Modeling with Recurrent Neural Networks for Open Vocabulary Slots

Jun-Seong Kim
Samsung Electronics Co., Ltd /
Suwon, South Korea
Js087.kim@samsung.com

Junghoe Kim, SeongUn Park,
Kwangyong Lee, Yoonju Lee
Samsung Electronics Co., Ltd /
Suwon, South Korea
{kjh94, seongun.park,
ky4you.lee,
yo0nju.lee}@samsung.com

Abstract

Dealing with 'open-vocabulary' slots has been among the challenges in the natural language area. While recent studies on attention-based recurrent neural network (RNN) models have performed well in completing several language related tasks such as spoken language understanding and dialogue systems, there has been a lack of attempts to address filling slots that take on values from a virtually unlimited set. In this paper, we propose a new RNN model that can capture the vital concept: Understanding the role of a word may vary according to how long a reader focuses on a particular part of a sentence. The proposed model utilizes a long-term aware attention structure, positional encoding primarily considering the relative distance between words, and multi-task learning of a character-based language model and an intent detection model. We show that the model outperforms the existing RNN models with respect to discovering 'open-vocabulary' slots without any external information, such as a named entity database or knowledge base. In particular, we confirm that it performs better with a greater number of slots in a dataset, including unknown words, by evaluating the models on a dataset of several domains. In addition, the proposed model also demonstrates superior performance with regard to intent detection.

1 Introduction

In spoken dialogue systems and goal-oriented dialogue systems, slot filling and intent detection are primarily involved in a process of understanding user utterances based on pre-designed semantic frames. Slot filling is to identify a sequence of tokens and extract semantic constituents from the utterances, and intent detection is to classify the intent of the utterances. Two examples relevant to these problems are shown in Figure 1 for the domains of flight and message.

In recent years, a significant amount of interest has built up around slot filling and intent detection due in part to the competition among commercial artificial intelligence assistants, such as Samsung Bixby, Apple Siri, Google Assistant, Microsoft Cortana, and Amazon Alexa. As a result, a multitude of studies regarding recurrent neural network (RNN) models for slot filling and intent detection has been completed and has resulted in outcomes that generally outperform most other past approaches. In particular, there have been various attempts to improve the performance of RNNs for spoken dialogue systems such as label sampling (Liu and Lane, 2015), the hybrid type of Elman and Jordan (Mesnil et al,. 2015), Deep LSTM (Yao et al., 2014), external memory (Peng et al., 2015), bidirectional LSTM (Hakkani-Tur et al., 2016), encoder-labeler and Deep LSTM (Kurata et al., 2016), attention (Liu and Lane, 2016), bidirectional LSTM and CRF (Huang et al., 2015), word hashing (Ravuri and Stolcke, 2015a; Ravuri and Stolcke, 2015b) CNN and bidirectional LSTM (Chiu and Nichols, 2015), external word embeddings (Kim et al., 2016), multi-task learning of a word-based language model (Shi et al., 2015), and multi-task learning of intent and domains (Shi et al., 2015; Hakkani-Tur et al., 2016; Bapna et al., 2017).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2778–2790
Santa Fe, New Mexico, USA, August 20-26, 2018.

Sentence	find	flights	to	new	york	tomorrow
Slots	O	O	O	B-toloc	I-toloc	B-date
Intent	find_flight					

Sentence	text	to	mary	that	sad	puppy	Noise
Slots	O	O	B-recipient	O	B-text	I-text	I-text
Intent	send_message						

Figure 1: Two examples of a flight domain and a message domain with intent and slot annotation, following the IOB (in-out-begin) slot representation

Although most previous studies that exploited RNN and attention methods (Liu and Lane, 2016; Vaswani et al., 2017) were able to achieve state-of-the-art performance across a wide range of natural language research areas, they failed to deal with 'open-vocabulary' slots, which is one of the most important problems in the context of end-user experience using voice assistant. 'Open-vocabulary' slots signify a slot type that can take on values from a virtually unlimited set, such as file name, album name, text body, or schedule title. Therefore, words that are not seen in the training set or are employed differently from the meaning inherent in them can be included in sentences because of the fact that they have no constraints on the length and specific patterns of content. For slot filling of 'open-vocabulary' slots with such characteristics, the goal of this study was to capture the vital concept in which the meaning or role of words could vary according to how long a reader focuses on a particular part of a sentence. For example, in the utterance "Send how about dinner tonight to him" in the message domain, if a reader focuses on 'how about dinner tonight,' the word 'tonight' signifies time. However, it should rather focus on the entire utterance that starts with "Send how …" and recognize the word 'tonight' as a part of the text body. Thus, it is necessary to understand the holistic semantics at the level of the whole sentence, not only the vicinity of a word.

