
Random	N/A	50	N/A	90.26
		80		90.94
		100		90.81

Table 2: F1 score on the ATIS task for different methods and training corpora for the embeddings, with the corresponding dimension d of word embeddings.

We first observe from Table 2 that fine-tuning word embeddings is helpful, improving results by a small but consistent margin across the board. We also find that SENNA embeddings gives the best performance. We hypothesize that this may essentially be due to a conditioning issue, since the norm of SENNA word vectors is kept normalized to 1 during training while it is not the case for RNNLM. As indicated by the RNNLM results, word embeddings trained on a semantically different corpus (Wikipedia, Gutenberg, and Broadcast) lead to similar performance, and the differences become even smaller after fine-tuning. In later experiments, we use the embeddings from SENNA.

4.3. Results on Jordan-RNN

There are several choices for feeding the Jordan-type RNN with outputs from previous or future time steps. The first option takes the output probabilities of the NN. Intuitively, probabilities would allow the model to perform a kind of disambiguation since no hard decision is made. The second option considers the hard decision of the model in both the training and testing phases. The third option differs from the last choice during training: the model is trained with ground truth labels, while at test time, since no ground truth labels are available, the hard decisions are used.

For all these options, we measure the precision, recall and F1-score using the conlleval.pl script [22] and compare it to a CRF baseline. The CRF hyper-parameters, i.e., window sizes and regularizers, have been chosen using 5-fold cross validation on the ATIS training set. We use the CRFpp toolkit [5] to run these experiments. Results are reported in Table 3. All the Jordan-type RNNs outperform the CRF baseline. As expected, the probability-based J-RNN outperforms the hard-decision-based version. More interestingly, the forward Jordan-type RNN trained with ground-truth labels obtains the best performance although there is a condition mismatch (e.g., no ground truth label is available in testing). This may be because training with model-predicted labels, either in the hard-decision form or the probability form, could introduce unnecessary noise or convergence difficulty in training.

Model	Precision	Recall	F1-score
CRF baseline	94.08	91.82	92.94
Prob. – (past)	92.93	93.66	93.39
Prob. – (future)	92.93	93.58	93.26
Hard – (past)	92.52	93.76	93.14
Hard – (future)	92.55	93.76	93.15
Ground – (past)	93.42	94.11	93.77
Ground – (future)	92.76	93.87	93.31

Table 3: results on several choices of sequential inputs in the Jordan-type RNN predicting from the past/future i.e., forward/backward: probabilities, hard decisions or ground-truth during training and hard decisions for testing.

4.4. Results on slot filling accuracy

We compare the performance of the introduced RNNs and CRF at the sequential level, along with a frame-level MLP and a Logistic Regression models. Since the NN-based models use word embeddings that leverage unsupervised information from Wikipedia, we clustered all the words in Wikipedia into 200 clusters and add a cluster ID for each word as a discrete feature to the CRF and the Logistic Regression models to make the results comparable. As before, baselines have been trained with CRFpp with 5-fold cross validation for the regularization parameter and the optimal window size.

Experimental results in Table 4 show that models that consider sequential dependency outperform models that don't, and the RNN models consistently outperform the CRF model. We also observe that the Elman-type RNN's forward version (e.g., use past information) performs very well while its backward version (e.g., use future information) gives worse results, though mathematically these two versions are symmetric to each other. Further analysis of the ATIS dataset shows that most of the concept slots to be predicted in ATIS are located in the second half of sentences, which makes the backward model perform predictions with very little historical information. This is also shown by the best hyper-parameters found for the Elman-type RNN which included a window of size 3 for the forward model and 13 for the backward model. The backward model was trying to get the historical information inside the word context window while it was available in the hidden layer for the forward model.

The Jordan-type RNNs, although giving similar results to the forward Elman-type RNN, has shown to be more robust to this problem. Further, the bi-directional version of the Jordan-type RNN improves upon both the backward and forward Jordan models. It is trained with dropout regularization [9] and rectifiers nodes i.e., $f(x) = \max(0, x)$, for 700 epochs and with a batch size of 100. Input nodes are dropped out with a probability $p = 0.2$ while for hidden nodes we used $p = 0.5$. Compared to the CRF+Wiki baseline, it yields an absolute improvement of the F1 score of 0.98%, corresponding to a relative error reduction of 14%.

Models	Prec.	Rec.	F1
Logistic Regress.	91.54	90.73	91.13
Logistic Regress.+Wiki	91.82	91.82	91.82
Frame-NN	92.17	92.59	92.38
CRF	94.08	91.82	92.94
CRF+Wiki	93.77	92.25	93.00
Elman-RNN (past)	93.25	94.04	93.65
Elman-RNN (future)	91.75	92.49	92.12
Jordan-RNN (past)	93.42	94.11	93.77
Jordan-RNN (future)	92.76	93.87	93.31
Bi-dir. Jordan-RNN	93.82	94.15	**93.98**

Table 4: Detailed performance measures (precision, recall, and F1 score) for a set of models evaluated on ATIS.

5. Conclusion and Discussion

We carried out comprehensive investigations of RNNs for the task of slot filling in SLU. We implemented and compared several RNN architectures, including the Elman-type and Jordan-type networks with their variants. We also studied the effectiveness of word embeddings for slot filling. To make the results easy to reproduce and to compare, we implemented all networks on the common Theano neural network toolkit, and evaluated them on the ATIS benchmark. Our results show that both Elman and Jordan-type networks outperform the CRF baseline substantially, both giving similar performance. A bi-directional version of the Jordan-RNN gave the best performance, outperforming the CRF-based baseline by 14% in relative error reduction. Future work will explore more efficient training of RNNs and the choice of more comprehensive features [28] and using a different RNN training toolkit [14] incorporating more advanced features.

6. References

[1] 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," *Proc. Python for Scientific Computing Conference (SciPy) 2010*.

[2] Y. Bengio, R. Ducharme and P. Vincent, "A Neural Probabilistic Language Model", in NIPS 2000

[3] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in ICML 2008.

[4] 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, 2011.

[5] CRPpp: http://crfpp.googlecode.com/svn/trunk/doc/index

[6] 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*," vol. 20, no. 1, pp. 30-42, January 2012.

[7] 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," IEEE Workshop on Spoken Language Technologies, December 2012.

[8] J. Elman, "Finding structure in time," in Cognitive Science, 14 (2), 1990.

[9] Y. He and S. Young, "A data-driven spoken language understanding system," in IEEE ASRU 2003.

[10] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv: 1207.0580v1, 2012.

[11] G. Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath,, and Brian Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," in *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82-97, Nov. 2012.

[12] M. Jordan, "Serial order: A parallel distributed processing approach," in Tech. Rep. No. 8604. San Diego: University of California, Institute for Cognitive Science.

[13] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in ICML 2001.

[14] T. Mikolov, http://www.fit.vutbr.cz/~imikolov/rnnlm/

[15] T. Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network based language model," in ICASSP 2011.

[16] A. Mnih and G. Hinton, "A scalable hierarchical distributed language model" in NIPS, 2008, pp. 1081-1088.

[17] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in Interspeech 2007.

[18] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, November 1997.

[19] H. Schwenk and J-L. Gauvain, "Training neural network language models on very large corpora," in HLT/EMNLP 2005.

[20] C. Shaoul and C. Westbury. 2010. The Westbury lab wikipedia corpus.

[21] Socher, R., Lin, C., Ng, A., and Manning, C. "Learning continuous phrase representations and syntactic parsing with recursive neural networks," Proc. ICML, 2011.

[22] G. Tur, D. Hakkani-Tur, and L. Heck, "What is left to be understood in ATIS?" in IEEE SLT, 2010.

[23] G. Tur and L. Deng, "Intent Determination and Spoken Utterance Classification," in Chapter 4, Tur and De Mori (eds). Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, pp. 81-104, Wiley, 2011

[24] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, "Towards Deeper Understanding Deep Convex Networks for Semantic Utterance Classification," in ICASSP, 2012.

[25] Y. Wang, L. Deng, and A. Acero, "Spoken Language Understanding — An Introduction to the Statistical Framework," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16-31, 2005.

[26] Y. Wang, L. Deng, and A. Acero, "Semantic Frame Based Spoken Language Understanding," in Chapter 3, Tur and De Mori (eds) Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, , pp. 35-80, Wiley, 2011.

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

[28] K. Yao, G. Zweig, M-Y. Hwang, Y. Shi, D. Yu, "Recurrent neural networks for language understanding", submitted to Interspeech 2013.

TOWARDS END-TO-END SPOKEN LANGUAGE UNDERSTANDING

Dmitriy Serdyuk [*1,2] *Yongqiang Wang*[1] *Christian Fuegen*[1] *Anuj Kumar*[1] *Baiyang Liu*[1] *Yoshua Bengio*[2]

[1] Facebook
One Hacker Way
Menlo Park, CA 94025, USA

[2] Université de Montréal, MILA
2920 Chemin de la Tour, office 3353,
Montreal, QC H3T 1J4, Canada

{*serdyuk, yqw, fuegen,anujk,baiyangliu*}*@fb.com*

ABSTRACT

Spoken language understanding system is traditionally designed as a pipeline of a number of components. First, the audio signal is processed by an automatic speech recognizer for transcription or n-best hypotheses. With the recognition results, a natural language understanding system classifies the text to structured data as domain, intent and slots for downstreaming consumers, such as dialog system, hands-free applications. These components are usually developed and optimized independently. In this paper, we present our study on an end-to-end learning system for spoken language understanding. With this unified approach, we can infer the semantic meaning directly from audio features without the intermediate text representation. This study showed that the trained model can achieve reasonable good result and demonstrated that the model can capture the semantic attention directly from the audio features.

Index Terms— Spoken language understanding, end-to-end training, recurrent neural networks

1. INTRODUCTION

With the growing demand of voice interfaces for mobile and virtual reality (VR) devices, spoken language understanding (SLU) has received many researchers' attention recently [1, 2, 3, 4, 5, 6, 7]. Given a spoken utterance, a typical SLU system performs three main tasks: domain classification, intent detection and slot filling [1]. Standard SLU systems are usually designed as a pipeline structure. As shown in Fig. 1, recorded speech signals are converted by an automatic speech recognition (ASR) module into the spoken format text, followed by an optional inverse text normalization module to translate the spoken domain text to the written domain; then a natural language understanding (NLU) module is used to determine intent and extract slots accordingly. Although there are some works (e.g. [8]) that take the possible ASR errors into consideration when designing the NLU module, the pipeline approach is widely adopted. One arguable limitation of this pipelined architecture is that each module is optimized separately under different criteria: the ASR module is trained to minimize the word error rate (WER) criterion, which typically weights each word equally; obviously not every word has the same impact on the intent classification or slot filling accuracy. On the other hand, the NLU module is typically trained on the clean text (transcription) without ASR errors, but during evaluation, it has to use recognized hypotheses outputted by the ASR module as the input: errors in ASR module, especially in noisy conditions, will be propagated to SLU degrading its performance. Second consideration is how human process speech signal to extract intent level concepts. Intuitively, when asked to perform intent detection tasks, humans do not understand speech by recognizing word by word; instead, humans interpret and understand speech directly, while attention is given to the high level concepts which are directly related with the task. It is thus desirable to train an SLU system in an end-to-end fashion.

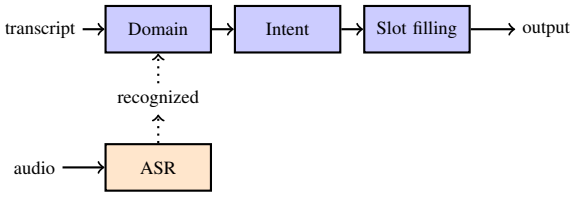


Fig. 1: Traditional SLU system. The NLU part (in blue) is trained on transcripts and consists of a domain classifier, intent classifier and a slot filling model. The speech recognition part (in orange) is trained independently. During the evaluation time the NLU system uses the recognized text from the ASR system, as denoted by the dotted arrows.

End-to-end learning has been widely used in several areas, such as machine translation [9, 10, 11] and image-captioning[12]. It has also been investigated for speech synthesis [13] and ASR tasks [14, 15, 16]. For example, the CTC loss function is used to train an ASR system to map the feature sequences directly to the word sequences in [16] and it has been shown to perform similarly to the traditional ASR systems; in [17, 18], encode-decoder models with attention have been used for ASR tasks, including large vocabulary ASR. End-of-end learning of memory networks is also used

*This work was done when the first author was an intern with Facebook.

for knowledge carryover in multi-turn spoken language understanding [19].

Inspired by these success, we explore the possibility to extend the end-to-end ASR learning to include NLU component and optimize the whole system for SLU purpose. As the first step towards an end-to-end SLU system, in this work, we focus on maximizing the single-turn intent classification accuracy using log-Mel filterbank feature directly. Contributions of this work are following: 1) we demonstrate the possibility of training a language understanding model from audio features (see Section 3). To the best knowledge of authors this is the first work on this topic; 2) We show that the performance of an SLU degrades when evaluating on the ASR output. The details of our experiments are described in Section 4, and results in Section 5; Additionally, we perform experiments on noise-corrupted data. We artificially add noise to the audio data to demonstrate the degradation of the performance of a standard SLU evaluation and test our system.

2. STANDARD ASR AND NLU SYSTEMS

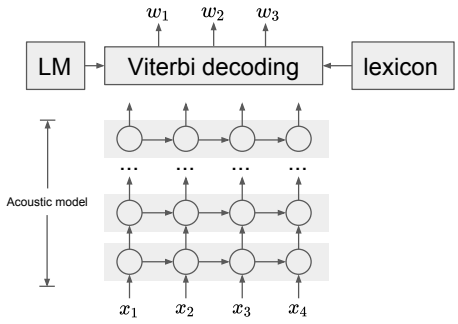


Fig. 2: A schematic picture of a traditional ASR system. The recurrent acoustic model predicts the states of an HMM. It is decoded with Viterbi algorithm using the language model (LM).

Given a sequence of feature vectors $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, an ASR system is trained to find the most likely word sequences $\boldsymbol{W}^* = (w_1, \ldots, w_n)$ using the chain rule:

$$\boldsymbol{W}^* = \arg\max_{\boldsymbol{W}} p(\boldsymbol{W}|\boldsymbol{X}) = \arg\max_{\boldsymbol{W}} p(\boldsymbol{X}|\boldsymbol{W})p(\boldsymbol{W}).$$

Therefore, the ASR system is usually divided into two models: an acoustic model $p(\boldsymbol{X}|\boldsymbol{W})$ and a language model $p(\boldsymbol{W})$ (AM and LM respectively). CD-HMM-LSTM [20] is widely used as an AM, in which the feature vector sequence is converted to the likelihood vectors of context-dependent HMM states for each acoustic frame. Together with the LM ($p(\boldsymbol{W})$ is usually a statistical n-gram model) and a dictionary, a Viterbi decoder is used to search for the most likely word sequence. Fig. 2 depicts the described standard ASR architecture: the core part of the AM is a multi-layer LSTM [21] network, which predicts the probability of CD-HMM states for each frame. Since most of the SLU systems requires spontaneous response, usually only uni-directional LSTM is used.

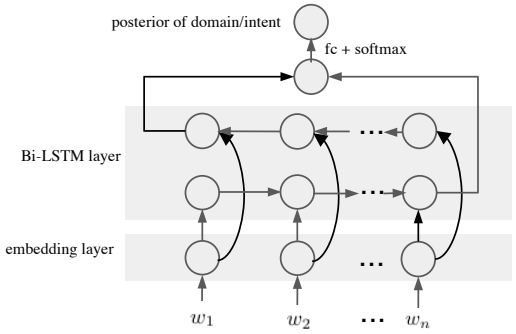


Fig. 3: A schematic diagram of a standard NLU system for domain/intent classification. "`fc`" means a fully-connected layer.

Given the word sequence output by the ASR module, an NLU module is used to perform domain and intent classification, and to fill slots for different intents accordingly. Following [5], we use LSTM-based utterance classifier, in which the input words are first embedded in a dense representation, and then the LSTM network is used to encode the word sequence. We found 2-layer bi-directional LSTM encoder performs better. Fig. 3 demonstrates how a standard NLU system predicts the domain/intent for a given word sequence. Note that since the NLU system incurs much smaller latency compared with the ASR system and the classification only starts after the entire word sequence become available, it is possible to use bi-directional RNNs for NLU.

In the pipelined approach to SLU, ASR, and NLU modules are usually trained independently, where the NLU module is trained using human transcription as the input. During the evaluation phase, the ASR output is piped into the NLU module.

3. END-TO-END SPOKEN LANGUAGE UNDERSTANDING

As the first step towards the end-to-end spoken language understanding, we focus on two tasks: speech-to-domain and speech-to-intent. Both tasks are sequence classification problems where the input is log-Mel filterbank feature vectors.

The task of end-to-end SLU is close to speech recognition with a difference that the structure of the output is simpler but the transformation is more complicated and the data is noisy and non-unimodal (close utterances may have sufficiently different intents). For this reason our model was inspired by works in end-to-end speech recognition [22, 10, 15, 23]. We use an encoder-decoder framework. The input log-Mel filterbank features are processed by an encoder which is a multi-layer bidirectional *gated recurrent unit* (GRU, [24]) network. A potential issue of using log-Mel filterbank feature is that it is generated every 10 ms: the 10-ms frame rate is good

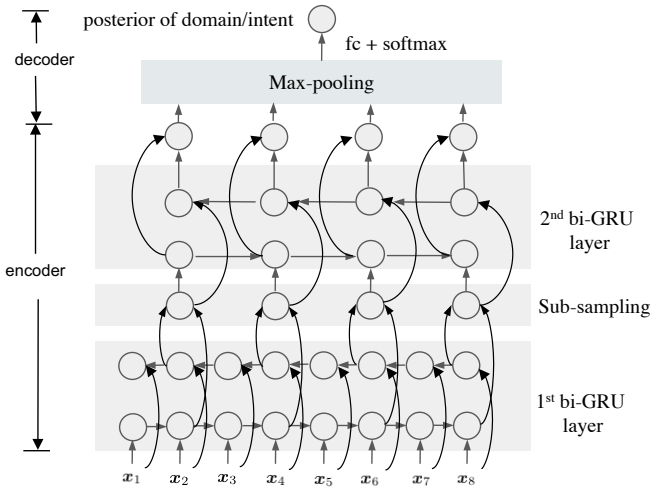


Fig. 4: End-to-end SLU system. Sub-sampling is performed at every layer to reduce the output sequence length. Our best model uses 4 layer bidirectional GRU.

for classifying sub-phone unit like CD-HMM states, while it may not be suitable for utterance-level classification as GRUs may forget most of the speech content when it arrives at the end of the utterance end due to gradient vanishing. In order to reduce the sequence length processed by GRUs, we sub-sample [10, 15] the hidden activations along the time domain for every bi-direction GRU layer (see Fig. 4). This allowed us to extract a representation roughly at a syllable level in a given utterance. On the other hand, this significantly reduced the computational time for both training and prediction, which enables us to use bi-directional GRU for real time intent and/or domain classification. Given the encoder output, a max-pooling layer along the time axis is used to compress it into a fixed dimension vector. This is followed by a fully-connected feed-forward layer. Finally, a softmax layer is used to compute the posterior probability of intents or domains.

4. EXPERIMENTS

We train and evaluate our models on an in-house dataset containing VR spoken commands collected for that purpose. The dataset is close in spirit to ATIS corpus [25]. The dataset contains about 320 hours of near field annotated data collected from a diverse set of more than 1000 de-identified speakers. It was recorded in two scenarios: scripted and free speech. The scripted half is read speech with a fixed script. For the free speech part, the participants were asked to achieve certain goal using any phrasing, then these utterances were transcribed. Every utterance has transcription as well as meta information including a domain label and an intent label. After cleaning data we extracted 5 domains and 35 distinct intents. Roughly 11,000 utterances, totaling 10 hours audio, are used as the evaluation set.

For all our experiments we used log-Mel filterbank features. We used an encoder-decoder architecture [9]. The encoder is a 4 layer bidirectional GRU network with 256 units every layer. The output was sub-sampled with a stride of 2 at every layer to reduce the length of the representation. The decoder network takes the output of the encoder and aggregates them with a max-pooling layer and puts it through a 1-layer feed-forward network (hidden size 1024) to produce either the domain or intent class. The network was optimized with Adam [26] algorithm until convergence on the validation set. We used the *batch normalization* [27] for every feed-forward connection to speed up the convergence of this network.

The baseline NLU model for our experiments was a recurrent network similar to our model with an exception that we did not use sub-sampling. The input word was represented as one-hot vector and a trainable embedding was used. This model is close to state of the art models and ones used in production [28, 29]. This network was evaluated in two regimes: using the transcript text, and using the recognized text. The former shows the upper bound for our models and corresponds to the perfect speech recognition. The later regime was transcribed by a speech recognizer: the core part of AM is a four-layer CD-HMM-LSTM network[20] with 800 memory cells in each layer, trained on the same 320 hours training data with a cross-entropy criterion to predict posterior probabilities of 6,133 clustered content-dependent states; the vocabulary size is 80,000 and the LM is trained on the transcribed 320 hours speech and interpolated with a large background LM which is trained on other sources; there are roughly 200M n-grams in the LM. Our ASR system achieved 3.5% word error rate on the evaluation set. Each utterance from the evaluation set was recognized with this ASR and inputed to the baseline NLU network. In order to provide the fair comparison we did not use any external data in both cases. No pretrained embeddings or dictionary look-up was used.