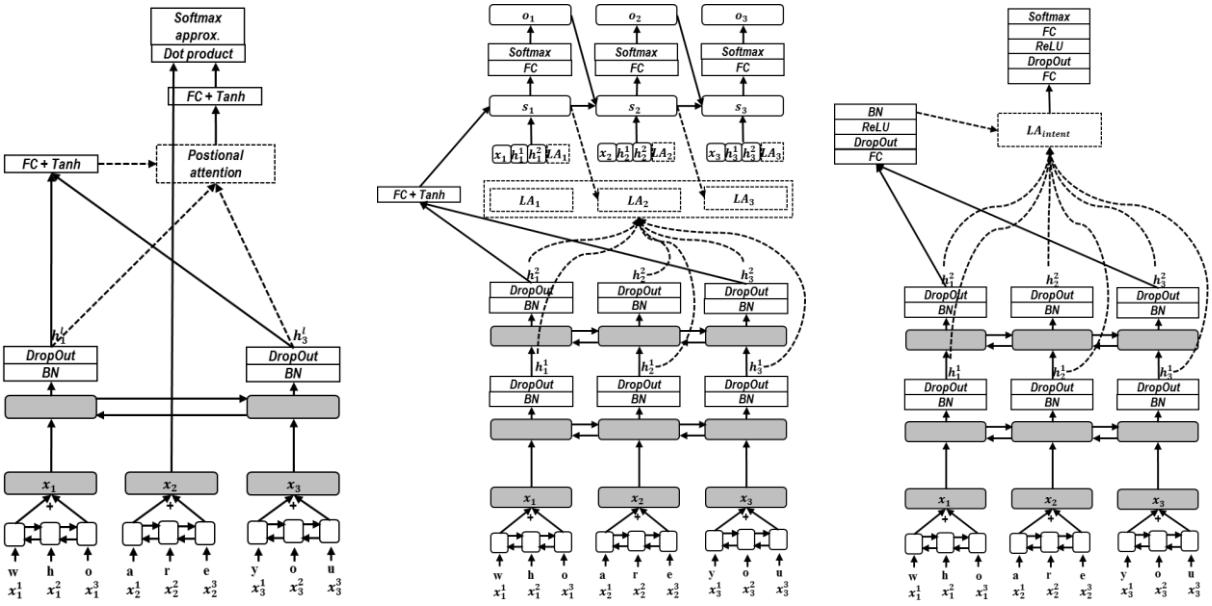
Our model embodies the above concept by introducing a long-term aware attention structure and positional encoding primarily intended for the relative distance between words. Long-term aware attention structure utilizes and stacks two layers of an RNN and treats them as a pair of short-term and long-term meanings. The lower layer denotes short-term or relatively local information corresponding to the vicinity of a word, and the upper layer denotes long-term or relatively global information corresponding to the whole sentence. A novel positional encoding, called the light-house encoding, considers the relative distance between current word and the word being compared in relation to the current word. In addition, it is completely based on characters rather than words, which is adequate for unknown words of 'open-vocabulary' slots. It is also trained with multi-task learning of a character-based language model and an intent detection model to improve the stability and utility of character-based representation and to obtain additional sentence-level information. The language model serves as a regularizer, and the intent detection model is beneficial in the presence of 'open-vocabulary' slots because it allows the proposed model to refer to the global information of a sentence. Lastly, to focus on the effectiveness of modeling, this study only uses the lexical input of language without the help of any external information, such as a named-entity or knowledge base.

We evaluated our model on slot filling and intent detection with the ATIS corpus (Hemphill et al., 1990) and five in-house domains. For the ATIS corpus, the proposed model achieved a state-of-the-art result on intent detection and a result comparable to the state-of-the-art result on slot filling. The proposed model also outperformed the existing RNN models for both tasks on the dataset of five domains. In particular, we demonstrate that our model has outstanding performance on 'open-vocabulary' slots and its efficacy is increased as more data in a domain contain 'open-vocabulary' slots including unknown words.

The main contribution of this study is a new RNN-based model that captures global information of words for discovering 'open-vocabulary' slots. To achieve this, we propose a long-term aware attention structure and a novel positional encoding with help from character-based modeling and multi-task learning.

2 Slot Filling and Intent Detection

Slot filling is a sequence labeling problem and intent detection can be treated as a classification problem that has multiple output labels. In the example in Figure 1, the flight domain may contain *find_flight*, *find_airfare*, and *find_distance* as intent list and *toloc*, *fromloc*, *date*, and *flight_number* as



(a) (b) (c)

Figure 2. Structures of our proposed model (a) Structure of our language model (b) Structure of our slot filling model (c) Structure of our intent detection model

the slot list. The task of slot filling is to find an explicit alignment between slots and a given user query, and the task of intent detection is to instantiate one or more intents from the intent list for a given user query.

More formally, we want to learn two functions for the $f_{\text{slot}} : \mathrm{X} \rightarrow \mathrm{Y}$ and $f_{\text{intent}} : \mathrm{X} \rightarrow \mathrm{Y}'$ that maps an input sequence $\mathrm{X} = x_1, x_2, \ldots, x_{T_x}$ to the corresponding label sequence $\mathrm{Y} = y_1, y_2, \ldots, y_{T_x}$ and intent labels $\mathrm{Y}' = y'_1, \ldots, y'_n$ where T_x is the length of the input sequence and n is the number of intents. The sequence of slots is in the form of IOB labels that have three outputs corresponding to 'B' as the beginning of a slot, 'I' as the continuation of a slot, and 'O' as the absence of a slot.

3 Model Description

The structure of our proposed model is shown in Figure 2. The input of the network is a sequence of characters x_i^j where i is a word index and j is a character index inside a word. Our model includes task models that include both the slot filling model (SFM) in (b) of Figure 2 and the intent detection model (IDM) in (c) of Figure 2, as well as language model (LM) in (a) of Figure 2. All three models share character embedding and a word-related layer to build word representation x_i, and task models (SFM and IDM) share two encoding layers as well to build sentence-level representations which are h_i^1 as the output of the 1st encoding layer and h_i^2 as the output of the 2nd encoding layer. The encoding layer of LM is defined separately as h_i^l because it is an independent encoding layer only for LM, which is unlike h_i^1 and h_i^2 which are shared between SFM and IDM. In addition, the slot filling model utilizes sequential outputs as input of the next decoder in the shape of a slot label embedding o_i.