We also emulate the real-world situation where the input to the SLU system is noisy. Both training and evaluation datasets were corrupted by convolving with recorded room impulse responses (RIRs) whose T60 times ranges from 200ms to 1 second. Background noise was added as well: for training data, the SNR ranges from 5 to 25dB, while for evaluation data, the SNR ranges from 0 to 20dB. Every training utterance is distorted 2 times by using different RIRs, sources of background noise and SNRs. This results in a 600 hours noise-corrupted training set. Utterances in evaluation set are only distorted once. Both the ASR system and our end-to-end system were retrained on this noise-corrupted training set. Due to the reverberation and relatively strong background noise, the ASR has considerably higher word error rate (28.6% WER).

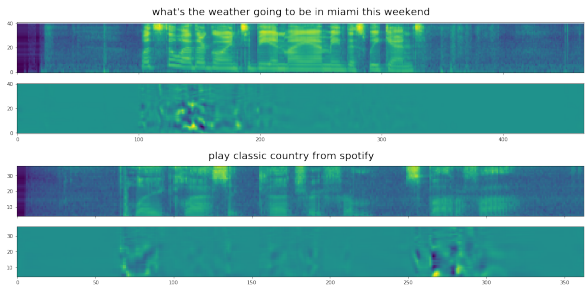


Fig. 5: Filter-bank features and corresponding saliency maps for our speech-to-domain model. It can be seen that the model responds to "what's the weather" part and "play" or "spotify" from the second example. Top image is "weather" domain, bottom image is "music" domain.

5. RESULTS AND DISCUSSION

We first present results of domain classification. Five domains in this corpus are "music," "weather," "news," "sports," "opt-in, opt-out." The results for domain recognition are summarized in Table 1. The performance in "Transcript text" row is evaluated on perfect transcriptions that correspond to human transcription and such performance is not achievable with the current ASR system. For qualitative analysis of our end-to-end model, we visualize the saliency map (gradients w.r.t. input) in Fig. 5 for a selected utterance. We notice that the position of the network response corresponds to meaningful positions in the utterance. As it can be seen in Table 1, the accuracy is close to perfect on this dataset. Therefore we continue with a more challenging task of the intent classification.

Table 1: Results for domain classification. The first row corresponds to evaluation on clean transcripts, the maximum performance achievable with this model.

Input	**Accuracy, %**
Transcript text	99.2
Recognized text	98.1
Audio	97.2

Examples of intents are "Learn about the weather in your location," "Learn about the weather in different location," "Learn the results of a completed sports match, game, tournament," "Learn the status of an ongoing sports match, game, tournament." These classes are more fine grained and require deeper understanding of a given utterance. For the ablation study we report the performance of variations of our end-to-end model: initially, we use last-layer hidden activations at the end of both left-to-right and right-to-left directions as the encoder output; this is compared with using max pooling to aggregate all the hidden activations in the last layers in Table 2, which shows that using max-pooling activations increases performance. We hypothesize that due to the long input speech sequence (ranging from 100 to 1,000 input frames), the activations in the last time step may not sufficiently summarize the utterance. This is contrary to the NLU model using text as input, whereas using hidden activations from the last time step usually suffice, as the input length is relatively small (a majority of them are less than 20 words). We also observe that using batch normalization (BN) in both encoder and decoder significantly improves the classification accuracy, which indicates that our end-to-end model is difficult to optimize due to the long input sequence. The results for the noise-corrupted data are also reported in the bottom part of Table 2. The performance of both models degrades significantly. A potential advantage of our end-to-end approach to SLU is that since the model is optimized under the same criterion, the model can be made very compact; this is demonstrated in the same table by comparing the number of parameters of neural networks in standard SLU system and our system. Note that the parameters in the LM are not included, although it is usually two or three magnitude more than AM in ASR. Due to the compactness of our end-to-end model, we are able to predict the intent or domain with a real time factor around 0.002, which makes the use of bi-GRU in our model possible for real time applications.

Table 2: Results for intent classification. As mentioned above, the first row is the maximum performance achievable with this model. We report the number of parameters for the models used to demonstrate that a much smaller end-to-end model achieves reasonable performance.

Input	**Parameters**	**Accuracy, %**
Text input		
Transcript	15.5M	84.0
Recognized	15.5M	80.9
Clean audio features		
no BN, no max-pool	0.4M	71.3
no BN, max-pool	0.4M	72.5
BN, max-pool	0.4M	74.1
Noisy audio features		
Recognized text	15.4M	72.0
Audio	0.4M	64.9

With this work we hope to start a discussion on the topic of the audio-based SLU. Although, the end-to-end approach does not show superior performance, it provides a promising research direction. With significantly less parameters our system is able to reach 10% relatively worse accuracy. One of the future directions for this work is to encompass the slot filling task into this framework. It requires simultaneous prediction of a word and a slot. This can be tackled with the attention-based networks. Other directions are to explore different architectures for the decoder, such as using different pooling strategies, using deeper networks and incorporating convolutional transformations.

6. REFERENCES

[1] Gokhan Tur and Renato De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*, John Wiley & Sons, 2011.

[2] Puyang Xu and Ruhi Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *ICASSP*, 2014.

[3] Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, "Recurrent neural networks for language understanding," in *Interspeech*, 2014.

[4] Aditya Bhargava, Asli Celikyilmaz, Dilek Hakkani-Tür, and Ruhi Sarikaya, "Easy contextual intent prediction and slot detection," in *ICASSP*, 2013.

[5] Suman V Ravuri and Andreas Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification," in *Interspeech*, 2015.

[6] Ruhi Sarikaya, Geoffrey E Hinton, and Anoop Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 2014.

[7] Gokhan Tur, Li Deng, Dilek Hakkani-Tür, and Xiaodong He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *ICASSP*, 2012.

[8] Fabrizio Morbini, Kartik Audhkhasi, Ron Artstein, Maarten Van Segbroeck, Kenji Sagae, Panayiotis Georgiou, David R Traum, and Shri Narayanan, "A reranking approach for recognition and classification of speech input in conversational dialogue systems," in *SLT*, 2012.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv:1409.0473*, 2014.

[11] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin, "Convolutional sequence to sequence learning," *arXiv:1705.03122*, 2017.

[12] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.

[13] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[14] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *ICML*, 2016.

[15] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[16] Hagen Soltau, Hank Liao, and Hasim Sak, "Neural speech recognizer: acoustic-to-word LSTM model for large vocabulary speech recognition," *arXiv:1610.09975*, 2016.

[17] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[18] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*, 2016.

[19] Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding," in *Interspeech*, 2016.

[20] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, 2014.

[21] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[22] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[23] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," *arXiv preprint arXiv:1707.07413*, 2017.

[24] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014.

[25] Charles T Hemphill, John J Godfrey, George R Doddington, et al., "The ATIS spoken language systems pilot corpus," in *DARPA speech and natural language workshop*, 1990.

[26] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[27] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[28] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa, "Natural language processing (almost) from scratch," *JMLR*, 2011.

[29] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao, "Recurrent convolutional neural networks for text classification," in *AAAI*, 2015.

WHAT IS LEFT TO BE UNDERSTOOD IN ATIS?

Gokhan Tur Dilek Hakkani-Tür Larry Heck

Speech at Microsoft | Microsoft Research
Mountain View, CA, 94041
gokhan.tur@ieee.org dilek@ieee.org larry.heck@microsoft.com

ABSTRACT

One of the main data resources used in many studies over the past two decades for spoken language understanding (SLU) research in spoken dialog systems is the airline travel information system (ATIS) corpus. Two primary tasks in SLU are intent determination (ID) and slot filling (SF). Recent studies reported error rates below 5% for both of these tasks employing discriminative machine learning techniques with the ATIS test set. While these low error rates may suggest that this task is close to being solved, further analysis reveals the continued utility of ATIS as a research corpus. In this paper, our goal is not experimenting with domain specific techniques or features which can help with the remaining SLU errors, but instead exploring methods to realize this utility via extensive error analysis. We conclude that even with such low error rates, ATIS test set still includes many unseen example categories and sequences, hence requires more data. Better yet, new annotated larger data sets from more complex tasks with realistic utterances can avoid over-tuning in terms of modeling and feature design. We believe that advancements in SLU can be achieved by having more naturally spoken data sets and employing more linguistically motivated features while preserving robustness due to speech recognition noise and variance due to natural language.

Index Terms— spoken language understanding, ATIS, discriminative training

1. INTRODUCTION

Spoken language understanding (SLU) aims to extract the *meaning* of the speech utterances. While understanding language is still considered an unsolved problem, in the last decade, a variety of practical goal-oriented conversational understanding systems have been built for limited domains. These systems aim to automatically identify the intent of the user as expressed in natural language, extract associated arguments or slots, and take actions accordingly to satisfy the user's requests. In such systems, the speaker's utterance is typically recognized using an automatic speech recognizer (ASR). Then the intent of the speaker is identified from the recognized word sequence using an SLU component. Finally, a dialog or task manager (DM) interacts with the user (not necessarily in natural language) and helps the user achieve the task that the system is designed to support.

In the early 90s, DARPA (Defense Advanced Research Program Agency) initiated the Airline Travel Information System (ATIS) project. The ATIS task consisted of spoken queries on flight-related information. An example utterance is *I want to fly to Boston from New York next week*. Understanding was reduced to the problem of extracting task-specific arguments, such as *Destination* and *Departure Date*. Participating systems employed either a data-driven statistical approach [1, 2] or a knowledge-based approach [3, 4, 5].

Almost simultaneously with the semantic frame filling-based SLU approaches, a new task emerged motivated by the success of the early commercial interactive voice response (IVR) applications used in call centers. The SLU was framed as *classifying users' utterances into predefined categories* (called as *intents* or *call-types*) [6].

The biggest difference between the call classification systems and semantic frame filling systems is that the former does not explicitly seek to determine the arguments provided by the user. The main goal is *routing* the call to an appropriate call center department. The arguments provided by the user are important only in the sense that they help make the right classification. While this has been a totally different perspective for the task of SLU, it was actually complementary to template filling in that each call-type can be viewed as a template to be filled. For example, in the case of the DARPA ATIS project, while the primary *intent* (or goal) was *Flight*, users also asked about many other things such as *Ground transportation* or *Airplane specifications*. The program also defined specialized templates for these less frequent intents. This led to a seamless integration of intent determination (ID) and slot filling (SF) based SLU approaches. This integrated approach actually yielded *improved* end-to-end automation rates as compared to the previous decoupled and sequential approaches. For example, Jeong *et al* [7] proposed to model these two systems *jointly* using a triangular chain conditional random field (CRF).

In this paper, rather than focus on specific techniques or features to improve ID and SF accuracy, our goal is to assess the continued utility of the ATIS corpus given the two decades of research it has supported. In the next section, we briefly describe the ATIS corpus and then discuss the evaluation metrics for ID and SF. In Section 4, we present the state-of-the-art discriminative training efforts for both ID and SF for the task of ATIS. Finally, in Sections 5 and 6 we present our detailed analyses on the errors we have seen using ID and SF models, respectively, with comparable performance to those reported in the literature. We will show that, by categorizing the erroneous cases that remain after N-fold cross validation experiments, ATIS is still useful and suggests future research directions in SLU.

2. AIRLINE TRAVEL INFORMATION (ATIS) CORPUS

An important by-product of the DARPA ATIS project was the ATIS corpus. This corpus is the most commonly used data set for SLU research [8]. The corpus has seventeen different intents, such as *Flight* or *Aircraft capacity*. The prior distribution is, however, heavily skewed, and the most frequent intent, *Flight* represents about 70% of the traffic. Table 1 shows the frequency of the intents in this corpus for training and test sets.

In this paper, we use the ATIS corpus as used in He and Young [9] and Raymond and Riccardi [10]. The training set contains 4,978 ut-

Intent	Training Set	Test Set
Abbreviation	2.4%	3.6%
Aircraft	1.6%	0.9%
Airfare	9.0%	5.8%
Airline	3.4%	4.3%
Airport	0.5%	2.0%
Capacity	0.4%	2.4%
City	0.3%	0.6%
Day_Name	0.1%	0.1%
Distance	0.4%	1.1%
Flight	73.1%	71.6%
Flight_No	0.3%	1.0%
Flight_Time	1.2%	0.1%
Ground_Fare	0.4%	0.8%
Ground_Service	5.5%	4.0%
Meal	0.1%	0.6%
Quantity	1.1%	0.9%
Restriction	0.3%	0.1%

Table 1. The frequency of intents for the training and test sets.

Utterance	*How much is the cheapest flight from Boston to New York tomorrow morning?*
Goal:	Airfare
Cost_Relative	*cheapest*
Depart_City	*Boston*
Arrival_City	*New York*
Depart_Date.Relative	*tomorrow*
Depart_Time.Period	*morning*

Table 2. An example utterance from the ATIS dataset.

terances selected from the Class A (context independent) training data in the ATIS-2 and ATIS-3 corpora, while the test set contains 893 utterances from the ATIS-3 Nov93 and Dec94 datasets. Each utterance has its named entities marked via table lookup, including domain specific entities such as city, airline, airport names, and dates.

The ATIS utterances are represented using semantic frames, where each sentence has a goal or goals (a.k.a. intent) and slots filled with phrases. The values of the slots are not normalized or interpreted. An example utterance with annotations is shown in Table 2.

3. EVALUATION METRICS

The most commonly used metrics for ID and SF are class (or slot) error rate (ER) and F-Measure. The simpler metric ER for ID can be computed as:

$$ER_{ID} = \frac{\#\text{ misclassified utterances}}{\#\text{ utterances}}$$

Note that one utterance can have more than one intent. A typical example is *Can you tell me my balance? I need to make a transfer*. In most cases, where the second intent is generic (a greeting, small talk with the human agent) or vague, it is ignored. If none of the true classes is selected, it is counted as a misclassification.

For SF, the error rate can be computed in two ways: The more common metric is the F-measure using the slots as units. This metric is similar to what is being used for other sequence classification tasks in the natural language processing community, such as parsing and named entity extraction. In this technique, usually the IOB schema is adopted, where each of the words are tagged with their position in the slot: beginning (B), in (I) or other (O). Then, recall and precision values are computed for each of the slots. A slot is considered to be correct if its range and type are correct. The F-Measure is defined as the harmonic mean of recall and precision:

$$F - Measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

where

$$Recall = \frac{\#\text{ correct slots found}}{\#\text{ true slots}}$$

$$Precision = \frac{\#\text{ correct slots found}}{\#\text{ found slots}}$$

.

4. BACKGROUND ON USING DISCRIMINATIVE CLASSIFIERS FOR SLU

With advances in machine learning over the last decade, especially in discriminative classification techniques, researchers have framed the ID problem as a sample classification task and SF as a sequence classification task. Typically, word n-grams are used as features after preprocessing with generic entities, such as dates, locations, or phone numbers. Because of the very large dimension of the input space, large margin classifiers such as SVMs [11] or Adaboost[12] were found to be very good candidates for ID and CRFs [13] for SF. To take into account context, the recent trend is to match n-*grams* (a substring of n words) rather than words.

As discovered, data driven approaches are very well-suited for processing spontaneous spoken utterances. The data driven approaches are typically more robust to sentences that are not well-formed grammatically, which occurs frequently in spontaneous speech. Even in broadcast conversations where participants are very well trained and prepared, a large percentage of the utterances have disfluencies: repetitions, false starts, and filler words (e.g., *uh*) [14]. Furthermore, speech recognition introduces significant "noise" to the SLU component caused by background noise, mismatched domains, incorrect recognition of proper names (such as city or person names), and reduced accuracy due to sub-real time processing requirements. A typical call routing system operates at around 20%-30% word error rate; one out of every three to five words is wrong [15]. Given that the researchers in this study also determined that one third of the ID errors are due to speech recognition noise, robust methods for spontaneous speech recognition are critically important for successful ID and SF in SLU systems. To this end, researchers have proposed many methods ranging from N-best rescoring, exploiting word confusion networks, and leveraging dialog context as prior knowledge (e.g., [15]).

4.1. Intent Determination

For ID, early work with discriminative classification algorithms was completed on the AT&T HMIHY system [6] using the Boostexter tool, an implementation of the AdaBoost.MH multiclass multilabel classification algorithm [12]. Hakkani-Tür *et al.* extended this work by using a lattice of syntactic and semantic features [16]. Discriminative call classification systems employing large margin classifiers (e.g., support vector machines) include work by Haffner *et al.* [17], who proposed a global optimization process based on an optimal

Correct-Estimated	a	b	b	d	e	f	g	h	i	j	k	l	m	n	o	p	q
a. *Abbreviation*	30									2							
b. *Aircraft*		6								3							
c. *Airfare*			64							1							
d. *Airline*				37						2							
e. *Airport*					15					2					1		
f. *Capacity*	1	5				13				2							
g. *City*							3			2							
h. *Day_Name*										2							
i. *Distance*									9	1							
j. *Flight*		1	1				1			623							
k. *Flight_No*										2	6						
l. *Flight_Time*										1							
m. *Ground_Fare*			1										3	3			
n. *Ground_Service*														36			
o. *Meal*										5							
p. *Quantity*																8	
q. *Restriction*	1																

Table 3. The confusion matrix for intent determination.

channel communication model that allowed a combination of *heterogeneous* binary classifiers. This approach decreased the call-type classification error rate for AT&T's HMIHY natural dialog system significantly, especially the false rejection rates.

Other work by Kuo and Lee [18] at Bell Labs proposed the use of discriminative training on the routing matrix, significantly improving their vector-based call routing system [19] for low rejection rates. Their approach is based on using the minimum classification error (MCE) criterion. Later they extended this approach to include Boosting and automatic relevance feedback (ARF) [20]. Cox [21] proposed the use of generalized probabilistic descent (GPD), corrective training (CT), and linear discriminant analysis (LDA).

Finally, Chelba *et al.* proposed using Maximum Entropy models for ID, and compared the performance with a Naive Bayes approach with the ATIS corpus. The discriminative method resulted in half the classification error rate compared to Naive Bayes on this highly skewed data set. They have reported about 4.8% top class error rate using a slightly different training and test corpora than the one used in this paper.

4.2. Slot Filling

For SF, the ATIS corpus has been extensively studied from the early days of the DARPA ATIS project. However, the use of discriminative classification algorithms is more recent. Some notable studies include the following:

Wang and Acero [22] compared the use of CRF, perceptron, large margin, and MCE using stochastic gradient descent (SGD) for SF in the ATIS domain. They obtained significantly reduced slot error rates, with best performance achieved by CRF (though it was the slowest to train).

Almost simultaneously Jeong and Lee [7] proposed the use of CRF, extended by non-local features, which are important to disambiguate the type of the slot. For example, a day can be the arrival day, departure day, or the return day. If the contextual cues disambiguating them are beyond the immediate context, it is not easy for the classifier to choose the correct class. Using non-local trigger features automatically extracted from the training data is shown to improve the performance significantly.

Finally, Raymond and Riccardi [10] compared SVM and CRF with generative models for the ATIS task. They concluded that discriminative methods perform significantly better, and furthermore, it is possible to incorporate a-priori information or long distance features easily. For example they added features such as "Does this utterance have the verb *arrive*". This resulted in about 10% relative reduction in slot error rate. The design of such features usually requires domain knowledge.

5. ANALYSIS OF INTENT DETERMINATION IN ATIS

In this section, our goal is to analyze the errors of a state-of-the-art ID system for the ATIS domain, cluster the errors, and then categorize the error types. These categories of error types will suggest potential areas of research that could yield improved accuracy. All experiments and analyses are performed using manual transcriptions of the training and test sets to isolate the study from noise introduced by the speech recognizer.

5.1. Discriminative Training and Experiments

For the following experiments, we used the ATIS corpus as described previously in Section 2. Since the superior performance of the discriminative training algorithms has been shown by the earlier work, we have employed the AdaBoost.MH algorithm in this study. We used only word n-grams as features. We have not optimized Boosting parameters on a tuning set nor learned weak classifiers. The data is normalized to lower case, but no stemming or stopword removal has been performed.

The ATIS test set was classified according to the classes defined in Table 1. The ID error rate we obtained was 4.5%, which is comparable to (and actually lower than) to what has been reported in the literature.

5.2. Analysis of Intent Determination Errors

Next, we checked the ID errors with three training and test set-ups:

1. *All Train*: uses all ATIS training data to train the model, and errors are computed on the ATIS test set. In total, this model erroneously classified only 40 utterances (an error rate of 4.5%). The intent confusion matrix for these errors is provided in Table 3.

2. *25% Train*: uses 25% of the training examples in the ATIS training set, and errors are computed on the ATIS test set. In total, this model erroneously classified 65 utterances (an error rate of 7.3%).

3. *N-fold*: uses all examples for both testing and training in 10-fold cross validation experiments. In total, this model erroneously classified 162 utterances (an error rate of 3.0%).

As seen in Table 3, the problem is mostly the non-*Flight* utterances erroneously classified as *Flight*. While one cause of these errors is the unbalanced intent distribution, we have manually checked each error and clustered them into 6 categories:

1. *Prepositional phrases embedded in noun phrases*: These errors involve phrases such as *Capacity of the flight from Boston to Orlando*, where the prepositional phrase suggests flight information, whereas the destination category is mainly determined by the head word of the noun phrase (*capacity* in this case). Since classifier has no syntactic features, such sentences are usually classified erroneously. Using features from a syntactic parser can alleviate this problem.

2. *Wrong functional arguments of utterances*: This category is similar to the first category but the difference is that, instead of a prepositional phrase, the confused phrase is a semantic argument of the utterance. Consider the example utterance *What day of the week does the flight from Boston to Orlando fly?* These are errors that can be solved by using either a syntactic parser that identifies functions of phrases or a semantic role labeler.