Basically, we use a bi-directional GRU (bGRU) as the RNN of the encoder and decoder in all models (Chung et al., 2014). The locations of dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015), selection of the activation function, connection between representations of the encoder and inputs of the decoder, and choices of summation and concatenation are obtained from extensive experiments.

3.1 Character-based Word Representation

A layer of bGRU to build word representation x_i, depicted in the bottommost layer of Figure 2, encodes word information into hidden representations from character inputs. Word representation x_i is the sum of two final hidden states of the bGRU. The first one is a vector concatenating forward pass and backward pass at the first time step of characters. The other one is the same type vector at the last time step of characters. All parameters including character embeddings in the bGRU are shared in SFM, IDM, and LM.

We chose character-based embeddings without word-based embeddings for two reasons. Firstly, for large vocabulary tasks, the size of the word embeddings overwhelms the number of other parameters if word embedding is used. Secondly, the character-based approach could give us better robustness in terms of unknown and rare words than the word-based approach, which is an important aspect for 'open-vocabulary' slots. The larger the number of words we model, the more the sparseness problem is exacerbated. Furthermore, Kim et al., (2015) said that word representation based on character embedding is able to better detect orthographic forms and semantic features and better learn to differentiate between morphemes, such as prefixes and suffixes, than word embedding. Moreover, Verwimp et al. (2017) claimed that it is possible to reveal structural similarities between words and be used when a word is out-of-vocabulary, unknown and infrequent.

3.2 Long-term Aware Attention Structure

Basically, the existing attention mechanism in alignment-based RNN models has the following form (Mnih et al., 2014; Bahdanau et al., 2015; Firat et al., 2016).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{1}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2}$$

$$e_{ij} = f(s_{i-1}, h_j) \tag{3}$$

where j is the sequential index of a word, s_{i-1} is the decoder state at step $i-1$, c_i is a corresponding attention vector, α_{ij} is position-wise weights related to s_{i-1} which is obtained by the function f.

In this formula, the query of attention is the previous decoder state s_{i-1}, and the keys and values of attention are each encoding information h_j. An attention weight α_{ij} is obtained from some operations including softmax and neural network f based on query and keys, and final attention vector c_i is made by weighted sum of attention weights and values.

To address the challenge presented at the beginning of the paper, we took two ideas which embody the concept where the meaning or role of words can vary according to how long a reader focuses on a particular part of a sentence. The first idea is to build multiple RNNs hierarchically and each upper layer uses final forward and backward hidden states of a layer right below it as initial hidden states in opposite directions. For example, SFM and IDM in our model share two layers of the bGRU as a sentence-level encoder as follows:

$$\overrightarrow{h_i^1} = \mathrm{GRU}_1\left(\overrightarrow{h_{i-1}^1}, x_i\right), \overleftarrow{h_{i-1}^1} = \mathrm{GRU}_1\left(\overleftarrow{h_i^1}, x_i\right), i = 1, \ldots, T_x \tag{4}$$

$$\overrightarrow{h_0^1} = 0_d, \overleftarrow{h_{T_x+1}^1} = 0_d \tag{5}$$

$$h_i^1 = \left[\overrightarrow{h_i^1}; \overleftarrow{h_i^1}\right] \tag{6}$$

$$\overrightarrow{h_i^2} = \mathrm{GRU}_2\left(\overrightarrow{h_{i-1}^2}, h_i^1\right), \overleftarrow{h_{i-1}^2} = \mathrm{GRU}_2\left(\overleftarrow{h_i^2}, h_i^1\right), i = 1, \ldots, T_x \tag{7}$$

$$\overrightarrow{h_0^2} = \overleftarrow{h_1^1}, \overleftarrow{h_{T_x+1}^2} = h_{T_x}^1 \tag{8}$$

$$h_i^2 = \left[\overrightarrow{h_i^2}; \overleftarrow{h_i^2}\right] \tag{9}$$

where $\overrightarrow{\ }$ and $\overleftarrow{\ }$ are the forward pass and backward pass of the bGRU, $\overrightarrow{h_0^1}$ and $\overrightarrow{h_0^2}$ are the initial hidden states of the forward pass for the 1st layer and the 2nd layer, $\overleftarrow{h_{T_x+1}^1}$ and $\overleftarrow{h_{T_x+1}^2}$ are the initial hidden states of the backward pass for the 1st layer and the 2nd layer, and [;] is the concatenation operator.

The intuition behind equation (8) is that each pass in an upper bGRU layer provides more global information than a lower one and updates the states of each word time step to redeem correlations between distant words with the help of sentence-level information in the opposite direction of lower bGRU layer. In other words, we stack two layers of the bGRU and treat them as a pair of short-term and long-term meanings. The first layer output h_i^1 of two bGRU layers denotes short-term or relatively local information and the second layer output h_i^2 denotes long-term or relatively global information.

The second idea is that, in the attention mechanism of the decoder, the output h_i^1 of the lower layer is used as both a value and a key and output h_i^2 of the upper layer is used only as a key. We intended that the model would keep short-term information for the concrete meaning of a word, and combine it

using additional sentence-level information for actual meaning or role in a sentence learned from training data. More formally,

$$LA_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j^1 \tag{10}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{11}$$

$$e_{ij} = f\left(s_{i-1}, h_j^1, h_j^2\right) \tag{12}$$

where LA_i is an attention vector at each decoding step i, h_j^1 is a hidden state of the 1st bGRU layer at word index j, h_j^2 is the hidden states of the 2nd bGRU layer at word index j.

We expect these simple changes allow the model to compensate for missing points or defects at the level of the whole sentence, and also disentangle the local and global requisite information as appropriate for the tasks. The key features of the ideas above are the ability to avoid insufficient tackling of long range dependencies and the ability to offer assistance in learning the actual meaning or role within a sentence. Although the LSTM (Gers and Schmidhuber, 2000) and GRU are well designed to solve the problem of long range dependencies that results in the model becoming unable to capture correlations between words separated by many time steps as a distance between the current word and the relevant word grows, the long range dependencies may still be a lingering issue, especially for 'open-vocabulary' slots, from an approximated learning method like truncated backpropagation through time (truncated BPTT), noisy and meaningless interim inputs, and ambiguity.

In practice, we constitute the function f in (12) used for computing weights by fully connected neural networks.

$$f\left(s_{i-1}, h_j^1, h_j^2\right) = v_a^T \left(Tanh\left(FC(s_{i-1})\right) + Tanh\left(FC\left(h_j^1\right)\right) + Tanh\left(FC\left(h_j^2\right)\right)\right) \tag{13}$$

where FC is a fully connected neural network with a linear matrix and a bias vector to be projected into attention dimension d_a and $v_a \in R^{d_a}$ is a vector to compute one scalar weight logit for each decoding step i in the decoder and word index j of hidden states in the encoder.

3.3 Light-house Positional Encoding

Positional encoding conveys positional information in a form of vectors, which is related to where words are located within a sentence. To appropriately impose the second idea of the long-term aware attention structure on our model, positional encoding in the attention mechanism of the decoder is necessary because the encoder is able to focus more on the meaning in a sentence as intended by the fact that applying it to the attention mechanism separately conveys interactions depending on the distance between words to the decoder.

We introduce a novel positional encoding, called light-house positional encoding, which considers the relative distance between the current word and the word being compared in relation to the current word. Our scheme has two differences from the previous positional encodings (Sukhbaatar et al., 2015; Vaswani et al., 2017; Gehring et al., 2016; Gehring et al., 2017). One is that it defines only one trained embedding, which is unlike position-wise encoding and fixed encoding, and the other is that it is not placed in the inputs of the encoder and decoder but on the inside of attention computation. The formula is as follows:

$$p_i^b = \beta * |b - i| * D \tag{14}$$

where i is a word index, b is the current word index as a basis, $D \in R^d$ is the distance embedding which has positional dimension d, β is a parameter which controls the norm of the encoding increased by the distance.

The existing positional encodings are mostly fixed encoding like sinusoid type or learned position-wise encoding which is set up per each word independently. However, light-house positional encoding defines only one distance embedding D with respect to one time step between words. It configures positional encodings p_i^b for each word index i according to time step distance $|b - i|$ from the current word index b multiplied by a parameter β. Thus, it has fewer position-related parameters than the position-wise encoding. Additionally, the further the distance between words being compared, the larger the norm of positional encoding it has, which makes deeper layers easily notice distance-related information from positive and negative values of encoding and weights. In our experiments, the distance embeddings of models for several domains have positive and negative values in a 50/50 ratio on aver-

age. In practice, although we are able to deal with β as a learned parameter, we treat it as a hyper-parameter, which is closely connected to the average length of sentences. $\beta = 1$ showed stable results in our experiments.

As for placement, we place it only inside of the attention computation function f of (13) as follows:

$$f(s_{i-1}, h_j^1, h_j^2) = v_a^T \left(Tanh(FC(s_{i-1})) + Tanh\left(FC([h_j^1; p_j^i])\right) + Tanh\left(FC([h_j^2; p_j^i])\right)\right) \quad (15)$$

where [;] is concatenation operator and p_j^i is the positional encodings for each word index j of hidden states in the encoder based on each decoding step i in the decoder.

The reason for this placement is also for the sake of using positional encoding only where it is absolutely necessary because the other layers are composed of the bGRU and it exhibits dynamic temporal behavior.

3.4 Configuration for Models and Multi-task Learning

The SFM in (b) of Figure 2 is composed of a layer of the bGRU to build word representation x_i, two layers of the bGRU in the encoder, and a layer of the bGRU including the attention mechanism in decoder. The encoder and decoder of the SFM are based on the long-term attention structure and light-house positional encoding that we explained above. In addition, we utilize both ends of the hidden states of the 2nd encoder layer, h_1^2 and $h_{T_x}^2$, to make an initial state of decoder s_0 by concatenating them and using a layer of the fully connected neural network with a hyperbolic tangent 'tanh'. That means that the initial state of the decoder is built from states with the broadest information in the encoder.

The IDM in (c) of Figure 2 is composed of encoder layers shared with the SFM and two layers of the fully connected neural network in the decoder. It is also based on the long-term attention structure and light-house positional encoding. The difference with the STM in relation to using two ideas is that there is only one attention vector LA_{intent}, its positional encoding is computed by treating the last word time step as the current word index b in (14), and a query of attention is made by a layer of the fully connected neural network with a rectified linear unit 'ReLU', not 'tanh'.