3. *Annotation errors*: These are utterances that were assigned the wrong category during manual annotation.

4. *Utterances with multiple sentences*: These are utterances with more than one sentence. In such cases, the intent is usually in the last sentence, whereas the classification output is biased by the other sentence.

5. *Other*: These include several infrequent error types such as ambiguous utterances, ill-formulated queries, and preprocessing/tokenization issues:

   - *Ambiguous utterances*: These errors involve utterances where the destination category is not clear in the utterance. An example from the ATIS test set is *list Los Angeles*. In this utterance, the speaker intent could either be to find cities that have flights *from* Los Angeles or flights *to* Los Angeles.

   - *Ill-formulated queries*: These are utterances which include a phrase that may mislead the classification or understanding. An example from the ATIS test set is: *What's the airfare for a taxi to the Denver airport?* In this case, the word *airfare* implies a destination category of *Airfare*, whereas what is meant is *Ground transportation fare*. These type of errors are easier for humans to handle, but it is not presently clear how they can be resolved in automatic processing.

   - *Preprocessing/Tokenization issues*: These are errors that could be resolved by using a domain ontology or special pre-processing or tokenization related to the domain. Some domain specific abbreviations and restriction codes are examples of this category.

Error Type	All Train	25% Train	10-Fold
1	42.5%	33.8%	24.5%
2	22.5%	13.8%	30.0%
3	2.5%	6.1%	18.4%
4	0%	0%	8.0%
5	17.5%	12.5%	7.2%
6	15.0%	33.8%	11.7%

Table 4. The distribution of error categories for ID using all and 25% of the training data, and using all the training and the test set with 10-fold cross validation.

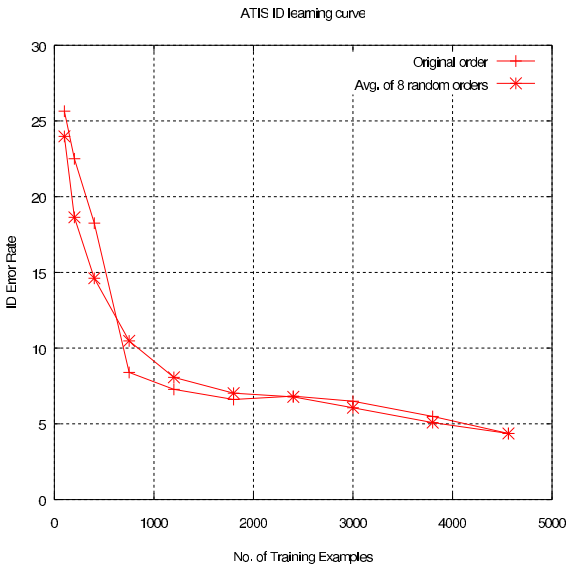


Fig. 1. Learning curve for intent determination using the training data with the original order and average of 8 shuffled orders.

6. *Difficult Cases*: These are utterances that include words or phrases that were previously unseen in the training data. For the example utterance *Are snack served on Tower Air?*, none of the content words and phrases appear with the *Meal* category in the training data.

Table 4 presents the frequency of each of these errors for the three experiments. As seen, categories 1 and 2 constitute a majority of the errors. Both of these categories can be resolved using a syntactic parser with function tags. However, note that the ATIS corpus is highly artificial and utterances are mostly grammatical and without disfluencies. Furthermore, when working with ASR, utterances may include recognition errors. In a more realistic scenario, one might consider shallow parsing or syntactic and semantic graphs [16] for extracting richer and linguistically-motivated features that could resolve such cases.

Figure 1 shows the error rate on the ATIS test set when varying training set sizes are used. When manually examining the test set, we found clusters of similar utterances occurring one after the other (probably are uttered by the same user). To eliminate the bias from the data collection order, we also estimated the error with a random ordering of the training set, and averaged the error rates over 8 such experiments. As can be seen from this plot, the error rate keeps shrinking as more data is added, suggesting that more training data would be beneficial.

6. ANALYSIS OF SLOT FILLING IN ATIS

In this section, our goal is similar to the ID analysis: analyze the results of a state-of-the-art SF system for the ATIS domain and cluster the errors into categories.

6.1. Discriminative Training and Experiments

Following methods described in the literature, we employed linear chain CRFs to model the slots in the ATIS Domain. We used only word n-gram features and did not use a development set to tune parameters. The ATIS test set was then classified using the trained model. We converted the data sets into the IOB format so that we have only one word per sample to classify. Using the CoNLL evaluation script[1], the SF F-Measure we obtained was 93.2% with the IOB representation[2], which is comparable to what has been reported in the literature.

6.2. Analysis of Slot Filling Errors

Analyzing the SF decisions, the model found 2,614 of 2,837 slots with the correct type and span for the input out of 9,164 words. We manually checked each of the 223 erroneous cases and clustered them into 8 categories:

1. *Long distance dependencies*: These are slots where the disambiguating tokens are out of the current n-gram context. For example, in the utterance *Find flights to New York arriving in no later than next Saturday*, a 6-gram context is required to resolve that *Saturday* is the arrival date. This category was previously addressed in the literature. For example, Raymond and Riccardi [10] extracted features using manually-designed patterns and Jeong and Lee [7] used trigger patterns to cover these cases.
2. *Partially correct slot value annotations*: These are slots assigned a category that is partially correct; either the category or the sub-category matches the manual annotation. For example, the word *tomorrow* can either be a *Depart_Date.Relative* or *Arrive_Date.Relative* for the utterance *flights arriving in Boston tomorrow*. Note that these can overlap with other error types.
3. *Previously unseen sequences:* While this category requires further analysis, the most common reason is the mismatch between the training and test sets. For example, *meal* related slots are missed by the model (8.0% of all errors) because there are no similar cases in the training set. This is also the case for the aircraft models (10.0%), and traveling to states instead of cities (3.3%), etc.
4. *Annotation errors*: These are the slots that were assigned the wrong category during manual annotation.
5. *Other*: These include several infrequent error types such as ambiguous utterances, ill-formulated queries, and preprocessing/tokenization issues:
   - *Ill-formulated queries*: These errors usually involve an ungrammatical phrase that may mislead the interpretation of the slot value or there is insufficient context to disambiguate the value of the slot. For example, in the utterance *Find a flight from Memphis to Tacoma dinner*, it is not clear if the word *dinner* refers to the description of the flight meal.
   - *Ambiguous utterances*: These are utterances where the slot category is not explicit given the utterance. For example, in the utterance *I would like to have the airline that flies Toronto, Detroit and Orlando*, it is not clear if the speaker is searching for airlines that have flights from Toronto to Detroit and Orlando or from some other location to Toronto, Detroit and Orlando.
   - *Preprocessing/Tokenization issues*: These are errors that could be resolved using a domain ontology or special pre-processing or tokenization related to the domain. For example, in the utterance *What airline is AS*, it would be helpful to know *AS* is a domain specific abbreviation.
   - *Ambiguous part-of-speech tag-related errors:* These are errors that could be resolved if the part-of-speech tags were resolved. For example, the word *arriving* can be a verb or an adjective, as in the utterance *I want to find the earliest arriving flight to Boston*. In this case, the slot category for the words *earliest arriving* is *Flight-Mod*, but since the word *arriving* is very frequently seen as a verb in this corpus, it is assigned no slot category.

[1] http://www.cnts.ua.ac.be/conll2000/chunking/output.html

[2] It is 94.7% using the representation used by [10], who reported 95.0%

Error Type	Percentage
1	26.9%
2	42.4%
3	57.6%
4	8.4%
5	6.7%

Table 5. The distribution of the types of errors in the ATIS test set. Note that these do not sum to 100% as some errors include multiple types.

Table 5 lists the frequency of each of these errors. Categories 1, 2, and 3 constitute vast majority of the errors. Each of these categories can be attacked using a different strategy. Category 1 utterances are the easiest to resolve using richer feature sets during discriminative training. Using a-priori information may also help when available. Also, discovering linguistically motivated long distance patterns is a promising research work. Category 2 utterances happen mainly due to the nuance between the *arrive* and *depart* concepts (23.1% of all errors), which are very hard to distinguish in some cases as in the example above. Category 3 utterances simply require a better training set or human intervention of manual patterns as they are underrepresented or missing in the training data.

7. DISCUSSION AND CONCLUSIONS

Leveraging recent improvements in machine learning and spoken language processing, the performance of the SLU systems for the ATIS domain has improved dramatically. Around 5% error rate for the SLU task implies a solved problem. It is clear, however, that the problem of SLU is far from being solved, especially for more realistic, naturally-spoken utterances of a variety of speakers from tasks more complex than simple flight information requests. New data sets from such tasks can avoid over-tuning to one particular data set in terms of modeling and feature design.

The recent French Media corpus [23] offers a step towards this goal: it has three times more data and greater than a 10% concept error rate for SF. However, the data was not collected from an operational system. Instead, data was collected using a wizard of Oz setup with selected volunteers. Another effort is the Let's Go dialog system used by real users of the Pittsburgh bus transportation system [24]. However, SLU annotations are not yet available.

Even with such low error rates, the ATIS test set includes many example categories and sequences unseen in the training data, and the error rates have not converged yet. In that respect, more data from just the ATIS domain may be useful for SLU research.

The error analysis on the ATIS domain shows the primary weaknesses of the current n-gram-based modeling approaches: The local context overrides the global, the model has no domain knowledge to make any inferences, and it tries to fit any utterance into some known sample, hence not really robust to any out-of-domain utterances. This was also observed by Raymond and Riccardi [10], where the CRF model fits 100% to the training data. One possible research direction consists of employing longer distance syntactically or semantically motivated features, while preserving the robustness of the system to the noise introduced by the speech recognizer and variance due to natural language.

A lesser studied set of the ATIS corpus, Class D utterances, which are contextual queries, is another significant portion of this corpus, waiting to be understood. While most people treated understanding in context with handcrafted rules (e.g., [4]), to the best of our knowledge, the only study towards building a statistical discourse model has been proposed by Miller *et al.* [25].

8. ACKNOWLEDGMENTS

We would like to thank Christian Raymond and Giuseppe Riccardi for sharing the ATIS data, revised for annotation inconsistencies and mistakes.

9. REFERENCES

[1] 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 *Proceedings of the ICASSP*, San Francisco, CA, March 1992.

[2] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden understanding models of natural language," in *Proceedings of the ACL*, Las Cruces, NM, June 1994.

[3] W. Ward and S.Issar, "Recent improvements in the CMU spoken language understanding system," in *Proceedings of the ARPA HLT Workshop*, March 1994, pp. 213–216.

[4] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1, pp. 61–86, 1992.

[5] J. Dowding, J. M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran, "Gemini: A natural language system for spoken language understanding," in *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, NJ, March 1993.

[6] A. L. Gorin, G. Riccardi, and J. H. Wright, "How May I Help You?," *Speech Communication*, vol. 23, pp. 113–127, 1997.

[7] M. Jeong and G. G. Lee, "Exploiting non-local features for spoken language understanding," in *Proceedings of the ACL/COLING*, Sydney, Australia, July 2006.

[8] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the DARPA Workshop on Speech and Natural Language*, Hidden Valley, PA, June 1990.

[9] Y. He and S. Young, "A data-driven spoken language understanding system," in *Proceedings of the IEEE ASRU Workshop*, U.S. Virgin Islands, December 2003, pp. 583–588.

[10] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proceedings of the Interspeech*, Antwerp, Belgium, 2007.

[11] V. N. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, New York, NY, 1998.

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

[13] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the ICML*, Williamstown, MA, 2001.

[14] A. Stolcke and E. Shriberg, "Statistical language modeling for speech disfluencies," in *Proceedings of the ICASSP*, Atlanta, GA, May 1996.

[15] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Rahim, "The AT&T spoken language understanding system," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 213–222, 2006.

[16] D. Hakkani-Tür, G. Tur, and A. Chotimongkol, "Using syntactic and semantic graphs for call classification," in *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, Ann Arbor, MI, June 2005.

[17] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proceedings of the ICASSP*, Hong Kong, April 2003.

[18] H.-K. J. Kuo and C.-H. Lee, "Discriminative training in natural language call-routing," in *Proceedings of ICSLP*, Beijing, China, 2000.

[19] J. Chu-Carroll and B. Carpenter, "Vector-based natural language call routing," *Computational Linguistics*, vol. 25, no. 3, pp. 361–388, 1999.

[20] I. Zitouni, H.-K. J. Kuo, and C.-H. Lee, "Boosting and combincation of classifiers for natural language call routing systems," *Speech Communication*, vol. 41, no. 4, pp. 647–661, 2003.

[21] Stephen Cox, "Discriminative techniques in call routing," in *Proceedings of the ICASSP*, Hong Kong, April 2003.

[22] Y.-Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *Proceedings of the ICSLP*, Pittsburgh, PA, September 2006.

[23] H. Bonneau-Maynard, S. Rosset, C. Ayache, A. Kuhn, and D. Mostefa, "Semantic annotation of the French MEDIA dialog corpus," in *Proceedings of the Interspeech*, Lisbon, Portugal, September 2005.

[24] A. Raux, B. Langner, D. Bohus, A. Black, and M. Eskenazi, "Let's go public! taking a spoken dialog system to the real world," in *Proceedings of the Interspeech*, Lisbon, Portugal, September 2005.

[25] S. Miller, D. Stallard, R. Bobrow, and R. Schwartz, "A fully statistical approach to natural language interfaces," in *Proceedings of the ACL*, Morristown, NJ, 1996.

Position-aware Attention and Supervised Data Improve Slot Filling

Yuhao Zhang, Victor Zhong, Danqi Chen,
Gabor Angeli, Christopher D. Manning
Stanford University
Stanford, CA 94305
{yuhao, vzhong, danqi}@cs.stanford.edu
{angeli, manning}@cs.stanford.edu

Abstract

Organized relational knowledge in the form of "knowledge graphs" is important for many applications. However, the ability to populate knowledge bases with facts automatically extracted from documents has improved frustratingly slowly. This paper simultaneously addresses two issues that have held back prior work. We first propose an effective new model, which combines an LSTM sequence model with a form of entity position-aware attention that is better suited to relation extraction. Then we build TACRED, a large (106,264 examples) supervised relation extraction dataset, obtained via crowdsourcing and targeted towards TAC KBP relations. The combination of better supervised data and a more appropriate high-capacity model enables much better relation extraction performance. When the model trained on this new dataset replaces the previous relation extraction component of the best TAC KBP 2015 slot filling system, its F_1 score increases markedly from 22.2% to 26.7%.

1 Introduction

A basic but highly important challenge in natural language understanding is being able to populate a knowledge base with relational facts contained in a piece of text. For the text shown in Figure 1, the system should extract triples, or equivalently, knowledge graph edges, such as ⟨*Penner*, *per:spouse*, *Lisa Dillman*⟩. Combining such extractions, a system can produce a knowledge graph of relational facts between persons, organizations, and locations in the text. This task involves entity recognition, mention coreference and/or entity linking, and relation extraction; we focus on the most challenging "slot filling" task of filling in the relations between entities in the text.

Penner *is survived by his brother,* ***John****, a* ***copy editor*** *at the* ***Times****, and his former wife,* ***Times sportswriter Lisa Dillman****.*

Subject	Relation	Object
Mike Penner	per:spouse	Lisa Dillman
Mike Penner	per:siblings	John Penner
Lisa Dillman	per:title	Sportswriter
Lisa Dillman	per:employee_of	Los Angeles Times
John Penner	per:title	Copy Editor
John Penner	per:employee_of	Los Angeles Times

Figure 1: An example of relation extraction from the TAC KBP corpus.

Organized relational knowledge in the form of "knowledge graphs" has become an important knowledge resource. These graphs are now extensively used by search engine companies, both to provide information to end-users and internally to the system, as a way to understand relationships. However, up until now, automatic knowledge extraction has proven sufficiently difficult that most of the facts in these knowledge graphs have been built up by hand. It is therefore a key challenge to show that NLP technology can effectively contribute to this important problem.

Existing work on relation extraction (e.g., Zelenko et al., 2003; Mintz et al., 2009; Adel et al., 2016) has been unable to achieve sufficient recall or precision for the results to be usable versus hand-constructed knowledge bases. Supervised training data has been scarce and, while techniques like distant supervision appear to be a promising way to extend knowledge bases at low cost, in practice the training data has often been too noisy for reliable training of relation extraction systems (Angeli et al., 2015). As a result most systems fail to make correct extractions even in apparently straightforward cases like Figure 1,

Example	Entity Types & Label
Carey will succeed **Cathleen P. Black**, who held the position for 15 years and will take on a new role as **chairwoman** of Hearst Magazines, the company said.	**Types:** PERSON/TITLE **Relation:** *per:title*
Irene Morgan Kirkaldy, who was born and reared in **Baltimore**, lived on Long Island and ran a child-care center in Queens with her second husband, Stanley Kirkaldy.	**Types:** PERSON/CITY **Relation:** *per:city_of_birth*
Pandit worked at the brokerage Morgan Stanley for about 11 years until 2005, when he and some Morgan Stanley colleagues quit and later founded the hedge fund **Old Lane Partners**.	**Types:** ORGANIZATION/PERSON **Relation:** *org:founded_by*
Baldwin declined further comment, and said JetBlue chief **executive** Dave Barger was unavailable.	**Types:** PERSON/TITLE **Relation:** *no_relation*

Table 1: Sampled examples from the TACRED dataset. Subject entities are highlighted in blue and object entities are highlighted in red.

where the best system at the NIST TAC Knowledge Base Population (TAC KBP) 2015 evaluation failed to recognize the relation between *Penner* and *Dillman*.[1] Consequently most automatic systems continue to make heavy use of hand-written rules or patterns because it has been hard for machine learning systems to achieve adequate precision or to generalize as well across text types. We believe machine learning approaches have suffered from two key problems: (1) the models used have been insufficiently tailored to relation extraction, and (2) there has been insufficient annotated data available to satisfy the training of data-hungry models, such as deep learning models.

This work addresses both of these problems. We propose a new, effective neural network sequence model for relation classification. Its architecture is better customized for the slot filling task: the word representations are augmented by extra distributed representations of word position relative to the subject and object of the putative relation. This means that the neural attention model can effectively exploit the combination of semantic similarity-based attention and position-based attention. Secondly, we markedly improve the availability of supervised training data by using Mechanical Turk crowd annotation to produce a large supervised training dataset (Table 1), suitable for the common relations between people, organizations and locations which are used in the TAC KBP evaluations. We name this dataset the **TAC R**elation **E**xtraction **D**ataset (TACRED), and will make it available through the Linguistic Data Consortium (LDC) in order to respect copyrights on the underlying text.

Combining these two gives a system with markedly better slot filling performance. This is shown not only for a relation classification task on the crowd-annotated data but also for the incorporation of the resulting classifiers into a complete cold start knowledge base population system. On TACRED, our system achieves a relation classification F_1 score that is 5.7% higher than that of a strong feature-based classifier, and 2.4% higher than that of the best previous neural architecture that we re-implemented. When this model is used in concert with a pattern-based system on the TAC KBP 2015 Cold Start Slot Filling evaluation data, the system achieves an F_1 score of 26.7%, which exceeds the previous state-of-the-art by 4.5% absolute. While this performance certainly does not solve the knowledge base population problem – achieving sufficient recall remains a formidable challenge – this is nevertheless notable progress.

2 A Position-aware Neural Sequence Model Suitable for Relation Extraction

Existing work on neural relation extraction (e.g., Zeng et al., 2014; Nguyen and Grishman, 2015; Zhou et al., 2016) has focused on convolutional neural networks (CNNs), recurrent neural networks (RNNs), or their combination. While these models generally work well on the datasets they are tested on, as we will show, they often fail to generalize to the longer sentences that are common in real-world text (such as in TAC KBP).

We believe that existing model architectures suffer from two problems: (1) Although modern sequence models such as Long Short-Term Memory (LSTM) networks have gating mechanisms to control the relative influence of each individual word to the final sentence representation (Hochreiter and Schmidhuber, 1997), these controls are not explicitly conditioned on the entire sentence being classified; (2) Most existing work either

[1] Note: former spouses count as spouses in the ontology.

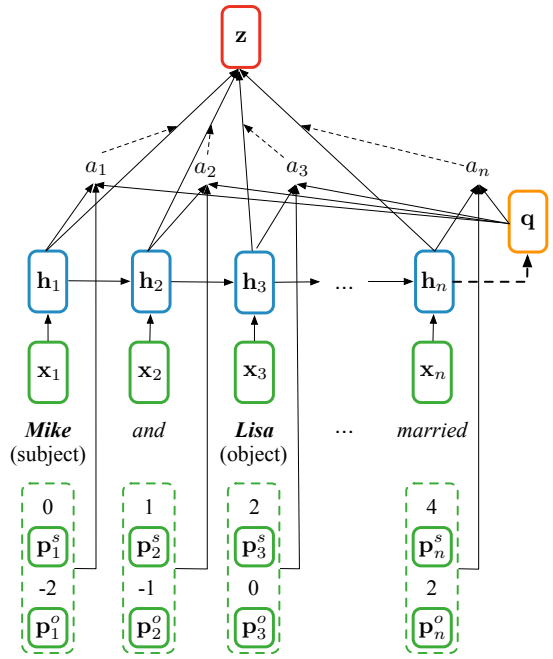


Figure 2: Our proposed position-aware neural sequence model. The model is shown with an example sentence *Mike and Lisa got married*.

does not explicitly model the positions of entities (i.e., subject and object) in the sequence, or models the positions only within a local region.