The LM in (a) of Figure 2 is composed of a layer of the bGRU for word representation x_i shared with the SFM and IDM, a separate layer of the bGRU in the encoder, and a layer of the fully connected neural network in the decoder. It adopts a conventional attention structure like (1) to (3) and light-house positional encoding, and it also has only one attention vector called 'Positional attention' like the IDM. The LM was trained by randomly removing a word in a sentence, inserting other words within window 10 ($\pm$5) of the removed word as an input, and using the removed word as an output. Thus, positional encoding in the LM treats the index of the removed word as the current word index b = 5 in (14), and its 10 indexes of input are 0, …,4,6, …,10. Finally, the formula of the decoder is as follows:

$$Positional\ attention = \sum_j \alpha_j h_j^l, j = 0, \ldots, 4, 6, \ldots, 10 \quad (16)$$

$$\alpha_j = \frac{\exp(e_j)}{\sum_k \exp(e_k)}, k = 0, \ldots, 4, 6, \ldots, 10 \quad (17)$$

$$e_j = f(s, h_j^l) \quad (18)$$

$$f(s, h_j^l) = v_a^T \left(Tanh(FC(s)) + Tanh\left(FC([h_j^l; p_j^5])\right)\right) \quad (19)$$

$$\mathrm{s} = Tanh\left(FC([h_0^l; h_{10}^l])\right) \quad (20)$$

where h_j^l is the outputs of a separate encoding layer for $j = 0, \ldots, 4, 6, \ldots, 10$, s is a query of attention, FC is a fully connected neural network with a linear matrix and a bias vector to be projected into attention dimension d_a and $v_a \in R^{d_a}$ is a vector to compute one scalar weight logit for each word index j of hidden states in the encoder, p_j^b is a positional encoding for each word index j of the hidden states in the encoder based on the word index 5 corresponding to the removed word.

In the LM, the decoder formulates one representation from 'Positional attention' via a layer of the fully connected neural network with 'tanh' and we use the dot product between the word representation for each word candidate and the final output to compute the final probability, which indicates how well each word fits into the position of the removed word. In addition, the LM utilizes infrequent normalization softmax approximation, combining the strengths of Noise Contrastive Estimation (NCE) and self-normalization, which results in a higher speed-up factor (Andreas and Klein, 2015).

Domains	# of train	# of test	# of slot labels	# of intent types
Calculator	3401	680	10	2
Calendar	9062	1812	70	24
Camera	9425	1885	63	61
Gallery	61885	12377	54	197
Message	18571	3714	49	74

Table 1. The number of training sets, test sets, slot labels, and intent types in 5 Samsung Bixby domains

	ATIS	Gallery	Calendar	Message
Ratio of "open vocabulary slots" to "close vocabulary slots" in the test set	59.58% (1689/2835)	15.13% (5305/35045)	34.56% (1064/3079)	79.44% (3655/4601)
Ratio of "open vocabulary slots including unknown words" to "the other open vocabulary slots" in the test set	2.42% (41/1689)	4.79% (254/5305)	13.82% (147/1064)	45.42% (1660/3655)

Table 2. Percentages of 'open-vocabulary' slots, including unknown words in the test set for each domain

In practice, all words existing in a training batch are treated as a result of down-sampling for approximating the normalization term.

We institute multi-task learning for the above three models with two types called 'Slot only' and 'Joint'. The first one is a combination of the SFM and the LM and the other is a combination of all three models. It also follows that joint learning of intent detection has a positive effect on performance of slot filling as in Liu and Lane (2016). A final cost function to be maximized is as follows:

$$\mathcal{L} = \log\big(p(\mathrm{Y}|\mathrm{X})\big) + \log\big(p(Y'|X)\big) + \lambda\Big(\log\big(p(x_b|X \setminus x_b)\big) + \textstyle\sum_{w_i \setminus embeddings} \| w_i \|_2^2\Big) \quad (21)$$

$$p(\mathrm{Y}|\mathrm{X}) = \textstyle\prod_{t=1}^{T_x} p\big(y_t \big| x_1, \dots, x_{T_x}, y_1, \dots, y_{t-1}\big) \quad (22)$$

$$p(Y'|X) = \textstyle\prod_{k=1}^{n} p\big(y'_k \big| x_1, \dots, x_{T_x}\big) \quad (23)$$

where λ is a weight-decay hyper-parameter for the LM and l2-norm, T_x is the length of the input sequence, n is the number of intents, and w_i is all the parameters learned from training data. Thus, the LM is used for stability and utility of character-based word representation and can be construed as a regularizer for the SFM and IDM.

4 Experiments

4.1 Data

The ATIS corpus (Hemphill et al., 1990) is the most commonly used dataset for the research on spoken language understanding. In this study, we used the ATIS corpus from Hakkani-Tur et al. (2016).[1]

The dataset consists of sentences of people making flight reservations. 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 datasets. There are in total 127 distinct slot labels and 18 different intent types.

For this study we collected English in-house real data of five different Bixby domains. We adopted domains and corresponding intent types to show the performance of our model in terms of various levels of difficulty. Table 1 describes the number of training sets, test sets, slot labels, and intent types in the five domains. The calculator and camera domains have no 'open-vocabulary' slots, and 'open-vocabulary' slots of the others, including the ATIS corpus, are shown in Table 5 of Appendix B. We did not utilize any external information, such as a named entity database or knowledge base with entities like city, airport, person, etc., to focus on the performance only using the lexical input of language. Thus, we treated named entity slots as 'open-vocabulary' slots. Additionally, we defined the difficulty of each domain associated with 'open-vocabulary' slots on the basis of how many values of 'open-vocabulary' slots in each domain included unknown words in the test set. Table 2 shows that ATIS, gallery, calendar, and message had a high level of difficulty in an increasing order.

For the five Bixby domains, we compared our model to three previous models: the hybrid RNN (Mesnil et al., 2015), the encoder-labeler deep LSTM (Kurata et al., 2016), and the attention biRNN for both 'Slot only' and 'Joint' (Liu and Lane, 2016).

[1] https://github.com/yvchen/JointSLU/tree/master/data

Model	Best F1 score (Slot filling)	Best error rate (Intent detection)
RNN with Label Sampling (Liu and Lane, 2015)	94.89	-
Hybrid RNN (Mesnil et al., 2015)	95.06	-
Deep LSTM (Yao et al., 2014)	95.08	-
RNN-EM (Peng et al., 2015)	95.25	-
bLSTM (Hakkani-Tur et al, 2016)	95.48	-
Encoder-labeler Deep LSTM (Kurata et al., 2016)	95.66	-
Word embeddings updated and bLSTM (Kim et al., 2016)	-	2.69
LSTM (Ravuri and Stolcke, 2015a)	-	2.45
Attention Encoder-Decoder NN (Slot only) (Liu and Lane, 2016)	**95.78**	
Attention BiRNN (Slot only) (Liu and Lane, 2016)	95.75	
Attention Encoder-Decoder NN (Joint) (Liu and Lane, 2016)	95.87	**1.57**
Attention BiRNN (Joint) (Liu and Lane, 2016)	**95.98**	1.79
Proposed model (Slot only)	**95.93**	-
Proposed model (Joint)	**95.93**	**1.46**

Table 3. Best F1 score of slot filling and best error rate of intent detection for ATIS in comparison with previous approaches

Model	Slot filling					Intent detection				
	Gallery	Calculator	Calendar	Message	Camera	Gallery	Calculator	Calendar	Message	Camera
Hybrid RNN (Mesnil et al., 2015)	88.41	97.18	71.57	74.17	87.44	-	-	-	-	-
Encoder-labeler Deep LSTM (Kurata et al., 2016)	94.03	98.2	87.34	88.42	93.51	-	-	-	-	-
Attention BiRNN (Slot only) (Liu and Lane, 2016)	98.48	99.73	93.94	91.07	99.44	-	-	-	-	-
Attention BiRNN (Joint) (Liu and Lane, 2016)	97.31	99.82	91.98	90.50	99.34	1.35	**0.00**	3.20	2.53	**0.53**
Proposed model (Slot only)	99.19	99.83	95.34	94.70	**99.58**	-	-	-	-	-
Proposed model (Joint)	**99.24**	**99.93**	**96.13**	**95.00**	**99.58**	**1.32**	**0.00**	**2.76**	**1.46**	**0.53**

Table 4. Best F1-score of slot filling and best error rate of intent detection for five Bixby domains in comparison with previous approaches

4.2 Training details

For the training of our model, we used a mini-batch stochastic descent method named Adadelta (Zeiler, 2012), a method with a relatively simple configuration, and set its momentum related parameter to a value of 0.95, but we used default settings and methods specified in published codes and papers for the comparison models.[2] We also utilized l2-norm regularization for parameters and gradient clipping for stable training.

For tuning hyper-parameters, we split the training set into 90% training and 10% development. After choosing the hyper-parameters, we re-trained the model with all of the training data. For evaluation, we evaluated the performance on slot filling using an F1 score and the performance on intent detection using the classification error rate, and each result was recorded from the best one among 10 different initializations.

We did not utilize any kind of pre-training or external data for any of the experiments. Details of the preprocessing procedures and the finally determined model parameters are given in Appendix A.

4.3 Results

For the ATIS corpus, Table 3 shows that our models for both 'Joint' and 'Slot only' achieved comparable results to the state-of-the-art result of slot filling. In particular, our 'Slot only' model provided improvement from 95.78 to 95.93 in comparison to the previous best 'Slot only' model. For intent detection, Table 3 also shows that we provided the state-of-the-art performance with a 1.46 error rate using the 'Joint' model. In addition, learning with additional intent information in our model had no positive effect on the performance of slot filling.

On the other five domains of Bixby, we compared our model with three previous methods for slot filling and also compared it with only one previous method for intent detection. The Hybrid RNN

[2] Attention BiRNN : https://github.com/HadoopIt/rnn-nlu and Hybrid RNN : https://github.com/mesnilgr/is13

(Mesnil et al., 2015) applied the RNN to slot filling for the first time, the Encoder-labeler Deep LSTM (Kurata et al., 2016) applied the encoder-decoder concept to slot filling, and the Attention BiRNN (Liu and Lane, 2016) applied attention and multi-task learning to slot filling and intent detection and achieved the state-of-the-art performance among the previous approaches for both. Thus, for slot filling, we intended to check change of the performance as additional ideas are introduced such as the RNN, encoder-decoder, attention, and long-term aware attention and light-house positional encoding. Not to our surprise, adding ideas improved the performance and our model demonstrated the best performance for both slot filling and intent detection. In particular, as the difficulty of domains associated with 'open-vocabulary' slots increased, like calendar and message, a larger gap in performance emerged to the amount of about a 5% absolute gain for slot filling and about a 1% error rate for intent detection. However, multi-task learning with the intent detection model did not have much effect on the performance of slot filling. The results are summarized in Table 4.

To see more detailed results in terms of 'open-vocabulary' slots, we divided the results of slot filling into an F1 score on 'closed-vocabulary' slots and an F1 score on 'open-vocabulary' slots as shown in Table 6 of Appendix C. We highlight some observations.

First, results from our models of both 'Joint' and 'Slot only' had better F1 scores over the previous best model with respect to 'open-vocabulary' slots for domains having them. In accordance with the results of all the slots, our model experienced more benefit in the performance of 'open-vocabulary' slots as the difficulty of the domains we defined increased. Furthermore, the first three samples of Table 7 in Appendix D and all the samples of Table 8 in Appendix D confirm that our model tends to draw conclusions by referring to the whole sentence as we intended.