Here, we propose a new neural sequence model with a **position-aware attention mechanism** over an LSTM network to tackle these challenges. This model can (1) evaluate the relative contribution of each word after seeing the entire sequence, and (2) base this evaluation not only on the semantic information of the sequence, but also on the global positions of the entities within the sequence.

We formalize the relation extraction task as follows: Let $\mathcal{X} = [x_1, ..., x_n]$ denote a sentence, where x_i is the i-th token. A subject entity s and an object entity o are identified in the sentence, corresponding to two non-overlapping consecutive spans: $\mathcal{X}_s = [x_{s_1}, x_{s_1+1}, \ldots, x_{s_2}]$ and $\mathcal{X}_o = [x_{o_1}, x_{o_1+1}, \ldots, x_{o_2}]$. Given the sentence $\mathcal{X}$ and the positions of s and o, the goal is to predict a relation $r \in \mathcal{R}$ ($\mathcal{R}$ is the set of relations) that holds between s and o or no relation otherwise.

Inspired by the position encoding vectors used in Collobert et al. (2011) and Zeng et al. (2014), we define a position sequence relative to the subject entity $[p_1^s, ..., p_n^s]$, where

$$p_i^s = \begin{cases} i - s_1, & i < s_1 \\ 0, & s_1 \le i \le s_2 \\ i - s_2, & i > s_2 \end{cases} \quad (1)$$

Here s_1, s_2 are the starting and ending indices of the subject entity respectively, and $p_i^s \in \mathbb{Z}$ can be viewed as the relative distance of token x_i to the subject entity. Similarly, we obtain a position sequence $[p_1^o, ..., p_n^o]$ relative to the object entities.

Let $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ be word embeddings of the sentence, obtained using an embedding matrix $\mathbf{E}$. Similarly, we obtain position embedding vectors $\mathbf{p}^s = [\mathbf{p}_1^s, ..., \mathbf{p}_n^s]$ and $\mathbf{p}^o = [\mathbf{p}_1^o, ..., \mathbf{p}_n^o]$ using a shared position embedding matrix $\mathbf{P}$ respectively. Next, as shown in Figure 2, we obtain hidden state representations of the sentence by feeding $\mathbf{x}$ into an LSTM:

$$\{\mathbf{h}_1, ..., \mathbf{h}_n\} = \text{LSTM}(\{\mathbf{x}_1, ..., \mathbf{x}_n\}) \quad (2)$$

We define a *summary* vector $\mathbf{q} = \mathbf{h}_n$ (i.e., the output state of the LSTM). This summary vector encodes information about the entire sentence. Then for each hidden state $\mathbf{h}_i$, we calculate an attention weight a_i as:

$$u_i = \mathbf{v}^\top \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_q \mathbf{q} + \mathbf{W}_s \mathbf{p}_i^s + \mathbf{W}_o \mathbf{p}_i^o) \quad (3)$$

$$a_i = \frac{\exp(u_i)}{\sum_{j=1}^{n} \exp(u_j)} \quad (4)$$

Here $\mathbf{W}_h, \mathbf{W}_q \in \mathbb{R}^{d_a \times d}$, $\mathbf{W}_s, \mathbf{W}_o \in \mathbb{R}^{d_a \times d_p}$ and $\mathbf{v} \in \mathbb{R}^{d_a}$ are learnable parameters of the network, where d is the dimension of hidden states, d_p is the dimension of position embeddings, and d_a is the size of attention layer. Additional parameters of the network include embedding matrices $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{P} \in \mathbb{R}^{(2L-1) \times d_p}$, where $\mathcal{V}$ is the vocabulary and L is the maximum sentence length.

We regard attention weight a_i as the relative contribution of the specific word to the sentence representation. The final sentence representation $\mathbf{z}$ is computed as:

$$\mathbf{z} = \sum_{i=1}^{n} a_i \mathbf{h}_i \quad (5)$$

$\mathbf{z}$ is later fed into a fully-connected layer followed by a softmax layer for relation classification.

Note that our model significantly differs from the attention mechanism in Bahdanau et al. (2015) and Zhou et al. (2016) in our use of the summary vector and position embeddings, and the way our attention weights are computed. An intuitive way to understand the model is to view the attention calculation as a selection process, where the goal is to select relevant contexts over irrelevant ones.

Dataset	# Rel.	# Ex.	% Neg.
SemEval-2010 Task 8	19	10,717	17.4%
ACE 2003–2004	24	16,771	N/A
TACRED	42	106,264	79.5%

Table 2: A comparison of existing datasets and our proposed TACRED dataset. % Neg. denotes the percentage of negative examples (no relation).

Here the summary vector ($\mathbf{q}$) helps the model to base this selection on the semantic information of the entire sentence (rather than on each word only), while the position vectors ($\mathbf{p}_i^s$ and $\mathbf{p}_i^o$) provides important spatial information between each word and the entities.

3 The TAC Relation Extraction Dataset

Previous research has shown that slot filling systems can greatly benefit from supervised data. For example, Angeli et al. (2014b) showed that even a small amount of supervised data can boost the end-to-end F_1 score by 3.9% on the TAC KBP tasks. However, existing relation extraction datasets such as the SemEval-2010 Task 8 dataset (Hendrickx et al., 2009) and the Automatic Content Extraction (ACE) (Strassel et al., 2008) dataset are less useful for this purpose. This is mainly because: (1) these datasets are relatively small for effectively training high-capacity models (see Table 2), and (2) they capture very different types of relations. For example, the SemEval dataset focuses on semantic relations (e.g., *Cause-Effect*, *Component-Whole*) between two nominals.

One can further argue that it is easy to obtain a large amount of training data using distant supervision (Mintz et al., 2009). In practice, however, due to the large amount of noise in the induced data, training relation extractors that perform well becomes very difficult. For example, Riedel et al. (2010) show that up to 31% of the distantly supervised labels are wrong when creating training data from aligning Freebase to newswire text.

To tackle these challenges, we collect a large supervised dataset TACRED, targeted towards the TAC KBP relations.

Data collection. We create TACRED based on query entities and annotated system responses in the yearly TAC KBP evaluations. In each year of the TAC KBP evaluation (2009–2015), 100 entities (people or organizations) are given as queries, for which participating systems should find associated relations and object entities. We make use of Mechanical Turk to annotate each sentence in the source corpus that contains one of these query entities. For each sentence, we ask crowd workers to annotate both the subject and object entity spans and the relation types.

Data Split	# Ex.	Years
Train	68,124	2009–2012
Dev	22,631	2013
Test	15,509	2014

Table 3: Statistics on TACRED: number of examples and the source of each portion.

Dataset stratification. In total we collect 106,264 examples. We stratify TACRED across years in which the TAC KBP challenge was run, and use examples corresponding to query entities from 2009 to 2012 as training split, 2013 as development split, and 2014 as test split. We reserve the TAC KBP 2015 evaluation data for running slot filling evaluations, as presented in Section 4. Detailed statistics are given in Table 3.

Discussion. Table 1 presents sampled examples from TACRED. Compared to existing datasets, TACRED has four advantages. First, it contains an order of magnitude more relation instances (Table 2), enabling the training of expressive models. Second, we reuse the entity and relation types of the TAC KBP tasks. We believe these relation types are of more interest to downstream applications. Third, we fully annotate all negative instances that appear in our data collection process, to ensure that models trained on TACRED are not biased towards predicting false positives on real-world text. Lastly, the average sentence length in TACRED is 36.4, compared to 19.1 in the SemEval dataset, reflecting the complexity of contexts in which relations occur in real-world text.

Due to space constraints, we describe the data collection and validation process, system interfaces, and more statistics and examples of TACRED in the supplementary material. We will make TACRED publicly available through the LDC.

4 Experiments

In this section we evaluate the effectiveness of our proposed model and TACRED on improving slot

filling systems. Specifically, we run two sets of experiments: (1) we evaluate model performance on the relation extraction task using TACRED, and (2) we evaluate model performance on the TAC KBP 2015 cold start slot filling task, by training the models on TACRED.

4.1 Baseline Models

We compare our model against the following baseline models for relation extraction and slot filling:

TAC KBP 2015 winning system. To judge our proposed model against a strong baseline, we compare against Stanford's top performing system on the TAC KBP 2015 cold start slot filling task (Angeli et al., 2015). At the core of this system are two relation extractors: a pattern-based extractor and a logistic regression (LR) classifier. The pattern-based system uses a total of 4,528 surface patterns and 169 dependency patterns. The logistic regression model was trained on approximately 2 million bootstrapped examples (using a small annotated dataset and high-precision pattern system output) that are carefully tuned for TAC KBP slot filling evaluation. It uses a comprehensive feature set similar to the MIML-RE system for relation extraction (Surdeanu et al., 2012), including lemmatized n-grams, sequence NER tags and POS tags, positions of entities, and various features over dependency paths, etc.

Convolutional neural networks. We follow the 1-dimensional CNN architecture by Nguyen and Grishman (2015) for relation extraction. This model learns a representation of the input sentence, by first running a series of convolutional operations on the sentence with various filters, and then feeding the output into a max-pooling layer to reduce the dimension. The resulting representation is then fed into a fully-connected layer followed by a softmax layer for relation classification. As an extension, positional embeddings are also introduced into this model to better capture the relative position of each word to the subject and object entities and were shown to achieve improved results. We use "CNN-PE" to represent the CNN model with positional embeddings.

Dependency-based recurrent neural networks. In dependency-based neural models, shortest dependency paths between entities are often used as input to the neural networks. The intuition is to eliminate tokens that are potentially less relevant to the classification of the relation. For the example in Figure 1, the shortest dependency path between the two entities is:

$$\text{[Penner]} \leftarrow \text{survived} \rightarrow \text{brother} \rightarrow \text{wife} \rightarrow \text{[Lisa Dillman]}$$

We follow the SDP-LSTM model proposed by Xu et al. (2015b). In this model, each shortest dependency path is divided into two separate sub-paths from the subject entity and the object entity to the lowest common ancestor node. Each sub-path is fed into an LSTM network, and the resulting hidden units at each word position are passed into a max-over-time pooling layer to form the output of this sub-path. Outputs from the two sub-paths are then concatenated to form the final representation.

In addition to the above models, we also compare our proposed model against an LSTM sequence model without attention mechanism.

4.2 Implementation Details

We map words that occur less than 2 times in the training set to a special $<$*UNK*$>$ token. We use the pre-trained GloVe vectors (Pennington et al., 2014) to initialize word embeddings. For all the LSTM layers, we find that 2-layer stacked LSTMs generally work better than one-layer LSTMs. We minimize cross-entropy loss over all 42 relations using *AdaGrad* (Duchi et al., 2011). We apply Dropout with $p = 0.5$ to CNNs and LSTMs. During training we also find a word dropout strategy to be very effective: we randomly set a token to be $<$*UNK*$>$ with a probability p. We set p to be 0.06 for the SDP-LSTM model and 0.04 for all other models.

Entity masking. We replace each subject entity in the original sentence with a special $<$*NER*$>$-*SUBJ* token where $<$*NER*$>$ is the corresponding NER signature of the subject as provided in TACRED. We do the same processing for object entities. This processing step helps (1) provide a model with entity type information, and (2) prevent a model from overfitting its predictions to specific entities.

Multi-channel augmentation. Instead of using only word vectors as input to the network, we augment the input with part-of-speech (POS) and named entity recognition (NER) embeddings. We run Stanford CoreNLP (Manning et al., 2014) to obtain the POS and NER annotations.

	Model	P	R	F_1
Traditional	Patterns	**86.9**	23.2	36.6
	LR	73.5	49.9	59.4
	LR + Patterns	72.9	51.8	60.5
Neural	CNN	75.6	47.5	58.3
	CNN-PE	70.3	54.2	61.2
	SDP-LSTM	66.3	52.7	58.7
	LSTM	65.7	59.9	62.7
	Our model	65.7	**64.5**	**65.1**
	Ensemble	70.1	**64.6**	**67.2**

Table 4: Model performance on the test set of TACRED, micro-averaged over instances. LR = Logistic Regression.

We describe our model hyperparameters and training in detail in the supplementary material.

4.3 Evaluation on TACRED

We first evaluate all models on TACRED. We train each model for 5 separate runs with independent random initializations. For each run we perform early stopping using the dev set. We then select the run (among 5) that achieves the *median* F_1 score on the dev set, and report its test set performance.

Table 4 summarizes our results. We observe that all neural models achieve higher F_1 scores than the logistic regression and patterns systems, which demonstrates the effectiveness of neural models for relation extraction. Although positional embeddings help increase the F_1 by around 3% over the plain CNN model, a simple (2-layer) LSTM model performs surprisingly better than CNN and dependency-based models. Lastly, our proposed position-aware mechanism is very effective and achieves an F_1 score of 65.1%, with an absolute increase of 2.4% over the best baseline neural model (LSTM) and 5.7% over the baseline logistic regression system. We also run an ensemble of our position-aware attention model which takes majority votes from 5 runs with random initializations and it further pushes the F_1 score up by 2.1%.

We find that different neural architectures show a different balance between precision and recall. CNN-based models tend to have higher precision; RNN-based models have better recall. This can be explained by noting that the filters in CNNs are essentially a form of "fuzzy n-gram patterns".

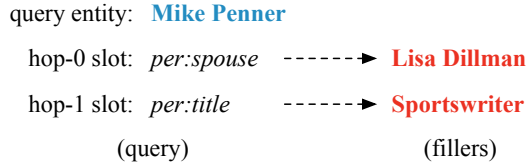


Figure 3: An example query and corresponding fillers in the TAC KBP cold start slot filling task.

4.4 Evaluation on TAC KBP Slot Filling

Second, we evaluate the slot filling performance of all models using the TAC KBP 2015 cold start slot filling task (Ellis et al., 2015). In this task, about 50k newswire and Web forum documents are selected as the evaluation corpus. A slot filling system is asked to answer a series of queries with two-hop slots (Figure 3): The first slot asks about fillers of a relation with the query entity as the subject (Mike Penner), and we term this a **hop-0** slot; the second slot asks about fillers with the system's hop-0 output as the subject, and we term this a **hop-1** slot. System predictions are then evaluated against gold annotations, and micro-averaged precision, recall and F_1 scores are calculated at the hop-0 and hop-1 levels. Lastly **hop-all** scores are calculated by combining hop-0 and hop-1 scores.[2]

Evaluating relation extraction systems on slot filling is particularly challenging in that: (1) End-to-end cold start slot filling scores conflate the performance of all modules in the system (i.e., entity recognizer, entity linker and relation extractor). (2) Errors in hop-0 predictions can easily propagate to hop-1 predictions. To fairly evaluate each relation extraction model on this task, we use Stanford's 2015 slot filling system as our basic pipeline.[3] It is a very strong baseline specifically tuned for TAC KBP evaluation and ranked top in the 2015 evaluation. We then plug in the corresponding relation extractor trained on TACRED, keeping all other modules unchanged.

Table 5 presents our results. We find that: (1) by only training our logistic regression model on TACRED (in contrast to on the 2 million bootstrapped examples used in the 2015 Stanford system) and combining it with patterns, we obtain a higher hop-0 F_1 score than the 2015 Stanford sys-

[2] In the TAC KBP cold start slot filling evaluation, a hop-1 slot is transferred to a pseudo-slot which is treated equally as a hop-0 slot. Hop-all precision, recall and F1 are then calculated by combining these pseudo-slot predictions and hop-0 predictions.

[3] This system uses the fine-grained NER system in Stanford CoreNLP (Manning et al., 2014) for entity detection and the Illinois Wikifier (Ratinov et al., 2011) for entity linking.

Model	Hop-0			Hop-1			Hop-all		
	P	R	F_1	P	R	F_1	P	R	F_1
Patterns	**63.8**	17.7	27.7	**49.3**	8.6	14.7	**58.9**	13.3	21.8
LR	36.6	21.9	27.4	15.1	10.1	12.2	25.6	16.3	19.9
+ Patterns (2015 winning system)	37.5	24.5	29.7	16.5	12.8	14.4	26.6	19.0	22.2
LR trained on TACRED	32.7	20.6	25.3	7.9	9.5	8.6	16.8	15.3	16.0
+ Patterns	36.5	26.5	30.7	11.0	15.3	12.8	20.1	21.2	20.6
Our model	39.0	28.9	33.2	17.7	13.9	15.6	28.2	21.5	24.4
+ Patterns	40.2	**31.5**	**35.3**	19.4	**16.5**	**17.8**	29.7	**24.2**	**26.7**

Table 5: Model performance on TAC KBP 2015 slot filling evaluation, micro-averaged over queries. Hop-0 scores are calculated on the simple single-hop slot filling results; hop-1 scores are calculated on slot filling results chained on systems' hop-0 predictions; hop-all scores are calculated based on the combination of the two. LR = logistic regression.

Model	Dev F_1
Final Model	**66.0**
– Position-aware attention	65.2
– Attention	64.1
– Pre-trained embeddings	64.3
– Word dropout	65.4
– All above	62.2

Table 6: An ablation test of our position-aware attention model, evaluated on TACRED dev set. Scores are median of 5 models.

tem, and a similar hop-all F_1; (2) our proposed position-aware attention model substantially outperforms the 2015 Stanford system on all hop-0, hop-1 and hop-all F_1 scores. Combining it with the patterns, we achieve a hop-all F_1 of 26.7%, an absolute improvement of 4.5% over the previous state-of-the-art result.

4.5 Analysis

Model ablation. Table 6 presents the results of an ablation test of our position-aware attention model on the development set of TACRED. The entire attention mechanism contributes about 1.9 F_1, where the position-aware term in Eq. (3) alone contributes about 0.8 F_1 score.

Impact of negative examples. Figure 4 shows how the slot filling evaluation scores change as we change the amount of negative (i.e., *no_relation*) training data provided to our proposed model. We find that: (1) At hop-0 level, precision increases as we provide more negative examples, while recall stays almost unchanged. F_1 score keeps increasing. (2) At hop-all level, F_1 score increases by about 10% as we change the amount of negative examples from 20% to 100%.

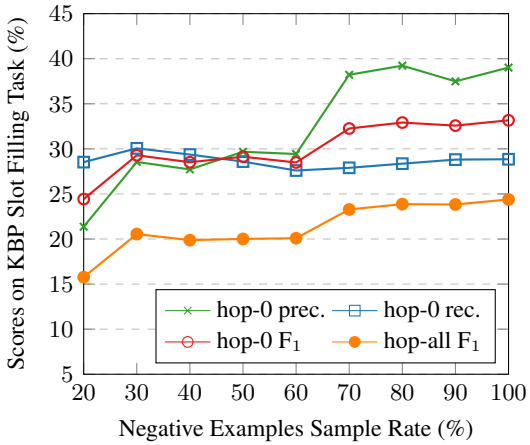


Figure 4: Change of slot filling hop-0 and hop-all scores as number of negative training examples changes. 100% is with all the negative examples included in the training set; the left side scores have positives and negatives roughly balanced.

Performance by sentence length. Figure 5 shows performance on varying sentence lengths. We find that: (1) Performance of all models degrades substantially as the sentences get longer. (2) Compared to the baseline Logistic Regression model, all neural models handle long sentences better. (3) Compared to CNN-PE model, RNN-based models are more robust on long sentences, and notably SDP-LSTM model is least sensitive to sentence length. (4) Our proposed model achieves equal or better results on sentences of all lengths, except for sentences with more than 60 tokens where SDP-LSTM model achieves the best result.

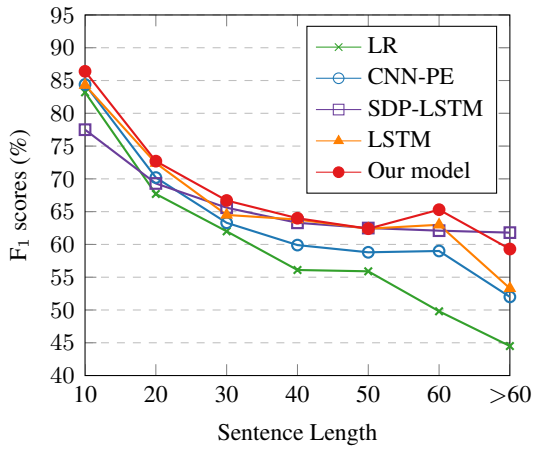


Figure 5: TACRED development set F_1 scores for sentences of varying lengths.

Improvement by slot types. We calculate the F_1 score for each slot type and compare the improvement from using our proposed model across slot types. When compared with the CNN-PE model, our position-aware attention model achieves improved F_1 scores on 30 out of the 41 slot types, with the top 5 slot types being *org:members*, *per:country_of_death*, *org:shareholders*, *per:children* and *per:religion*. When compared with SDP-LSTM model, our model achieves improved F_1 scores on 26 out of the 41 slot types, with the top 5 slot types being *org:political/religious_affiliation*, *per:country_of_death*, *org:alternate_names*, *per:religion* and *per:alternate_names*. We observe that slot types with relatively sparse training examples tend to be improved by using the position-aware attention model.

Attention visualization. Lastly, Figure 6 shows the visualization of attention weights assigned by our model on sampled sentences from the development set. We find that the model learns to pay more attention to words that are informative for the relation (e.g., "graduated from", "niece" and "chairman"), though it still makes mistakes (e.g., "refused to name the three"). We also observe that the model tends to put a lot of weight onto object entities, as the object NER signatures are very informative to the classification of relations.

5 Related Work

Relation extraction. There are broadly three main lines of work on relation extraction: first, fully-supervised approaches (Zelenko et al., 2003; Bunescu and Mooney, 2005), where a statistical classifier is trained on an annotated dataset; second, distant supervision (Mintz et al., 2009; Surdeanu et al., 2012), where a training set is formed by projecting the relations in an existing knowledge base onto textual instances that contain the entities that the relation connects; and third, Open IE (Fader et al., 2011; Mausam et al., 2012), which views its goal as producing subject-relation-object triples and expressing the relation in text.

Slot filling and knowledge base population. The most widely-known effort to evaluate slot filling and KBP systems is the yearly TAC KBP slot filling tasks, starting from 2009 (McNamee and Dang, 2009). Participants in slot filling tasks usually make use of hybrid systems that combine patterns, Open IE, distant supervision and supervised systems for relation extraction (Kisiel et al., 2015; Finin et al., 2015; Zhang et al., 2016).

Datasets for relation extraction. Popular general-domain datasets include the ACE dataset (Strassel et al., 2008) and the SemEval-2010 task 8 dataset (Hendrickx et al., 2009). In addition, the BioNLP Shared Tasks (Kim et al., 2009) are yearly efforts on creating datasets and evaluations for biomedical information extraction systems.

Deep learning models for relation extraction. Many deep learning models have been proposed for relation extraction, with a focus on end-to-end training using CNNs (Zeng et al., 2014; Nguyen and Grishman, 2015) and RNNs (Zhang et al., 2015). Other popular approaches include using CNN or RNN over dependency paths between entities (Xu et al., 2015a,b), augmenting RNNs with different components (Xu et al., 2016; Zhou et al., 2016), and combining RNNs and CNNs (Vu et al., 2016; Wang et al., 2016). Adel et al. (2016) compares the performance of CNN models against traditional approaches on slot filling using a portion of the TAC KBP evaluation data.

6 Conclusion

We introduce a state-of-the-art position-aware neural sequence model for relation extraction, as well as TACRED, a large-scale, crowd-sourced dataset that is orders of magnitude larger than previous relation extraction datasets. Our proposed model outperforms a strong feature-based classifier and all baseline neural models. In combination with the new dataset, it improves the state-of-the-

Sampled Sentences	Predicted Labels
PER-SUBJ graduated from North Korea 's elite Kim Il Sung University and ORG-OBJ ORG-OBJ .	*per:schools_attended*
The cause was a heart attack following a case of pneumonia , said PER-SUBJ 's niece , PER-OBJ PER-OBJ .	*per:other_family*
Independent ORG-SUBJ ORG-SUBJ ORG-SUBJ (ECC) chairman PER-OBJ PER-OBJ refused to name the three , saying they would be identified when the final list of candidates for the august 20 polls is published on Friday .	*org:top_members/employees*

Figure 6: Sampled sentences from the TACRED development set, with words highlighted according to the attention weights produced by our best model.

art hop-all F_1 on the TAC KBP 2015 slot filling task by 4.5% absolute.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions. We gratefully acknowledge the support of the Allen Institute for Artificial Intelligence and the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract No. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Gabor Angeli, Sonal Gupta, Melvin Johnson Premkumar, Christopher D. Manning, Christopher Ré, Julie Tibshirani, Jean Y. Wu, Sen Wu, and Ce Zhang. 2014a. Stanford's distantly supervised slot filling systems for KBP 2014. In *Text Analysis Conference (TAC) Proceedings 2014*.

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. 2014b. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *Text Analysis Conference (TAC) Proceedings 2015*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 724–731.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Text Analysis Conference (TAC) Proceedings 2015*, pages 16–17.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1535–1545.

Tim Finin, Dawn Lawrie, Paul McNamee, James Mayfield, Douglas Oard, Nanyun Peng, Ning Gao, Yiu-Chang Lin, Joshi MacKin, and Tim Dowd. 2015.

HLTCOE participation in TAC KBP 2015: Cold start and TEDL. In *Text Analysis Conference (TAC) Proceedings 2015*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9.

Bryan Kisiel, Bill McDowell, Matt Gardner, Ndapandula Nakashole, Emmanouil A Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, and Tom Mitchell. 2015. CMUML System for KBP 2015 cold start slot filling. In *Text Analysis Conference (TAC) Proceedings 2015*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In *Text Analysis Conference (TAC) Proceedings 2009*, volume 17, pages 111–113.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*, pages 39–48.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, volume 14, pages 1532–1543.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*, pages 1375–1384.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stephanie Strassel, Mark A Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with

data augmentation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1785–1794.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*, pages 2335–2344.

Dongxu Zhang, Dong Wang, and Rong Liu. 2015. Relation classification via recurrent neural network. Technical report, CSLT 20150024, Tsinghua University.

Yuhao Zhang, Arun Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, and Christopher D. Manning. 2016. Stanford at TAC KBP 2016: Sealing pipeline leaks and understanding chinese. In *Text Analysis Conference (TAC) Proceedings 2016*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, page 207.

A TACRED Data Collection and Validation

In this appendix, we describe the way we collect and validate TACRED in full detail.

A.1 Data Collection

TACRED leverages the work done selecting query entities and annotating system responses in the TAC KBP evaluations. In each year of the TAC KBP evaluation (2009–2015), 100 query entities are given to participating KBP systems with the aim of filling in valid knowledge base entries for these entities. Our annotation effort re-uses these query entities, annotating each sentence in the source corpus that contains one of these entities. Given the set of mention pairs (e.g., *Penner* and *Lisa Dillman*) containing an evaluation entity, the mention pair can have either 1) been extracted during a previous KBP competition and marked correct by an LDC annotator, or 2) been generated automatically from candidate mention pairs in the corpus. For clarity, we refer to the former as **LDC examples** and the latter as **generated examples**, and describe them separately.

LDC examples. For examples in this category, although the relations have been annotated by an LDC annotator, the provenance for the mention pairs provided in TAC KBP evaluation files are often too general or imprecise; for example in early years only the document that contains a mention pair is given as provenance. We solve this problem with a two-stage annotation task (HIT) in Mechanical Turk: In the first task, Turk annotators are provided with the mention pair and its relation (annotated by LDC), and asked to find a sentence in the document that expresses the extraction. In the second task, annotators are asked to identify the spans of both the subject and object entities. See Figure 7 and Figure 8 for example interfaces provided to Turk annotators.

Generated examples. To further collect examples that are not annotated by LDC, we first run annotations on the corpus using a combination of Stanford's statistical coreference system (Clark and Manning, 2015) and the Illinois Wikifier (Ratinov et al., 2011). Then we collect all mention pairs in which one mention is linked to one of the query entities by the entity linker. To prevent the resulting dataset from being skewed towards commonly occurring query entities such as "Barack Obama", we enforce a hard upper limit on the number of collected mention pairs containing a query entity. Specifically, for each query entity q, we retrieve N_q sentences from the KBP corpus that contain an entity mention linked to q. Then let N_{q^c} denote the number of extractions submitted by competing KBP systems that were also deemed correct by human annotators, we want N_q to be proportional to N_{q^c}, and heuristically set: $N_q = \min(9 \cdot N_{q^c}, 300)$. Next, each mention pair, along with the corresponding sentence in which it occurs, is annotated for its relation type (or *no_relation*) as a task on Mechanical Turk. Figure 9 shows an example task interface for generated examples on Mechanical Turk.

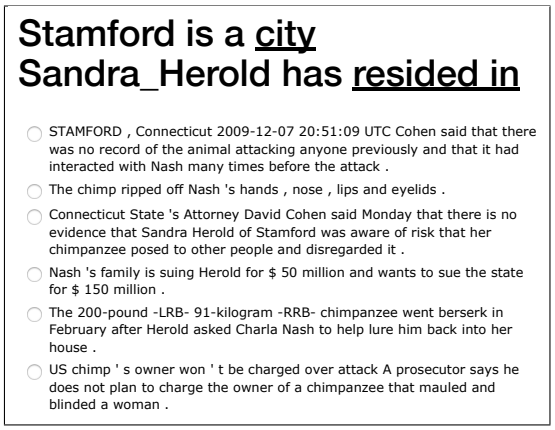


Figure 7: Example of an LDC examples HIT on Mechanical Turk for identifying the relevant sentence. The annotator is presented with every sentence from the document as well as the extraction for which to find the sentence.

A.2 Data Validation

In order to maintain the quality of TACRED, we validate the collected data both during and after the annotation process. We made use of crowdsourced data from a previous annotation effort on the same relation set (Angeli et al., 2014a). During annotation, 10% of the HITs presented to a worker are sanity check examples from this previous data, and annotators whose error rate on these examples exceeds 25% were asked to have their work re-annotated.

After the data collection is done, one of the authors manually examined 300 sampled instances. The estimated annotation accuracy is 93.3%, with a confidence interval of (89.9%, 95.9%). In addition, for the collected generated examples, we estimate inter-annotator agreement using 761 sampled

Figure 8: Example of an LDC examples HIT on Mechanical Turk for identifying the mention spans. The annotator is presented with a sentence obtained from the HIT shown in Figure 7 as well as the corresponding extraction and asked to identify the spans of the subject and object mentions in the extraction.

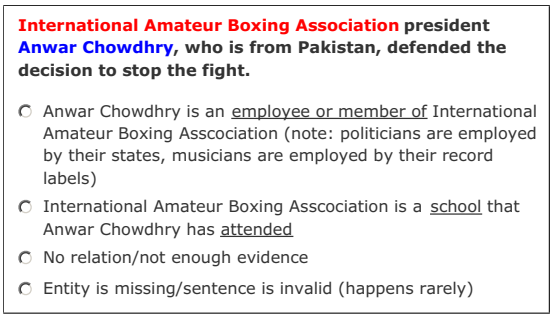


Figure 9: Example of a generated examples HIT. The subject entity is highlighted in blue and the object entity is highlighted in red. The annotator is asked to select among a set of plausible relations that are compatible with the subject and object entity types, along with an option to state that none of the presented relations hold.

mention pairs shown to five annotators. Results are shown in Table 7.

A.3 Data Statistics

In total, we collect 10,691 annotations from the LDC examples task and 110,021 annotations from the generated examples task. After removing duplicates and examples where the subject and object entities overlap, we arrive at a total of 106,264 examples. About 79.5% of all examples are annotated as *no_relation*, which we showed to be crucial for training high-precision relation extraction models for the TAC KBP 2015 slot filling evaluation. Furthermore, we find that sentences in TACRED tend to be much longer than in the SemEval

Metric	Score
5 annotators agree	74.2%
$\geq$ 4 annotators agree	90.5%
$\geq$ 3 annotators agree	100.0%
Fleiss Kappa	54.4%

Table 7: Estimated inter-annotator agreement using 761 sampled mention pairs.

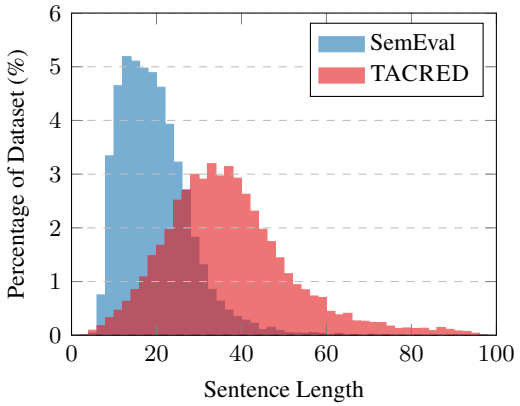


Figure 10: Distribution of sentence lengths in SemEval 2010 task 8 and TACRED.

dataset (Figure 10).

Table 8 presents detailed statistics on this dataset. We also include sampled training examples in Table 9.

B Model Training Details

Here we describe the way we train our models in detail for replicability.

Model hyperparameters. We use 200 for word embedding size and 30 for every other embedding (i.e., position, POS or NER) size. For CNN models, we use filter window sizes ranging from 2 to 5, and 500 filters for each window size. For the SDP-LSTM model, in addition to POS and NER embeddings, we also include the type of dependency edges as an additional embedding channel. For our proposed position-aware neural sequence model, we use attention size of 200. For all models that require LSTM layers, we find a 2-layer stacked LSTMs works better than a single-layer LSTM. We use one-directional LSTM layers in all of our experiments. Empirically we find bi-directional LSTM layers give no improvement to our proposed position-aware sequence model and marginal improvement to the simple LSTM model. We do not add max-pooling layers after

LSTM layers as we find this harms the performance.

Training. During training, we employ standard dropout (Srivastava et al., 2014) for CNN models, and RNN dropout (Zaremba et al., 2014) for LSTM models. Additionally, for CNN models we apply ℓ_2 regularization with coefficient 10^{-3} to all filters to avoid overfitting. We use *AdaGrad* (Duchi et al., 2011) with a learning rate of 0.1 for CNN models and 1.0 for all other models. We train CNN models for 50 epochs and other models for 30 epochs, with a mini-batch size of 50. We monitor the training process by looking at the micro-averaged F_1 score on the dev set. Starting from the 20th epoch, we decrease the learning rate with a decay rate of 0.9 if the dev set micro-averaged F_1 score does not increase after every epoch. Finally, we evaluate the model that achieves the best dev set F_1 score on the test set.

Relation	Total	Percentage	Train 2009–2012	Development 2013	Test 2014
no_relation	84491	79.51%	55112	17195	12184
org:alternate_names	1359	1.28%	808	338	213
org:city_of_headquarters	573	0.54%	382	109	82
org:country_of_headquarters	753	0.71%	468	177	108
org:dissolved	33	0.03%	23	8	2
org:founded	166	0.16%	91	38	37
org:founded_by	268	0.25%	124	76	68
org:member_of	171	0.16%	122	31	18
org:members	286	0.27%	170	85	31
org:number_of_employees/members	121	0.11%	75	27	19
org:parents	444	0.42%	286	96	62
org:political/religious_affiliation	125	0.12%	105	10	10
org:shareholders	144	0.14%	76	55	13
org:stateorprovince_of_headquarters	350	0.33%	229	70	51
org:subsidiaries	453	0.43%	296	113	44
org:top_members/employees	2770	2.61%	1890	534	346
org:website	223	0.21%	111	86	26
per:age	833	0.78%	390	243	200
per:alternate_names	153	0.14%	104	38	11
per:cause_of_death	337	0.32%	117	168	52
per:charges	280	0.26%	72	105	103
per:children	347	0.33%	211	99	37
per:cities_of_residence	742	0.70%	374	179	189
per:city_of_birth	103	0.10%	65	33	5
per:city_of_death	227	0.21%	81	118	28
per:countries_of_residence	819	0.77%	445	226	148
per:country_of_birth	53	0.05%	28	20	5
per:country_of_death	61	0.06%	6	46	9
per:date_of_birth	103	0.10%	63	31	9
per:date_of_death	394	0.37%	134	206	54
per:employee_of	2163	2.04%	1524	375	264
per:origin	667	0.63%	325	210	132
per:other_family	319	0.30%	179	80	60
per:parents	296	0.28%	152	56	88
per:religion	153	0.14%	53	53	47
per:schools_attended	229	0.22%	149	50	30
per:siblings	250	0.24%	165	30	55
per:spouse	483	0.45%	258	159	66
per:stateorprovince_of_birth	72	0.07%	38	26	8
per:stateorprovince_of_death	104	0.10%	49	41	14
per:stateorprovinces_of_residence	484	0.46%	331	72	81
per:title	3862	3.63%	2443	919	500
Total	106264	100.00	68124	22631	15509

Table 8: Relation distribution of the TACRED dataset.

Example Sentences	Subject Type	Object Type	Relation Labels
Carey will succeed **Cathleen P. Black**, who held the position for 15 years and will take on a new role as *chairwoman* of Hearst Magazines, the company said.	Person	Title	per:title
Baldwin declined further comment, and said JetBlue chief *executive* Dave Barger was unavailable.	Person	Title	no_relation
Irene Morgan Kirkaldy, who was born and reared in Baltimore, lived on Long Island and ran a child-care center in Queens with her second husband, *Stanley Kirkaldy*.	Person	Person	per:spouse
Cummings, current holder of the Seventh District seat held by *Mr. Mitchell*, sponsored legislation last year that named a Baltimore post office in the veteran congressman's honor.	Person	Person	no_relation
Blackburn Rovers announced Tuesday they had sacked *Paul Ince* as their manager, a statement on the Premier League club's website said.	Organization	Person	org:top_members/employees
Kerry wrote a letter to *Pickens*, saying he would donate any proceeds to the **Paralyzed Veterans of America**, the Associated Press reported.	Organization	Person	no_relation
Forsberg launched the anti-nuclear movement with a paper she wrote while obtaining a doctorate in international studies at *Massachusetts Institute of Technology*.	Person	Organization	per:schools_attended
He received an undergraduate degree from Morgan State University in 1950 and applied for admission to graduate school at the *University of Maryland in College Park*.	Person	Organization	no_relation

Table 9: Sampled training examples from the TACRED dataset, with subject entity highlighted in bold and blue and object entities highlighted in italics and red.

Note on Dataset Issue:
We noticed that there were duplicated examples in the original version of the TACRED dataset. We therefore updated the dataset by removing all duplicates and updated the dataset statistics and experiment results in the paper accordingly, with most changes in Table 4, Table 6 and Table 8.

In summary, the F_1 scores of all models have slightly changed, yet no conclusion has changed.

Sequential Convolutional Neural Networks for Slot Filling in Spoken Language Understanding

Ngoc Thang Vu

Institute of Natural Language Processing, University of Stuttgart

thangvu@ims.uni-stuttgart.de

Abstract

We investigate the usage of convolutional neural networks (CNNs) for the slot filling task in spoken language understanding. We propose a novel CNN architecture for sequence labeling which takes into account the previous context words with preserved order information and pays special attention to the current word with its surrounding context. Moreover, it combines the information from the past and the future words for classification. Our proposed CNN architecture outperforms even the previously best ensembling recurrent neural network model and achieves state-of-the-art results with an F1-score of 95.61% on the ATIS benchmark dataset without using any additional linguistic knowledge and resources.

Index Terms: spoken language understanding, convolutional neural networks

1. Introduction

The slot filling task in spoken language understanding (SLU) is to assign a semantic concept to each word in a sentence. In the sentence *I want to fly from Munich to Rome*, an SLU system should tag *Munich* as the departure city of a trip and *Rome* as the arrival city. All the other words, which do not correspond to real slots, are then tagged with an artificial class O. Traditional approaches for this task used generative models, such as hidden markov models (HMM) [1], or discriminative models, such as conditional random fields (CRF) [2, 3]. More recently, neural network (NN) models, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been applied successfully to this task [4, 5, 6, 7, 8].

Overall, RNNs outperformed other NN models and achieved the state-of-the-art results on the ATIS benchmark dataset [9]. Furthermore, bi-directional RNNs have worked best so far showing that information from both the past and the future is important in predicting the semantic label of the current word. It is, however, well known that it is difficult to train an RNN due to the vanishing gradient problem [10]. Introducing long short-term memory (LSTM) [11] or other variants of LSTM such as the gated recurrent unit (GRU) can solve this problem but, in turn increases the number of parameters significantly. Previous results reported in [8] did not show any improvement on the ATIS data set using LSTM or GRU.

In contrast to previous papers which reported state-of-the-art results with RNNs, we explore the usage of convolutional neural networks for a sequence labeling task like slot filling. Previous research in [6] showed promising results on the slot filling task. The motivation behind this is to allow the model to search for patterns in order to predict the label of the current word independent of the feature representation of the previous word. Moreover, CNNs provide several advantages: it preserves the word order information, it is faster and easier to train and does not mix up the word sequence and therefore it is able to interpret the features learnt for the current task to some extent.

This study investigates the usage of CNNs for a sequential labeling task like slot filling with the following contributions:

(1) We propose a novel CNN architecture for sequence labeling which takes into account the previous context words with preserved order information and pays special attention to the current word with its surrounding context.

(2) We extend the proposed CNN model to a bi-directional sequential CNN (bi-sCNN) which combines the information from past and future words for prediction.

(3) We compare the impact of two different ranking objective functions on the recognition performance and analyze the most important n-grams for semantic slot filling.

(4) On the ATIS benchmark dataset, the proposed bi-directional sequential CNN outperforms all RNN related models and defines a new start-of-the-art F1-score of 95.61%.

2. Related Work

Neural network models such as RNNs and CNNs have been used in a wide range of natural language processing tasks. Vanilla RNNs or their extensions such as LSTMs or GRUs showed their success in many different tasks such as language modeling [12] or machine translation [13]. Another trend is to use convolutional neural networks for sequence labeling [14, 15] or modeling larger units such as phrases [16] or sentences [17, 18]. For both models, distributed representations of words [19, 20] are used as input.

In the spoken language understanding research area, neural networks have also been applied to intent determination or semantic utterance classification tasks [21, 22]. For the slot filling task, RNNs [4, 5] and their extensions [7, 8] outperformed not only traditional approaches but also other neural network related models [6] and defined the state-of-the-art results on the ATIS benchmark data set. Recently it was shown in [9] that applying ranking loss to train the model is effective for tasks that involve an artificial class like O. They achieved state-of-the-art F1-scores of 95.47% with a single model and 95.56% by combining several models. In summary, the RNNs appear to be the best model for this task to date. The only previous study using convolutional neural networks was presented in [6] showing promising results. However, it did not outperform the RNN related models.

3. Bi-directional Sequential CNN

This section describes the architecture of the bi-directional sequential CNN (bi-sCNN) illustrated in Figure 1. It contains

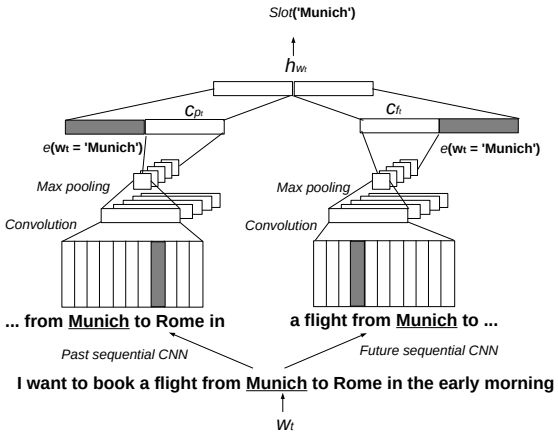


Figure 1: Bi-directional sequential CNN (bi-sCNN) which combines past and future sequential CNNs for slot filling

three main components: a vanilla sequential CNN, an extended surrounding context and a bi-directional extension.