Second, there was a trade-off to some degree between 'closed-vocabulary' slots and 'open-vocabulary' slots as was revealed in the ATIS corpus case. That is why the performance of our model could not exceed the state-of-the-art result in Table 3. If the distinction or boundary between slots is not clear like 'Boston late night' in a sentence "ground transportation that I could get in Boston late night", the whole can be treated as an 'open-vocabulary' slot due to a shortage of training data about the '*period_of_day*' slot corresponding to 'late night'. We have to recognize it depending only on the content of the sentence without any information about the city name, and moreover, 'boston late night' may be construed as one of city names. In addition, our model tends to open up more possibilities for various conclusions. For an example of "flight ## from jfk to lax", our model considers 'lax' as a '*city_name*' slot despite the fact that '*airport_code*' slot consists of three letters. Samples of slot filling results for the ATIS corpus are in Table 7 of Appendix D.

Third, multi-task learning of our 'Joint' model generally had a higher performance than our 'Slot only' model regarding 'open-vocabulary' slots on all datasets, including the ATIS corpus. Therefore, it is safe to say that the performance of 'open-vocabulary' slots in our approach benefits more from the joint training with intent detection and the help of sentence-level information. On the contrary, unlike the result of Liu and Lane (2016) the 'Joint' model of them had a lower performance than the 'Slot only' model of them for 'closed-vocabulary' slots as well as for 'open-vocabulary' slots.

5 Conclusion and Future Work

In this study, we presented a new modeling with RNNs and successfully dealt with 'open-vocabulary' slots. It focuses more on relatively global information within a sentence using a long-term aware attention structure and light-house positional encoding with the help of multi-task learning of a character-based language model and intent detection model. Compelling experimental results on several domains demonstrated the advantages of our model, especially for 'open-vocabulary' slots.

In this study, we did not utilize additional methods for generalization, such as variational dropout (Gal and Ghahramani, 2016), attention weights dropout, label smoothing, and dropout in decoder (Vaswani et al., 2017; Merity et al., 2017; Melis et al., 2017), and we did not search extensively hyper-parameter space and choices of optimizers extensively as in Merity et al. (2017) and Melis et al. (2017). These could enhance the performance of our RNN model. Furthermore, we will conduct not only a delicate analysis of weights of the long-term aware attention structure but also experiments using models based on architectures other than RNN and hierarchically deeper structures.

Reference

Jacob Andreas and Dan Klein. 2015. *When and why are log-linear models self-normalizing?* in Proc. NAACL.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. *Neural Machine Translation by jointly learning to align and translate.* in ICLR.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. *Towards Zero-Shot Frame Semantic Parsing for Domain Scaling.* in Proc. Interspeech.

Jason PC Chiu and Eric Nichols. 2015. *Named Entity Recognition with Bidirectional LSTM-CNNs.* arXiv preprint arXiv:1511.08308.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* in Proc. NIPS.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. *Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism.* in Proc. NAACL.

Yarin Gal and Zoubin Ghahramani. 2016. *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks.* in Proc. NIPS.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. *A Convolutional Encoder Model for Neural Machine Translation.* arXiv preprint arXiv:1611.02344.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. *Convolutional Sequence to Sequence Learning.* arXiv preprint arXiv:1705.03122.

Felix A. Gers and Jurgen Schmidhuber. 2000. *Recurrent Nets that Time and Count.* In Neural Networks. in Proc. IJCNN.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. *Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM.* in Proc. Interspeech.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. *Bidirectional LSTM-CRF Models for Sequence Tagging.* arXiv preprint arXiv:1508.01991

Sergey Ioffe and Christian Szegedy. 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* in Proc. ICML.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. *Character-Aware Neural Language Models.* CoRR, abs/1508.06615.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. *Intent detection using semantically enriched word embeddings.* in Proc. IEEE SLT.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. *Leveraging Sentence-level Information with Encoder LSTM for Natural Language Understanding.* arXiv preprint arXiv:1601.01530.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. *On the state of the art of evaluation in neural language models.* arXiv preprint arXiv:1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. *Regularizing and Optimizing LSTM Language Models.* arXiv preprint arXiv:1708.02182.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. *Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding.* in Proc. IEEE/ACM.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. *Recurrent Models of Visual Attention.* in Proc. NIPS.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington.1990. *The ATIS Spoken Language Systems Pilot Corpus.* in Proc. DARPA Speech and natural language workshop.

Bing Liu and Ian Lane. 2015. *Recurrent Neural Network Structured Output Prediction for Spoken Language Understanding.* in Proc. NIPS.

Bing Liu and Ian Lane. 2016. *Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling.* in Proc. Interspeech.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. *Recurrent Neural Networks with External Memory for Spoken Language Understanding.* Natural Language Processing and Chinese Computing, pp. 25–35, Springer.

Suman Ravuri and Andreas Stolcke. 2015a. *A comparative study of neural network models for lexical intent classification.* in Proc. IEEE ASRU.

Suman Ravuri and Andreas Stolcke. 2015b. *Recurrent Neural Network and LSTM Models for Lexical Utterance Classification.* in Proc. Interspeech.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. *Contextual spoken language understanding using recurrent neural networks.* in Proc. ICASSP.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* JMLR, 15:1929–1958.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. *End-to-End Memory Networks*. In NIPS.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. *Attention Is All You Need.* arXiv preprint arXiv:1706.03762.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. *Character-Word LSTM Language Models.* in Proc. EACL.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. *Spoken Language Understanding using Long Short-Term Memory Neural Networks.* in Proc. IEEE SLT.