3.1. Model

Vanilla sequential CNN. To predict the semantic slot of the current word w_t, we consider n previous words in combination with the current word. In order to avoid the border effect, the m future padding words are also included. Each of the words is embedded into an d-dimensional word embedding space. Thus for each current word, we form a matrix $w \in R^{(n+m+1)\times d}$ as an input to the CNN for prediction.

There are several possibilities for convolving the input matrix: applying 1D filters to each dimension independently or applying 2D filters spanning some or all dimensions of the word embeddings. In this paper, we use 2D filters f (with width $|f|$) spanning all embedding dimensions d. This is described by the following equation:

$$(w * f)(x, y) = \sum_{i=1}^{d} \sum_{j=-|f|/2}^{|f|/2} w(i, j) \cdot f(x - i, y - j) \quad (1)$$

where w is the word matrix and f is the filter matrix. On each output, a nonlinear function such as the sigmoid function can be applied. After convolution, we use a max pooling operation to find the most important features. This function stores only the highest activation of each convolutional filter for the succeeding steps. If s filter matrices are used, an s-dimensional feature representation vector c_{p_t} is created for further classification.

Extended surrounding context. When moving from one word to the next, the input matrix changes only slightly which leads to a large overlap of detected features from the convolutional and max pooling operator. Furthermore, the model needs to know which word is the current word for slot prediction. Therefore, in order to pay special attention to the current word and use the information of the word itself directly for the prediction, we introduce an additional component which uses the current word and its surrounding context words as input vector $e(w_t)$ with $d(2 * cs + 1)$ dimensions. cs is the surrounding context length. The feature representation of the current word is computed as follows:

$$h_{w_t} = f(U \cdot e(w_t) + V_p \cdot c_{p_t}) \quad (2)$$

where $U \in R^{s\times d(2*cs+1)}$ and $V_p \in R^{s\times s}$.

Bi-directional sequential CNN. As reported in [9], information not only from the past but also from the future contributes to the recognition accuracy. We therefore extend the sequential CNN to the future context. Because CNN preserves order information, we do not scan the input text from right to left like a bi-directional recurrent neural network. Instead, we take n future words in combination with the current word and the m previous padding words in the original order to form a matrix $w \in R^{(n+m+1)\times d}$ as an input to the future sequential CNN. Convolutional and max pooling operators are applied as in the vanilla sequential CNN to obtain a feature representation vector c_{f_t} for the future context information.

There are two different ways to combine the information from the past and future contexts. The combination can be achieved by a weighted sum of the forward and the backward hidden layer. This leads to the following hidden layer output at time step t:

$$h_{w_t} = f(U \cdot e(w_t) + V_p \cdot c_{p_t} + V_f \cdot c_{f_t}) \quad (3)$$

Another combination option is to concatenate the forward and the backward hidden layer.

$$h_{w_t} = [f(U \cdot e(w_t) + V_p \cdot c_{p_t}), f(U \cdot e(w_t) + V_f \cdot c_{f_t})] \quad (4)$$

The combined hidden layer output is then used to predict the semantic label for the current word. The experimental results in Section 4 show that the combination method is an important design choice that effects the final performance.

3.2. Training objective function

It was shown in [9] that using ranking loss is more accurate than cross entropy to train the model for this task. One reason might be that it does not force the network to learn a pattern for the ○ class which in fact may not exist. In this paper, we compare two different kinds of ranking loss functions.

The first function is the well known hinge loss function:

$$L = max(0, 1 - s_\theta(w_t)_{y^+} + s_\theta(w_t)_{c^-}) \quad (5)$$

with $s_\theta(w_t)_{y^+}$ and $s_\theta(w_t)_{c^-}$ as the scores for the target class and the wrongly predicted class of the model given the current word w respectively. This loss function maximizes the margin between those two classes.

The second one was proposed by Dos Santos et al. [23] and used in [9] to achieve the current best performance on the slot filling task till now. Instead of using the softmax activation function, we train a matrix W^{class} whose columns contain vector representations of the different classes. Therefore, the score for each class c can be computed by using the product

$$s_\theta(w_t)_c = h_{w_t}^T [W^{class}]_c \quad (6)$$

We use the same ranking loss function as in [9] to train the CNNs. It maximizes the distance between the true label y^+ and the best competitive label c^- given a data point x. The objective function is

$$\begin{aligned} L = &\log(1 + \exp(\gamma(m^+ - s_\theta(w_t)_{y^+}))) \\ &+ \log(1 + \exp(\gamma(m^- + s_\theta(w_t)_{c^-}))) \end{aligned} \quad (7)$$

with $s_\theta(w_t)_{y^+}$ and $s_\theta(w_t)_{c^-}$ as the scores for the classes y^+ and c^- respectively. The parameter γ controls the penalization of the prediction errors and m^+ and m^- are margins for

the correct and incorrect classes. γ, m^+ and m^- are hyper-parameters which can be tuned on the development set. For the class O, only the second summand of Equation 7 is calculated during training, i.e. the model does not learn a pattern for class O but nevertheless increases its difference to the best competitive label. Furthermore, it implicitly solves the problem of un-balanced data since the number of class O data points is much larger than in other classes. During testing, the model will predict class O if the scores for all other classes are < 0.

3.3. Comparison with other neural models

The information flow of the proposed model is comparable with a bi-directional RNN. Instead of using the recurrent architecture to save the information from a long context, we use a convolutional operator to scan all the n-grams in the contexts and find the most important features with max pooling. At every time step, the most important features are then learnt independently from the previous time step. This poses an advantage over bi-directional RNNs when the previous word is a word of class O and the current word is not of class O because the information to predict class O is not helpful to predict other classes. Another difference is the integration of future information. In the backward RNN model, the sentence is scanned from right to left which is against the nature of languages like English. In contrast, the CNN keeps the correct order of the sentence and searches for important n-grams.

Another interpretation of this model is a joint training of a feed-forward NN and a CNN. The feedforward NN takes the current word with its surrounding context as input for prediction while the CNN searches for n-gram features from the past and future contexts. The context representation of the CNN is used as additional input of the feedforward NN. This is an advantage of this model over the CNN model proposed in [15] which has problems identifying the current word for labeling.

4. Experimental Results

4.1. Data

To compare our work with previously studied methods, we report results on the widely used ATIS dataset [24, 25]. This dataset is from the air travel domain and consists of audio recordings of speakers making travel reservations. All the words are labeled with a semantic label in a BIO format (B: begin, I: inside, O: outside), e.g. *New York* contains two words *New* and *York* and is therefore labeled with `B-fromloc.city_name` and `I-fromloc.city_name` respectively. Words which do not have semantic labels are tagged with O. In total, the number of semantic labels is 127, including the label of the class O. The training data consists of 4,978 sentences and 56,590 words. The test set contains 893 sentences and 9,198 words. To evaluate our models, we used the script provided in the text chunking CoNLL shared task 2000[1] in line with other related work.

4.2. Model training

We used the Theano library [26] to implement the model. To train the model, stochastic gradient descent (SGD) was applied. We performed 5-fold cross-validation to tune the hyper-parameters. The learning rate was kept constant for the first 10 epochs. Afterwards, we halved the learning rate after each epoch and stopped the training after 25 epochs. Note that with more advanced techniques like AdaGrad [27] and AdaDelta [28] we did not achieve improvements over SGD with the described simple learning rate schedule. Since the learning schedule does not need a cross-validation set, we trained the final best model with the complete training data set. Table 1 shows the hyper-parameters used for all the CNN models.

[1] http://www.cnts.ua.ac.be/conll2000/chunking/

Table 1: *Hyper-parameters of sequential CNN*

Parameters	Value
activation function	sigmoid
number of features maps	100
features map window	(50, 5)
surrounding context	3
context length (past or future)	9
word embs	50
regularization	L2
L2 weight	1e-7
initial learning rate	0.02

4.3. Results

We adopted the window approach proposed in [15] as the baseline system. Five left context words, five right context words and the current word form the input of a feed-forward neural network with one hidden layer with size 100. We obtained an F1-score of 94.23% and 94.14% with this simple feed-forward network using ranking loss and hinge loss respectively. Table 2 summarizes the performance on the ATIS test set with different CNN architectural setups. The results show that the context information from the past is more important than the future context. The future context, however, appears to provide meaningful information because their combination leads to better results. Moreover, the comparison between two different kinds of combinations of previous and future context (concatenation vs. addition) suggests to not mix up the information using addition. Finally, results in Table 2 also reveal that using the ranking loss function proposed in [23] outperforms the hinge loss function.

Table 2: *F1-score (%) of uni vs. bi-directional sequential CNNs trained with two different ranking loss functions*

Objectives	Methods	Score
Hinge loss	Words with surrounding context = 5	94.14
Ranking loss	Words with surrounding context = 5	94.23
Hinge loss	Past sequential CNN	94.89
	Future sequential CNN	93.04
	Bi-directional sequential CNN (add)	94.78
	Bi-directional sequential CNN (concat)	94.98
Ranking loss	Past sequential CNN	95.31
	Future sequential CNN	93.59
	Bi-directional sequential CNN (add)	95.19
	Bi-directional sequential CNN (concat)	**95.61**

5. Analysis

We performed analyses regarding the choice of context length, the impact of including the current word with its surrounding context and the most important detected n-grams.

5.1. Context length

First, the impact of the context length on the final performance was explored. The number of parameters remained unchanged when reducing or increasing the context length. Short context means information loss while a long context length potentially adds noise to the input of the model. Table 3 shows that F1-scores increased when increasing the context length from 5 up to 9. Increasing the context length to 10 and 11, however, decreased the results slightly but the F1-scores stayed quite stable around 95.5%. This confirms our hypothesis that a longer context adds noise to the input while the model is still able to extract the important information for slot prediction.

Table 3: *Impact of the context length on the F1-score (%)*

Context length	5	7	9	10	11
F1-score	94.19	95.17	95.61	95.42	95.51

5.2. Surrounding context

Table 4 summarizes the F1-score without using the current word or with the current context with various lengths of the surrounding contexts. The results revealed the strong impact of including the current word with its surrounding context into the CNN on the final F1-score. Without paying attention to the current word, the F1-score dropped significantly to 92.01%. Successively adding the current word and increasing its surrounding contexts up to three left and three right neighbour words resulted in better performance. Increasing the surrounding context to four, however, decreased F1-score. The best F1-score was obtained with three left and three right neighbour words.

Table 4: *Impact of including the current word with surrounding context into the CNN on the F1-score (%)*

Methods	F1-score
Bi-directional sequential CNN (concat) - current word	92.01
+ current word w/o context	95.09
+ surrounding context = 1	95.21
+ surrounding context = 2	95.37
+ surrounding context = 3	95.61
+ surrounding context = 4	95.41

5.3. Most important n-grams

We analyzed the most significant patterns for the four most frequent semantic slots in the test data. For each of them, we present up to three n-grams which contributed the most to scoring the correctly classified test data points. To compute the most important n-grams, we first detected the position of the maximum contribution to the dot product and traced it back to the corresponding feature map. Based on the max pooling, we were able to trace back and identify the n-grams which were used. To create the results presented in Table 5, we ranked the n-grams which were selected as the most important features in all the sentences based on frequency and picked the most frequent ones. Table 5 shows that the model has learnt something meaningful for this task. For example, a pattern such as *flights from A to B* was used to predict `fromloc.city_name` while the model only used *A to B* or *to B* for `toloc.city_name` prediction. Other examples are patterns such as *afternoon, evening* and *night* which appeared quite frequently after `depart_date.day_name` and therefore are learnt as indicators.

Table 5: *Most important n-grams for slot prediction*

Slots	n-grams
fromloc.city_name	*'flights from washington dc to'* *'flights from ontario california to'* *'from toronto to san diego'*
toloc.city_name	*'toronto to san diego'* *'st. louis to burbank'*
depart_date.day_name	*'afternoon sentence_end'* *'evening sentence_end'* *'night sentence_end'*
airline_name	*'northwest us air and united'* *'show delta airlines flights from'*

6. Comparison with state of the art

Table 6 lists several previous results on the ATIS data set including our best results. The proposed R-bi-sCNN outperforms

Table 6: *Comparison with state-of-the-art results*

Methods	F1-score
CRF [5]	92.94
simple RNN [4]	94.11
CNN [6]	94.35
LSTM [7]	94.85
RNN-EM [8]	95.25
R-bi-RNN [9]	95.47
R-bi-sCNN	**95.61**

the previously best ranking bi-directional RNN (R-bi-RNN). A more detailed comparison with R-bi-RNN shows that R-bi-sCNN performed as well as R-bi-RNN on the frequent semantic slots but outperformed R-bi-RNN on the rare slots. For example, rare slots such as `toloc.country_name`, `days_code`, `period_of_day`, which appeared less than six times in the training data, were correctly predicted with the R-bi-sCNN model but not with R-bi-RNN .

7. Conclusions

This paper explored convolutional neural networks for the slot filling task in spoken language understanding. Our novel CNN architecture - bi-directional sequential CNN - takes into account the information from the past and the future with preserved order information and pays special attention to the current word with its surrounding contexts. To train the model, we compared two different ranking objective functions. Our findings revealed that not forcing the model to learn a pattern for `O` class is helpful to improve the final performance. Finally, our bi-directional sequential CNN achieves state-of-the-art results with an F1-score of 95.61% on the ATIS benchmark dataset without using any additional linguistic knowledge and resources. As future work, we aim to evaluate the proposed model on other datasets (e.g. data presented in [29, 30]).

8. Acknowledgements

This work was funded by the German Science Foundation (DFG), Sonderforschungsbereich 732 *Incremental Specification in Context*, Project A8, at the University of Stuttgart.

9. References

[1] Y. Wang, L. Deng, and A. Acero. Spoken Language Understanding An Introduction to the Statistical Framework, IEEE Signal Processing Magazine, vol. 22, no. 5, pp. 16-31, 2005.

[2] J. Lafferty, A. McCallum, and F. P ereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in Proc. of ICML, 2001.

[3] Y. Wang, L. Deng, and A. AceroSemantic Frame Based Spoken Language Understanding, in Chapter 3, Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, pp. 35-80, Wiley, 2011.

[4] K. Yao, G. Zweig, M. Hwang, Y. Shi, and D. Yu, Recurrent neural networks for language understanding, in Proc. of Interspeech, 2013.

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

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

[7] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, Spoken language understanding using long short-term memory neural networks, in Proc. of SLT, 2014.

[8] B. Peng, K. Yao. Recurrent Neural Networks with External Memory for Language Understanding, in arXiv, 2015.

[9] N.T. Vu, P. Gupta, H. Adel and H. Schuetze. Bi-directional Recurrent Neural Network with Ranking Loss for Spoken Language Understanding, in Proc. of ICASSP, 2016.

[10] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

[11] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory, Neural Computation, 9(8):1735?1780, 1997.

[12] T. Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, Extensions of recurrent neural network based language model, in Proc. of ICASSP, 2011.

[13] K. Cho, B. van Merrienboer, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio Learning phrase representations using RNN encoder-decoder for statistical machine translation, in Proc. of EMNLP, 2014.

[14] R. Collobert and J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in Proc. of ICML, 2008.

[15] 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, 2011.

[16] Y. Wenpeng, and H. Schtze. MultiGranCNN: An Architecture for General Matching of Text Chunks on Multiple Levels of Granularity, in Proc. of ACL, 2015.

[17] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188, 2014.

[18] Y. Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[19] Y. Bengio, R. Ducharme and P. Vincent, A Neural Probabilistic Language Model, in Proc. of NIPS, 2000.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space, in Proc. of Workshop at ICLR, 2013.

[21] 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 Proc. of SLT, 2012.

[22] G. Tur, L. Deng, D. Hakkani-Tur, and X. He, Towards Deeper Understanding Deep Convex Networks for Semantic Utterance Classification, in Proc. of ICASSP, 2012.

[23] C.N. Dos Santos, B. Xiang, and B. Zhou. Classifying relations by ranking with convolutional neural networks, in Proc. of ACL, 2015.

[24] C. Hemphill, J. Godfrey, and G. Doddington, The ATIS spoken language systems pilot corpus, in Proc. of the DARPA speech and natural language workshop, 1990.

[25] P. Price, Evaluation of spoken language systems: The ATIS domain, in Proc. of the Third DARPA Speech and Natural Language Workshop. Morgan Kaufmann, 1990.

[26] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I.J. Goodfellow, A. Bergeron, N. Bouchard, Y. and Bengio, Y. Theano: new features and speed improvements, in Proc. of Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2012.

[27] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research, vol. 12, pp. 2121-2159, 2010.

[28] M.D. Zeiler. ADADELTA: An Adaptive Learning Rate Method, CoRR, abs/1212.5701, 2012.

[29] G. Tur, D. Hakkani-Tur, L. Heck. What is left to be understood in ATIS?, in Proc. of SLT, 2010.

[30] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, R.D. Mori, A. Moschitti, H. Ney, G. Riccardi. Comparing stochastic approaches to spoken language understanding in multiple languages, in IEEE Transactions on Audio, Speech, and Language Processing, pp. 1569-1583, 2011.

Generative Encoder-Decoder Models for Task-Oriented Spoken Dialog Systems with Chatting Capability

Tiancheng Zhao, Allen Lu, Kyusong Lee and Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
{tianchez,arlu,kyusongl,max+}@cs.cmu.edu

Abstract

Generative encoder-decoder models offer great promise in developing domain-general dialog systems. However, they have mainly been applied to open-domain conversations. This paper presents a practical and novel framework for building task-oriented dialog systems based on encoder-decoder models. This framework enables encoder-decoder models to accomplish slot-value independent decision-making and interact with external databases. Moreover, this paper shows the flexibility of the proposed method by interleaving chatting capability with a slot-filling system for better out-of-domain recovery. The models were trained on both real-user data from a bus information system and human-human chat data. Results show that the proposed framework achieves good performance in both offline evaluation metrics and in task success rate with human users.

1 Introduction

Task-oriented spoken dialog systems have transformed human-computer interaction by enabling people interact with computers via spoken language (Raux et al., 2005; Young, 2006; Bohus and Rudnicky, 2003). The task-oriented SDS is usually domain-specific. The system creators first map the user utterances into semantic frames that contain domain-specific slots and intents using spoken language understanding (SLU) (De Mori et al., 2008). Then a set of domain-specific dialog state variables is tracked to retain the context information over turns (Williams et al., 2013). Lastly, the dialog policy decides the next move from a list of dialog acts that covers the expected communicative functions from the system.

Although the above approach has been successfully applied to many practical systems, it has limited ability to generalize to out-of-domain (OOD) requests and to scale up to new domains. For example, even within in a simple domain, real users often make requests that are not included in the semantic specifications. Due to this, proper error handling strategies that guide users back to the in-domain conversation are crucial to dialog success (Bohus and Rudnicky, 2005). Past error handling strategies were limited to a set of predefined dialog acts, e.g. request repeat, clarification etc., which constrained the system's capability in keeping users engaged. Moreover, there has been an increased interest in extending task-oriented systems to multiple topics (Lee et al., 2009; Gašić et al., 2015b) and multiple skills, e.g. grouping heterogeneous types of dialogs into a single system (Zhao et al., 2016). Both cases require the system to be flexible enough to extend to new slots and actions.

Our goal is to move towards a domain-general task-oriented SDS framework that is flexible enough to expand to new domains and skills by removing domain-specific assumptions on the dialog state and dialog acts (Bordes and Weston, 2016). To achieve this goal, the neural encoder-decoder model(Cho et al., 2014; Sutskever et al., 2014) is a suitable choice, since it has achieved promising results in modeling open-domain conversations (Vinyals and Le, 2015; Sordoni et al., 2015). It encodes the dialog history using deep neural networks and then generates the next system utterance word-by-word via recurrent neural networks (RNNs). Therefore, unlike the traditional SDS pipeline, the encoder-decoder model is theoretically only limited by its input/output vocabulary.

A naïve implementation of an encoder-decoder-based task-oriented system would use RNNs to encode the raw dialog history and generate the next system utterance using a separate RNN decoder. However, while this implementation might achieve good performance in an offline evaluation of a closed dataset, it would certainly fail when used by humans. There are several reasons for this: 1) real users can mention new entities that do not appear in the training data, such as a new restaurant name. These entities are, however, essential in delivering the information that matches users' needs in a task-oriented system. 2) a task-oriented SDS obtains information from a knowledge base (KB) that is constantly updated ("today's" weather will be different every day), so simply memorizing KB results that occurred in the training data would produce false information. Instead, an effective model should learn to query the KB constantly to get the most up-to-date information. 3) users may give OOD requests (e.g. say, "how is your day", to a slot-filling system), which must be handled gracefully in order to keep the conversation moving in the intended direction.

This paper proposes an effective encoder-decoder framework for building task-oriented SDSs. We propose *entity indexing* to tackle the challenges of out-of-vocabulary (OOV) entities and to query the KB. Moreover, we show the extensibility of the proposed model by adding chatting capability to a task-oriented encoder-decoder SDS for better OOD recovery. This approach was assessed on the Let's Go Bus Information data from the 1st Dialog State Tracking Challenge (Williams et al., 2013), and we report performance on both offline metrics and real human users. Results show that this model attains good performance for both of these metrics.