Matthew D. Zeiler. 2012. *ADADELTA: An adaptive learning rate method.* arXiv preprint arXiv:1212.5701.

Appendix A. Experimental Details

We preprocessed the datasets as follows for all datasets. To deal with unseen or unknown inputs in the test set, we replaced all 'words' with at most three occurrences in the training data by <UNK> for comparison models and all 'characters' with at most ten occurrences in the training data also by <UNK> for our model. We also changed all sequences of numbers to sequences of '#', for example, '2017' is converted to '####'. For comparison models, we applied basic tokenizer before training, which is related to apostrophe and a combination of number and word.

For the training of our model, we used the Adadelta with 0.95 momentum related parameter and 5 gradient clipping. Parameters related to batch normalization are initialized by 1, parameters related to the hidden of RNN are initialized by orthogonal values obtained from the singular vector decomposition and the standard normal distribution with scale 1, bias parameters are initialized by zeros, and all other parameters are initialized by the standard normal distribution with scale 0.01. All dimension settings are simply 100 except 50 dimension of a slot label embedding o_i.

Hyper-parameters are 0.5 dropout, $\beta = 1$ hyper-parameter of the light-house positional encoding, 0.02 noise probability of character replacement for each characters in the inputs, 0.1 weight decay for the l2-norm of parameters learned from the training data and the language model cost, and 5 to 12 batch size according to the domains.

Appendix B. Lists of 'Open-vocabulary' Slots

Domains	'Open-vocabulary' slots
ATIS	{*airport_name, airline_name, city_name, state_name, fromloc.airport_name, fromloc.city_name, fromloc.state_name, stoploc.airport_name, stoploc.city_name, toloc.airport_name, toloc.city_name, toloc.country_name, toloc.state_name*}
Gallery	{*album_name, app_name, recipient, contact, file_name, location, poi, recipient, story_name, tag_name, title*}
Calendar	{*app_name, recipient, save_invitee, save_location, save_notes, save_title, search_keyword*}
Message	{*appname, blocked_phrase, search_keyword, text_body, recipient, contact*}

Table 5. List of 'open-vocabulary' slots for ATIS, gallery, calendar, and message.

Appendix C. Slot Filling for 'Open-vocabulary' Slots

Models	Slot types	ATIS	Gallery	Calendar	Messages
Attention BiRNN (Joint) (Liu and Lane, 2016)	Closed vocabulary slots	94.64	97.91	93.71	95.07
	Open vocabulary slots	96.68	93.97	88.69	89.31
Attention BiRNN (Slot only) (Liu and Lane, 2016)	Closed vocabulary slots	**95.06**	98.85	94.82	95.64
	Open vocabulary slots	96.58	96.42	90.88	89.89
Proposed model (Joint)	Closed vocabulary slots	94.38	**99.39**	**97.07**	**97.30**
	Open vocabulary slots	**96.98**	**98.36**	**94.34**	**94.40**
Proposed model (Slot only)	Closed vocabulary slots	94.68	99.34	96.79	96.89
	Open vocabulary slots	96.77	**98.38**	92.57	94.13

Table 6. F1 score on 'open-vocabulary' slots and 'closed-vocabulary' slots for ATIS, gallery, calendar, and message in comparison with the previous best approach.

Appendix D. Slot Filling Samples

Samples	Ground Truth	Slot results of Attention BiRNN (Slot only) (Liu and Lane, 2016)	Slot results of the proposed model
Show me flights from montreal to orlando and **"long beach"**	toloc.city_name	fromloc.city_name	**toloc.city_name**
find me a flight from cincinnati to any airport in the "**new york"** city area	toloc.city_name	fromloc.city_name	**toloc.city_name**
please find all the flights from cincinnati to any airport in the "**new york city"** area that arrive next saturday before # pm	toloc.city_name	fromloc.city_name	**toloc.city_name**
and now show me ground transportation that i could get in "**boston late night"**	city_name (boston) & period_of_day (late night)	**city_name (boston)** & state_code (late) & **period_of_day (night)**	city_name (Boston late night)
list the airfare for american airlines flight ## from jfk to **"lax"**	toloc.airport_code	**toloc.airport_code**	toloc.city_name

Table 7. Samples of the slot filling results for the ATIS corpus. The bold and underlined letters in samples are targets for analysis. The slots in bold means that they are correct and the letters in parentheses are the outputs corresponding to the slots.

Samples	Ground Truth	Slot results of Attention BiRNN (Slot only) (Liu and Lane, 2016)	Slot results of the proposed model
get rid of all the sms with **"bolt from the blue"** in them	search_keyword (bold from the blue)	search_keyword (the blue)	**search_keyword (bolt from the blue)**
Deliver tony jones and nine six zero four a message with **"say it til you believe it"**	text_body (say it til you believe it)	text_body (it til you believe it)	**text_body (say it til you believe it)**
snap a selfie camera picture also share with **"ronnie"**	recipient	search_keyword	**recipient**
Text to lou **"i just finished the last of my exams"**	text_body (I just finished the last of my exams)	text_body (just finished) & search_keyword (exams)	**text_body (I just finished the last of my exams)**
For phone charging on 12 december invite **"kayla green"**	save_invitee (kayla green)	search_keyword (kayla) & save_title (green)	**save_invitee (kayla green)**
i've gotta go **"guatemala city"** with resendez today so place that a new event	save_location	save_title	**save_location**

Table 8. Samples of slot filling results for message and calendar. The bold and underlined letters in samples are targets for analysis. The slots in bold means that they are correct and the letters in parentheses are the outputs corresponding to the slots.