2 Related Work

Past research in developing domain-general dialog systems can be broadly divided into three branches. The first one focuses on learning domain-independent dialog state representation while still using hand-crafted dialog act system actions. Researchers proposed the idea of extracting slot-value independent statistics as the dialog state (Wang et al., 2015; Gašić et al., 2015a), so that the dialog state representation can be shared across systems serving different knowledge sources. Another approach uses RNNs to automatically learn a distributed vector representation of the dialog state by accumulating the observations at each turn (Williams and Zweig, 2016; Zhao and Eskenazi, 2016; Dhingra et al., 2016; Williams et al., 2017). The learned dialog state is then used by the dialog policy to select the next action. The second branch of research develops a domain-general action space for dialog policy. Prior work replaced the domain-specific dialog acts with domain-independent natural language semantic schema as the action space of dialog managers (Eshghi and Lemon, 2014), e.g. Dynamic Syntax (Kempson et al., 2000). More recently, Wen, et al. (2016) have shown the feasibility of using an RNN as the decoder to generate the system utterances word by word, and the dialog policy of the proposed model can be fine tuned using reinforcement learning (Su et al., 2016). Furthermore, to deal with the challenge of developing end-to-end task-oriented dialog models that are able to interface with external KB, prior work has unified the special KB query actions via deep reinforcement learning (Zhao and Eskenazi, 2016) and soft attention over the database (Dhingra et al., 2016). The third branch strives to solve both problems at the same time by building an end-to-end model that maps an observable dialog history directly to the word sequences of the system's response. By using an encoder-decoder model, it has been successfully applied to open-domain conversational models (Serban et al., 2015; Li et al., 2015, 2016; Zhao et al., 2017), as well as to task oriented systems (Bordes and Weston, 2016; Yang et al., 2016; Eric and Manning, 2017). In order to better predict the next correct system action, this branch has focused on investigating various neural network architectures to improve the machine's ability to reason over user input and model long-term dialog context.

This paper is closely related to the third branch, but differs in the following ways: 1) these models are slot-value independent by leveraging domain-general entity recognizer, which is more extensible to OOV entities, 2) these models emphasize the interactive nature of dialog and address out-of-domain handling by interleaving chatting in task-oriented conversations, 3) instead of testing on a synthetic dataset, this approach focuses on real world use by testing the system on human users via spoken interface.

3 Proposed Method

Our proposed framework consists of three steps as shown in Figure 2: a) entity indexing (EI), b) slot-value independent encoder-decoder (SiED), c) system utterance lexicalization (UL). The intuition is to leverage domain-general named entity recognition (NER) (Tjong Kim Sang and De Meulder, 2003) techniques to extract salient entities in the raw dialog history and convert the lexical values of the entities into entity indexes. The encoder-decoder model is then trained to focus solely on reasoning over the entity indexes in a dialog history and to make decisions about the next utterance to produce (including KB query). In this way, the model can be unaffected by the inclusion of new entities and new KB, while maintaining its domain-general input/output interface for easy extension to new types of conversation skills. Lastly, the output from the decoder networks are lexicalized by replacing the entity indexes and special KB tokens with natural language. The following sections explain each step in detail.

3.1 Entity Indexing and Utterance Lexicalization

Entity Indexing EI has two parts. First, the EI utilizes an existing domain-general NER to extract entities from both the user and system utterances. Note that the entity here is assumed to be a super-set of the slots in the domain. For example, for a flight-booking system, the system may contain two slots: [from-LOCATION] and [to-LOCATION] for the departure and arrival city, respectively. However, EI only extracts every mention of [LOCATION] in the utterances and leaves the task of distinguishing between departure and arrival to the encoder-decoder model. Furthermore, this step replaces each KB search result with its search query (e.g. the weather is cloudy → [kb-search]-[DATETIME-0]). The second step of EI involves constructing a *indexed entity table*. Each entity is indexed by its order of occurrence in the conversation. Figure 1 shows an example in which there are two [LOCATION] mentions.

Properties of Entity Indexing In this section, several properties of EI and their assumptions are addressed. First, each entity is indexed uniquely by its entity type and index. Note that the index is not associated with the entity value, but rather solely by the order of appearance in the dialog. Despite the actual words being hidden,

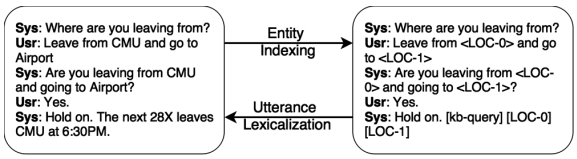


Figure 1: An example of entity indexing and utterance lexicalization.

a human can still easily predict which entity the system should confirm or search for in the KB based on logical reasoning. Therefore, that the EI not only alleviates the OOV problem of deploying the encoder-decoder model in the real world, but also forces the encoder-decoder model's focus on learning the reasoning process of task-oriented dialogs instead of leveraging too much information from the language modeling.

Moreover, most slot-filling SDSs, apart from informing the concepts from KBs, usually do not introduce novel entities to users. Instead, systems mostly corroborate the entities introduced by the users. With this assumption, every entity mention in the system utterances can always be found in the users' utterances in the dialog history, and therefore can also be found in the indexed entity table. This property reduces the grounding behavior of the conventional task-oriented dialog manager into selecting an entity from the indexed entity table and confirming it with the user.

Utterance Lexicalization is the reverse of EI. Since EI is a deterministic process, its effect can always be reversed by finding the corresponding entity in the indexed entity table and replacing the index with its word. For KB search, a simple string matching algorithm can search for the special [kb-search] token and take the following generated entities as the argument to the KB. Then the actual KB results can replace the original KB query. Figure 1 shows an example of utterance lexicalization.

3.2 Encoder-Decoder Models

The encoder-decoder model can then read in the EI-processed dialog history and predict the system's next utterance in EI format. Specifically, a dialog history of k turns is represented by $[(a_0, u_0, c_0), ...(a_{k-1}, u_{k-1}, c_{k-1})]$, in which a_i, u_i and c_i are, respectively, the system, user utterance and ASR confidence score at turn i. Each utterance in the dialog history is encoded into fixed-size vectors using Convolutional Neural Networks

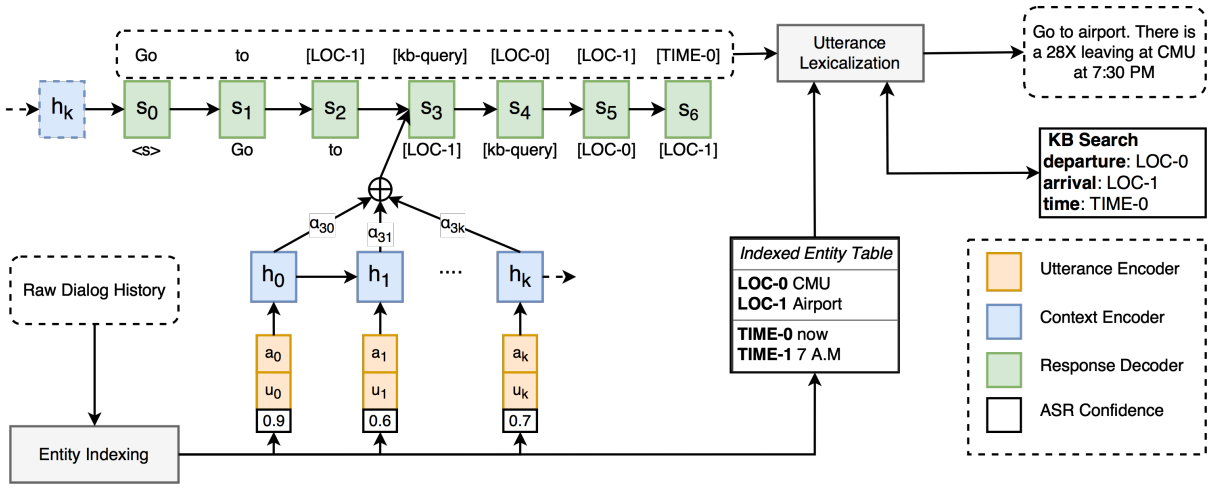


Figure 2: The proposed pipeline for task-oriented dialog systems.

(CNNs) proposed in (Kim, 2014). Specifically, each word in an utterance x is mapped to its word embedding, so that an utterance is represented as a matrix $R \in R^{|x| \times D}$, in which D is the size of the word embedding. Then L filters of size 1,2,3 conduct convolutions on R to obtain a feature map, c, of n-gram features in window size 1,2,3. Then c is passed through a nonlinear ReLu (Glorot et al., 2011) layer, followed by a max-pooling layer to obtain a compact summary of salient n-gram features, i.e. $e^t(x) = \text{maxpool}(\text{ReLu}(c + b))$. Using CNNs to capture word-order information is crucial, because the encoder-decoder has to be able to distinguish between fine-grained differences between entities. For example, a simple bag-of-word embedding approach will fail to distinguish between the two location entities in "leave from [LOCATION-0] and go to [LOCATION-1]", while a CNN encoder can capture the context information of these two entities.

After obtaining utterance embedding, a turn-level dialog history encoder network similar to the one proposed in (Zhao and Eskenazi, 2016) is used. Turn embedding is a simple concatenation of system, user utterance embedding and the confidence score $t = [e^u(a_i); e^u(u_i); c_i]$. Then an Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network reads the sequence turn embeddings in the dialog history via recursive state update $s_{i+1} = \text{LSTM}(t_{i+1}, h_i)$, in which h_i is the output of the LSTM hidden state.

Decoding with/without Attention A vanilla decoder takes in the last hidden state of the encoder as its initial state and decodes the next system utterance word by word as shown in (Sutskever et al., 2014). This assumes that the fixed-size hidden state is expressive enough to encode all important information about the history of a dialog. However, this assumption may often be violated for a task that has long-dependency or complex reasoning of the entire source sequence. An attention mechanism proposed (Bahdanau et al., 2014) in the machine translation community has helped encoder-decoder models improve state-of-art performance in various tasks (Bahdanau et al., 2014; Xu et al., 2015). Attention allows the decoder to look over every hidden state in the encoder and dynamically decide the importance of each hidden state at each decoding step, which significantly improves the model's ability to handle long-term dependency. We experiment decoders both with and without attention. Attention is computed similarly multiplicative attention described in (Luong et al., 2015). We denote the hidden state of the decoder at time step j by s_j, and the hidden state outputs of the encoder at turn i by h_i. We then predict the next word by

$$a_{ji} = \text{softmax}(h_i^T W_a s_j + b_a) \quad (1)$$

$$c_j = \sum_i a_{ji} h_i \quad (2)$$

$$\widetilde{s_j} = tanh(W_s \begin{bmatrix} s_j \\ c_j \end{bmatrix}) \quad (3)$$

$$p(w_j | s_j, c_j) = \text{softmax}(W_o \widetilde{s_j}) \quad (4)$$

The decoder next state is updated by $s_{j+1} = \text{LSTM}(s_j, e(w_{j+1}), \widetilde{s_j})$.

3.3 Leveraging Chat Data to Improve OOD Recovery

Past work has shown that simple supervised learning is usually inadequate for learning robust sequential decision-making policy (Williams and Young, 2003; Ross et al., 2011). This is because the model is only exposed to the expert demonstration, but not to examples of how to recover from its own mistakes or users' OOD requests. We present a simple yet effective technique that leverages the extensibility of the encoder-decoder model in order to obtain a more robust policy in the setting of supervised learning. Specifically, we artificially augment a task-oriented dialog dataset with chat data from an open-domain conversation corpus. This has been shown to be effective in improving the performance of task-oriented systems (Yu et al., 2017). Let the original dialog dataset with N dialogs be $\mathcal{D} = [d_0..., d_n, ...d_N]$, where d_n is a multi-turn task-oriented dialog of $|d_n|$ turns. Furthermore, we assume we have access to a chat dataset $\mathcal{D}_c = [(q_0, r_0), ...(q_m, r_m), ...(q_M, r_M)]$, where q_m, r_m are common adjacency pairs that appear in chats, (e.g. q = hello, r = hi, how are you). Then we can create a new dataset $\mathcal{D}^*$ by repeating the following process a certain number of times:

1. Randomly sample dialog d_n from $\mathcal{D}$
2. Randomly sample turn $t_i = [a_i, u_i]$ from d_n
3. Randomly sample an adjacency pair (q_m, r_m) from $\mathcal{D}_c$
4. Replace the user utterance of t_i by q_m so that $t_i = [a_i, q_m]$
5. Insert a new turn after t_i, i.e. $t_{i+1} = [r_m + e_{i+1}, u_i]$

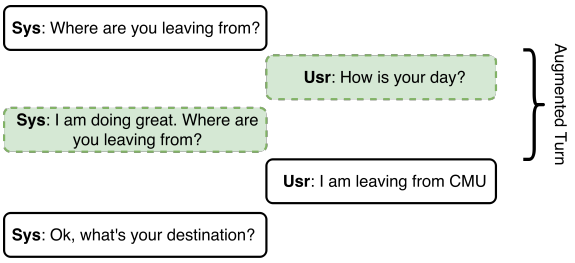


Figure 3: Illustration of data augmentation. The turn in the dashed line is inserted in the original dialog.

In Step 5, e_i is an error handling system utterance after the system answers the user's OOD request, q_m. In this study, we experimented with a simple case where $e_{i+1} = a_i$ so that the system should repeat its previous prompt after responding to q_m via r_m. Figure 3 shows an example of an augmented turn. Eventually, we train the model on the union of the two datasets $\mathcal{D}^+ = \mathcal{D} \cup \mathcal{D}^*$

Discussion: There are several reasons that the above data augmentation process is appealing. First, the model effectively learns an OOD recovery strategy from $\mathcal{D}^*$, i.e. it first gives chatting answers to users' OOD requests and then tries to pull users back to the main-task conversation. Second, chat data usually has a larger vocabulary and more diverse natural language expressions, which can reduce the chance of OOVs and enable the model to learn more robust word embeddings and language models.

4 Experiment Setup

4.1 Dataset and Domain

The CMU Let's Go Bus Information System (Raux et al., 2005) is a task-oriented spoken dialog system that contains bus information. We combined the train1a and train1b datasets from DSTC 1 (Williams et al., 2013), which contain 2608 total dialogs. The average dialog length is 9.07 turns. The dialogs were randomly splitted into 85/5/10 proportions for train/dev/test data. The data was noisy since the dialogs were collected from real users via telephone lines. Furthermore, this version of Let's Go used an in-house database containing the Port Authority bus schedule. In the current version, that database was replaced with the Google Directions API, which both reduces the human burden of maintaining a database and opens the possibility of extending Let's Go to cities other than Pittsburgh. Connecting to Google Directions API involves a POST call to their URL, with our given access key as well as the parameters needed: departure place, arrival place and departure time, and the travel mode, which we always set as TRANSIT to obtain relevant bus routes. There are 14 distinct dialog acts available to the system, and each system utterance contains one or more dialog acts. Lastly, the system vocabulary size is 1311 and the user vocabulary size is 1232. After the EI process, the sizes become 214 and 936, respectively.

For chat data, we use a publicly available chat

corpus used in (Yu et al., 2015)[1]. In total, there are 3793 chatting adjacency pairs. We control the number of data injections to 30% of the number of turns in the original DTSC dataset, which leads to a user vocabulary size of 3537 and system vocabulary size of 4047.

4.2 Training Details

For all experiments, the word embedding size was 100. The sizes of the LSTM hidden states for both the encoder and decoder were 500 with 1 layer. The attention context size was also 500. We tied the CNN weights for the encoding system and user utterances. Each CNN has 3 filter windows, 1, 2, and 3, with 100 feature maps each. We trained the model end-to-end using Adam (Kingma and Ba, 2014), with a learning rate of 1e-3 and a batch size of 40. To combat overfitting, we apply dropout (Zaremba et al., 2014) to the LSTM layer outputs and the CNN outputs after the maxpooling layer, with a dropout rate of 40%.

5 Experiments Results

This approach was assessed both offline and online evaluations. The offline evaluation contains standard metrics to test open-domain encoder-decoder dialog models (Li et al., 2015; Serban et al., 2015). System performance was assessed from three perspectives that are essential for task-oriented systems: dialog acts, slot-values, and KB query. The online evaluation is composed of objective task success rate, the number of turns, and subjective satisfaction with human users.

5.1 Offline Evaluation

Dialog Acts (DA): Each system utterance is made up of one or more dialog acts, e.g. "leaving at [TIME-0], where do you want to go?" → [implicit-confirm, request(arrival place)]. To evaluate whether a generated utterance has the same dialog acts as the ground truth, we trained a multi-label dialog tagger using one-vs-rest Support Vector Machines (SVM) (Tsoumakas and Katakis, 2006), with bag-of-bigram features for each dialog act label. Since the natural language generation module in Let's Go is handcrafted, the dialog act tagger achieved 99.4% average label accuracy on a held-out dataset. We used this dialog act tagger to tag both the ground truth and the generated responses. Then we computed the micro-average precision, recall, and the F-score.

Slots: This metric measures the model's performance in generating the correct slot-values. The slot-values mostly occur in grounding utterances (e.g. explicit/implicit confirm) and KB queries. We compute precision, recall, and F-score.

KB Queries: Although the slots metric already covers the KB queries, here the precision/recall/F-score of system utterances that contain KB queries are also explicitly measured, due to their importance. Specifically, this action measures whether the system is able to generate the special [kb-query] symbol to initiate a KB query, as well as how accurate the corresponding KB query arguments are.

BLEU (Papineni et al., 2002): compares the n-gram precision with length penalty, and has been a popular score used to evaluate the performance of natural language generation (Wen et al., 2015) and open-domain dialog models (Li et al., 2016). Corpus-level BLEU-4 is reported.

Metrics	Vanilla	EI	EI +Attn	EI+Attn +Chat
DA	83.5	79.7	80.0	81.8
(p/r/f1)	77.9	80.1	83.1	83.5
	80.5	80.0	81.5	82.7
Slot	42.0	60.6	63.7	64.6
(p/r/f1)	30.3	63.6	64.7	69.1
	35.2	62.1	64.2	66.8
KB	N/A	48.9	55.4	58.2
(p/r/f1)		55.3	70.8	71.9
		51.9	62.2	64.4
BLEU	36.9	54.6	59.3	60.5

Table 1: Performance of each model on automatic measures.

Four systems were compared: the basic encoder-decoder models without EI (vanilla), the basic model with EI pre-processing (EI), the model with attentional decoder (EI+Attn) and the model trained on the dataset augmented with chatting data (EI+Attn+Chat). The comparison was carried out on exactly the same held-out test dataset that contains 261 dialogs. Table 1 shows the results. It can be seen that all four models achieve similar performance on the dialog act metrics, even the vanilla model. This confirms the capacity of encoder-decoders models to learn the "shape" of a conversation, since they have

[1] github.com/echoyuzhou/ticktock_text_api

achieved impressive results in more challenging settings, e.g. modeling open-domain conversations. Furthermore, since the DSTC1 data was collected over several months, there were minor updates made to the dialog manager. Therefore, there are inherent ambiguities in the data (the dialog manager may take different actions in the same situation). We conjecture that ∼80% is near the upper limit of our data in modeling the system's next dialog act given the dialog history.

On the other hand, these proposed methods significantly improved the metrics related to slots and KB queries. The inclusion of EI alone was able to improve the F-score of slots by a relative 76%, which confirms that EI is crucial in developing slot-value independent encoder-decoder models for modeling task-oriented dialogs. Likewise, the inclusion of attention further improved the prediction of slots in system utterances. Adding attention also improved the performance of predicting KB queries, more so than the overall slot accuracy. This is expected, since KB queries are usually issued near the end of a conversation, which requires global reasoning over the entire dialog history. The use of attention allows the decoder to look over the history and make better decisions rather than simply depending on the context summary in the last hidden layer of the encoder. Because of the good performance achieved by the models with the attentional decoder, the attention weights in Equation 1 at every step of the decoding process in two example dialogs from test data are visualized. For both figures, the vertical axes show the dialog history flowing from the top to the bottom. Each row is a turn in the format of (system utterance # user utterance). The top horizontal axis shows the predicted next system utterance. The darkness of a bar indicates the value of the attention calculated in Equation 1.

The first example shows attention for grounding the new entity [LOCATION-1] in the previous turn. The attention weights become focus on the previous turn when predicting [LOCATION-1] in the implicit confirm action. The second dialog example shows a more challenging situation, in which the model is predicting a KB query. We can see that the attention weights when generating each input argument of the KB query clearly focus on the specific mention in the dialog history. The visualization confirms the effectiveness of the attention mechanism in dealing with long-term dependency at discourse level.

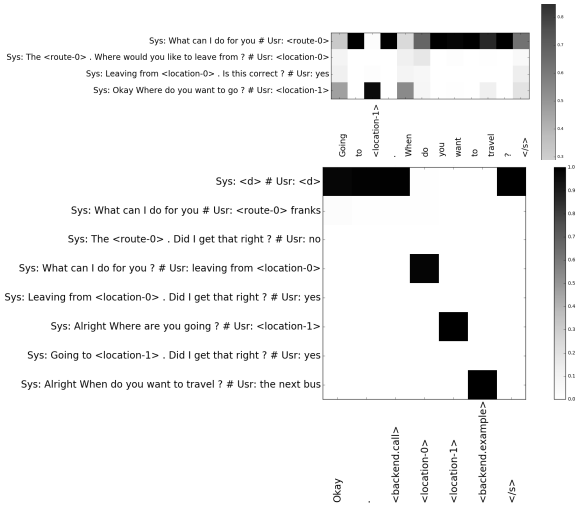


Figure 4: Visualization of attention weights when generating implicit confirm (top) and KB query (bottom).

Surprisingly, the model trained on the data augmented with chat achieved slightly better slot accuracy performance, even though the augmented data is not directly related to task-oriented dialogs. Furthermore, the model trained on chat-augmented data achieved better scores for the KB query metrics. Several reasons may explain this improvement: 1) since chat data exposes the model to a significantly larger vocabulary, the resulting model is more robust to words that it had not seen in the original task-oriented-only training data, and 2) the augmented dialog turn can be seen as noise in the dialog history, which adds extra regularization to the model and enables the model to learn more robust long-term reasoning mechanisms.

5.2 Human Evaluation

Although the model achieves good performance in offline evaluation, this may not carray over to real user dialogs, where the system must simultaneously deal with several challenges, such as automatic speech recognition (ASR) errors, OOD requests, etc. Therefore, a real user study was conducted to evaluate the performance of the proposed systems in the real world. Due to the limited number of real users, only two best performing system were compared, EI+Attn and EI+Attn+Chat. Users were able to talk to a web interface to the dialog systems via speech. Google

Chrome Speech API [2] served as the ASR and text-to-speech (TTS) modules. Turn-taking was done via the built-in Chrome voice activity detection (VAD) plus a finite state machine-based end-of-turn detector (Zhao et al., 2015). Lastly, a hybrid named entity recognizer (NER) was trained using Conditional Random Field (CRF) (McCallum and Li, 2003) and rules to extract 4 types of entities (location, hour, minute, pm/am) for the EI process.

The experiment setup is as follows: when a user logs into the website, the system prompts the user with a goal, which is a randomly chosen combination of departure place, arrival place and time (*e.g. leave from CMU and go to the airport at 10:30 AM*). The system also instructs the user to say goodbye if the he/she thinks the goal is achieved or wants to give up. The user begins a conversation with one of the two evaluated systems, with a 50/50 chance of choosing either system (not visible to the user). After the user's session is finished, the system asks the him/her to give two scores between 1 and 5 for correctness and naturalness of the system respectively. The subjects in this study consist of undergraduate and graduate students. However, many subjects did not follow the prompted goal, but rather asked about bus routes of their own. Therefore, the dialog was manually labeled for dialog success. A dialog is successful if and only if the systems give at least one bus schedule that matches with all three slots expressed by the users. Table 2 shows the

Metrics	EI+Attn	EI+Attn +Chat
# of Dialog	75	74
Slot Precision	73.3%	71.8%
KB Precision	88.6%	93.7%
Success Rate	73.3%	77.0%
Avg Turns	4.88	4.91
Avg Correctness	3.45 (1.32)	3.22 (1.40)
Avg Naturalness	3.46 (1.41)	3.53 (1.34)

Table 2: Performance of each model on automatic measures. The standard deviations of subjective scores are in parentheses.

results. Overall, our systems achieved reasonable performance in terms of dialog success rate. The EI+Attn+Chat model achieves slightly higher success and subjective naturalness metrics (although the difference between EI+Attn+Chat and EI+Attn was not statistically significant due to the limited number of subjects). The precision of grounding the correct slots and predicting the correct KB query was also manually labelled. EI+Attn model performs slightly better than the EI+Attn+Chat model in slot precision, while the latter model performs significantly better in KB query precision. In addition, EI+Attn+Chat leads to slightly longer dialogs because sometimes it generates chatting utterances with users when it cannot understand users' utterances.

At last, we investigated the log files and identified the following major types of sources of dialog failure: **RNN Decoder Invalid Output:** Occasionally, the RNN decoder outputs system utterances as "Okay going to [LOCATION-2]. Did I get that right?", in which [LOCATION-2] cannot be found in the indexed entity table. Such invalid output confuses users. This occurred in 149 of the dialogs, where 4.1% of system utterances contain invalid symbols. **Imitation of Suboptimal Dialog Policy:** Since our models are only trained to imitate the suboptimal hand-crafted dialog policy, their limitations show when the original dialog manager cannot handle the situation, such as failing to understand slots that appeared in compound utterances. Future plans involves improving the models to perform better than the suboptimal teacher policy.

6 Conclusions

In conclusion, this paper discusses constructing task-oriented dialog systems using generative encoder decoder models. EI is effective in solving both the OOV entity and KB query challenges for encoder-decoder-based task-oriented SDSs. Additionally, the novel data augmentation technique of interleaving task-oriented dialog corpus with chat data led to better model performance in both online and offline evaluation. Future work includes developing more advanced encoder-decoder models that to better deal with long-term dialog history and complex reasoning challenges than current models do. Furthermore, inspired by the success of mixing chatting with slot-filling dialogs, we will take full advantage of the extensibility of encoder-decoder models by investigating how to make systems that are able to interleave various conversational tasks, e.g. different domains, chatting or task-oriented, which in turn can create a more versatile conversational agent.

[2] www.google.com/intl/en/chrome/demos/speech.html

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Dan Bohus and Alexander I Rudnicky. 2003. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda .

Dan Bohus and Alexander I Rudnicky. 2005. Error handling in the ravenclaw dialog management framework. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 225–232.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. 2008. Spoken language understanding. *IEEE Signal Processing Magazine* 25(3).

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777* .

Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024* .

Arash Eshghi and Oliver Lemon. 2014. How domain-general can we be? learning incremental dialogue systems without dialogue acts. *DialWattSemdial 2014* page 53.

M Gašić, N Mrkšić, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015a. Policy committee for adaptation in multi-domain spoken dialogue systems. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, pages 806–812.

Milica Gašić, Dongho Kim, Pirros Tsiakoulis, and Steve Young. 2015b. Distributed dialogue policies for multi-domain statistical dialogue management. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pages 5371–5375.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Aistats*. volume 15, page 275.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Ruth Kempson, Wilfried Meyer-Viol, and Dov M Gabbay. 2000. *Dynamic syntax: The flow of language understanding*. Wiley-Blackwell.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Cheongjae Lee, Sangkeun Jung, Seokhwan Kim, and Gary Geunbae Lee. 2009. Example-based dialog modeling for practical multi-domain dialog system. *Speech Communication* 51(5):466–484.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 188–191.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Lets go public! taking a spoken dialog system to the real world. In *in Proc. of Interspeech 2005*. Citeseer.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*. volume 1, page 6.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Zhuoran Wang, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou. 2015. Learning domain-independent dialogue policies via ontology parameterisation. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 412.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* .

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

Jason Williams and Steve Young. 2003. Using wizard-of-oz simulations to bootstrap reinforcement-learning-based dialog management systems. In *Proceedings of the 4th SIGDIAL Workshop on Discourse and Dialogue*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* .

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *arXiv preprint arXiv:1611.01628* .

Steve J Young. 2006. Using pomdps for dialog management. In *SLT*. pages 8–13.

Zhou Yu, Alan W Black, and Alexander I Rudnicky. 2017. Learning conversational systems that interleave task and non-task content. *arXiv preprint arXiv:1703.00099* .

Zhou Yu, Alexandros Papangelis, and Alexander Rudnicky. 2015. Ticktock: A non-goal-oriented multimodal dialog system with engagement awareness. In *Proceedings of the AAAI Spring Symposium*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Tiancheng Zhao, Alan W Black, and Maxine Eskenazi. 2015. An incremental turn-taking model with active system barge-in for spoken dialog systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 42.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Tiancheng Zhao, Maxine Eskenazi, and Kyusong Lee. 2016. Dialport: A general framework for aggregating dialog systems. *EMNLP 2016* page 32.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960* .

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 paper, 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*—Recurrent neural network (RNN), slot filling, spoken language understanding (SLU), word embedding.**

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.

Manuscript received July 16, 2014; revised October 29, 2014; accepted December 01, 2014. Date of current version February 26, 2015. This work was supported in part by Compute Canada and Calcul Québec. Part of this work was done while Y. Dauphin and G. Mesnil were interns at Microsoft Research. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. James Glass.

G. Mesnil is with the Department of Computer Science, University of Rouen, 76821 Mont-Saint-Aignan, France, and also with the Department of Computer Science, University of Montreal, Montreal, QC H3T 1J4, Canada (e-mail: gregoire.mesnil@umontreal.ca).

Y. Dauphin and Y. Bengio are with the Department of Computer Science and Operations Research, University of Montreal, Montreal, QC H3T 1J4, Canada.

K. Yao, L Deng, X. He, D. Yu, and G. Zweig are with Microsoft Research, Redmond, WA 98052-6399 USA.

D. Hakkani-Tur is with Microsoft Research, Mountain View, CA 94043 USA.

G. Tur is with Apple, Cupertino, CA 95014 USA.

L. Heck is with Google, Mountain View, CA, 94043 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2014.2383614

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 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.

ATIS UTTERANCE EXAMPLE IOB REPRESENTATION

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*							

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.

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:

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

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 $L_1^N = l_1, \ldots, l_N$, the linear-chain CRF models the conditional probability of a concept/slot sequence $S_1^N = s_1, \ldots, s_N$ as follows:

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

where

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

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,

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

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], [69], [70] 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], [65], [66]. 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

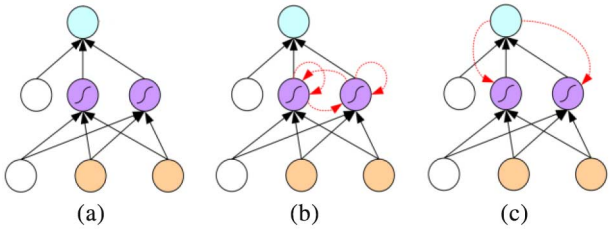


Fig. 1. Three types neural networks. (a) Feed-forward NN; (b) Elman-RNN; (c) Jordan-RNN.

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 vised 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 $\tilde{l}_{i-d}^{i+d} = [\tilde{l}_{i-d}, \ldots, \tilde{l}_i, \ldots, \tilde{l}_{i+d}]^T \in \mathbb{R}^{|V|(2d+1)}$ corresponds to the concatenation of one-hot word index vectors $\tilde{l}_i$.

$$\tilde{l}_i\,(''flight'') = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow \text{The index of word } \textit{flight} \text{ in the vocabulary}$$

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." $l_{\boldsymbol{form}}$ 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 Fig. 1.

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:

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

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 $\underset{i}{\mathrm{argmax}} P_i(y(t-1))$ 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:

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

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 $\vec{h}(t)$ and the backward $\overleftarrow{h}(t)$ hidden layers:

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

where $\vec{U}$ 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:

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

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, $\mathrm{s_i} \in \mathrm{S}$ as states, where the state transition probabilities, $\mathrm{P}_{\{\mathrm{LM}\}}(\mathrm{s_i}|\mathrm{s_j})$ 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.

$$\begin{aligned}\hat{S} &= \arg\max_{\mathrm{S}} \mathrm{P}(\mathrm{S}|\mathrm{L}) \\ &= \arg\max_{\mathrm{S}} \mathrm{P}_{\{\mathrm{LM}\}}(\mathrm{S})^{\alpha} \times \mathrm{P}(\mathrm{L}|\mathrm{S}) \\ &\sim \arg\max_{\mathrm{S}} \mathrm{P}_{\{\mathrm{LM}\}}(\mathrm{S})^{\alpha} \\ &\quad \times \prod_t P_{\{RecNN\}}(s_t|l_t)/P(s_t)\end{aligned}$$

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 $\mathrm{t} = 0$. Eq. (2) is re-written as follows

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

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).

TABLE I
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

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

TABLE II
MEASUREMENT OF THE IMPACT OF USING DIFFERENT WAYS OF TRAINING THE MODELS AND RANDOM SEED ON THE PERFORMANCE

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

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

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 II. 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.

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.

TABLE III
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

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 IV
PERFORMANCE WITH NAMED ENTITY FEATURES

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

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 III. 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.

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].

For the ATIS dataset, we used the gazetteers of flight related entities, such as airline or airport names as named entities. In Table IV, 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

TABLE V
COMPARISON BETWEEN MANUALLY LABELED WORD AND ASR OUTPUT

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

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.

Table V 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 VI 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.

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 VI (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 IV. 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.

TABLE VI
COMPARISON WITH VITERBI DECODING WITH DIFFERENT METHODS ON SEVERAL DATASETS

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 VII
COMPARISON WITH R-CRF AND RNN ON ATIS, MOVIES, AND ENTERTAINMENT DATASETS

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

As shown in Table VII, 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. Further, the recurrent neural networks for the SLU application as presented in this paper can be generalized to other types of speech-centric information processing tasks [67], [68].

References

[1] G. Tur and R. De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY, USA: Wiley, 2011.

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

[3] P. Haffner, G. Tur, and J. Wright, "Optimizing SVMs for complex call classification," in *Proc. ICASSP*, 2003, pp. 632–635.

[4] S. Yaman, L. Deng, D. Yu, Y.-Y. Wang, and A. Acero, "An integrative and discriminative technique for spoken utterance classification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 6, pp. 1207–1214, Aug. 2008.

[5] Y. Wang, L. Deng, and A. Acero, "Spoken Language Understanding—An Introduction to the Statistical Framework," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 16–31, Sep. 2005.

[6] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. ICML*, 2001.

[7] Y. Wang, L. Deng, and A. Acero, , Tur and D. Mori, Eds., "Semantic frame based spoken language understanding," in *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY, USA: Wiley, 2011, ch. 3, pp. 35–80.

[8] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 20, no. 1, pp. 33–42, Jan. 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 *Proc. JMLR W&CP: Proc. Unsupervised Transfer Learn.*, 2011, vol. 7.

[10] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 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 *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2013.

[12] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for Deep Belief Nets," *Neural Comput.*, vol. 18, pp. 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 *Proc. IEEE SLT*, 2012, pp. 210–215.

[14] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, pp. 13-281–305, 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 *Proc. Interspeech*, 2013.

[16] J. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, no. 2, 1990.

[17] M. Jordan, Serial order: a parallel distributed processing approach Univ. of California, Inst. of Comput. Sci., San Diego, CA, USA, Tech. Rep no 8604, 1997.

[18] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[19] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[20] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[21] H. Schwenk and J.-L. Gauvain, "Training neural network language models on very large corpora," in *Proc. HLT/EMNLP*, 2005.

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

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

[24] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Proc. 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 Sci. Comput. Conf. (SciPy)*, 2010.

[26] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Proc. NIPS*, 2000.

[27] A. Deoras and R. Sarikaya, "Deep belief network based semantic tagger for spoken language understanding," in *Proc. Interspeech*, 2013.

[28] A. McCallum, D. Freitag, and F. Pereira, "Maximum entropy Markov models for information extraction and segmentation," in *Proc. ICML*, 2000, pp. 591–598.

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

[30] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proc. ICASSP*, 2011, pp. 5680–5683.

[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 *Proc. ICASSP*, 1992, pp. 193–196.

[32] Y.-Y. Wang and A. Acero, "Discriminative models for spoken language understanding," in *Proc. ICSLP*, 2006.

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

[34] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proc. Interspeech*, 2007.

[35] S. Miller, R. Bobrow, R. Ingria, and R. Schwartz, "Hidden understanding models of natural language," in *Proc. ACL*, 1994.

[36] M. Henderson, M. Gasic, B. Thomson, P. Tsiakoulis, K. Yu, and S. Young, "Discriminative spoken language understanding using word confusion networks," in *Proc. IEEE SLT*, 2012.

[37] R. Kuhn and R. De Mori, "The application of semantic classification trees to natural language understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 5, pp. 449–460, May 1995.

[38] G. Tur, D. Hakkani-Tür, and L. Heck, "What is left to be understood in ATIS," in *Proc. IEEE SLT*, 2010.

[39] G. Tur, L. Deng, D. Hakkani-Tür, and X. He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *Proc. ICASSP*, 2012, pp. 5045–5048.

[40] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.

[41] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random field for language understanding," in *Proc. ICASSP*, 2014, pp. 4105–4009.

[42] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao, "Recurrent conditional random fields," in *Proc. NIPS Deep Learn. Workshop*, 2013.

[43] J. Peng, L. Bo, and J. Xu, "Conditional neural fields," in *Proc. NIPS*, 2009.

[44] D. Yu, S. Wang, and L. Deng, "Sequential labeling using deep-structured conditional random fileds," *J. Sel. Topics Signal Process.*, vol. 4, no. 6, pp. 965–973, Dec. 2010.

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

[46] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. 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 *Proc. 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 *Proc. ICASSP*, 2013, pp. 6664–6668.
[49] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. IEEE SLT*, 2012, pp. 234–239.
[50] Y. Dauphin, G. Tur, D. Hakkani-Tur, and L. Heck, "Zero-shot learning and clustering for semantic utterance classification," in *Proc. Int. Conf. Learn. Represent. (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 *Proc. Interspeech*, 2012.
[52] T. Kudo and Y. Matsumoto, "Chunking with support vector machine," in *Proc. ACL*, 2001.
[53] M. Macherey, F. Och, and H. Ney, "Natural language understanding using statistical machine translation," in *Proc. Eur. Conf. Speech Commun. Technol.*, 2001, pp. 2205–2208.
[54] M. Jeong and G. Lee, "Structures for spoken language understanding: A two-step approach," in *Proc. ICASSP*, 2007, pp. 141–144.
[55] V. Zue and J. Glass, "Conversational interface: Advances and challenges," *Proc. IEEE*, vol. 88, no. 8, pp. 1166–1180, Aug. 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 *Proc. IEEE SLT*, 2014.
[57] Y. Bengio, *Learning deep architectures for AI*. Norwell, MA, USA: Now, 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 *Proc. CIKM*, 2014.
[59] R. Socher, B. Huval, C. Manning, and A. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proc. EMNLP-CoNLL*, 2012, pp. 1201–1211.
[60] W. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proc. ACL*, 2014.
[61] M. Yu, T. Zhao, D. Dong, H. Tian, and D. Yu, "Compound embedding features for semi-supervised learning," in *Proc. NAACL-HLT*, 2013, 2013, pp. 563–568.
[62] D. Hakkani-Tur, L. Heck, and G. Tur, "Exploiting query click logs for utterance domain detection in spoken language understanding," in *Proc. ICASSP*, 2011, pp. 5636–5639.
[63] L. Heck and D. Hakkani-Tur, "Exploiting the semantic web for unsupervised spoken language understanding," in *Proc. IEEE-SLT*, 2012, pp. 228–233.
[64] L. Heck and H. Huang, "Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2014.
[65] L. Deng and D. Yu, *Deep Learning: Methods and Applications*. Delft, The Netherlands: Now, 2014.
[66] J. Gao, P. Pantel, M. Gamon, H. He, and L. Deng, "Modeling interestingness with deep neural networks," in *Proc. EMNLP*, 2014, pp. 2–13.
[67] X. He and L. Deng, "Speech-centric information processing: An optimization-oriented approach," *Proc. IEEE*, vol. 101, no. 5, pp. 1116–1135, May 2013.
[68] X. He and L. Deng, "Speech recognition, machine translation, and speech translation—A unified discriminative learning paradigm," *IEEE Signal Process. Mag.*, vol. 28, no. 5, pp. 126–133, Sep. 2011.
[69] G, Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[70] D. Yu and L. Deng, *Automatic Speech Recognition—A Deep Learning Approach*. New York, NY, USA: Springer, Oct. 2014.

Grégoire Mesnil is a Ph.D. student in computer science at the University of Montréal in Canada and the 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 an M Sc. in machine learning from École Normale Supérieure de Cachan and a B.Sc. in applied mathematics from the University of Caen in France. He also received an Engineer degree in applied mathematics from the National Institute of Applied Sciences in Rouen.

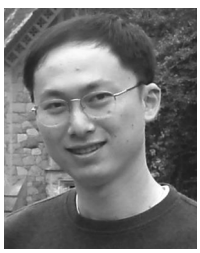
Yann Dauphin is a Machine Learning Researcher and Computer Engineer. He is currently finishing his Ph.D. at the University of Montreal on deep learning algorithms for large-scale problems. He received an M.S. degree in computer science from the University of Montreal in Canada in 2011 and a B.Eng. degree 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 was an Invited Researcher at the Advanced Telecommunication Research Lab in Japan. From 2002 to 2004, he was a Post-Doc Researcher at the Institute for Neural Computation at the 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 the IEEE and a member of ACL.

Yoshua Bengio received the Ph.D. degree in computer science from McGill University in 1991. He was a post-doc with Michael Jordan at MIT and worked at AT&T Bell Labs before becoming a Professor at the University of 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 the 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.

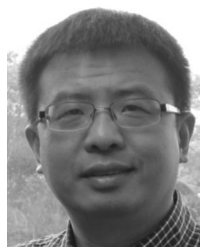
Dilek Hakkani-Tür (F'14) is a Principal Researcher at Microsoft Research. Prior to joining Microsoft, she was a Senior Researcher at the International Computer Science Institute (ICSI) speech group (2006–2010) and she was a Senior Technical Staff Member in the Voice Enabled Services Research Department at AT&T Labs-Research in Florham Park, NJ (2001–2005). She received her B.Sc. degree from Middle East Technical University in 1994 and the M.Sc. and Ph.D. 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 the 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 with 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. He and his colleagues have developed entries that obtained No. 1 place in the 2008NIST 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 the IEEE SIGNAL PROCESSING MAGAZINE and IEEE SIGNAL PROCESSING LETTERS, as Guest Editors of the IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING 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 the 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 Ph.D. 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. He 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. He is also the recipient of the *Speech Communication Journal* Best Paper awards by ISCA for 2004–2006 and by EURASIP for 2005–2006. He 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 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 the 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, he 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 Ph.D. 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.