Recurrent Neural Network with Pre-trained Language Model Embedding for Slot Filling Task

Liang Qiu liangqiu@g.ucla.edu
Yuanyi Ding semiswiet@g.ucla.edu
Dui Lin dui@g.ucla.edu

Abstract

Modern methods to solve Slot Filling problem is using Recurrent Neural Network models. However, typical RNNs are data thirsty which limit their use in practical terms, especially for domain specific tasks. In this paper, inspired by the work of Peters et al. (2017), we introduces an extra language model embedding layer whose weights were pre-trained on a large, unlabeled corpus to help encode the context of each token in the word sequence and use it in the supervised sequence tagging model. We use Airline Travel Information System (ATIS) data corpus to compare the performance of our LSTM-LM model with the baseline LSTM model, prove that with the pre-trained language model embedding we can dramatically reduce the size of the training dataset for LSTM without compromising F1 score. We will also introduce GloVe word embedding and bidirectional LSTM, showing that adding them both provide a better performance using the same training dataset.

1 Credits

This paper has been inspired by the trail works in Semi-supervised sequence tagging with bidirectional language models by Peters, M. E., Ammar, W., Bhagavatula, C., Power, R. (2017); The application of Recurrent Neural Network on slot filling by Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ..., Zweig, G. (2015); Bi-directional recurrent neural network by Vu, T. N., Gupta, P., Adel, H., and Schutze, H. (2016); Multi-domain joint semantic frame parsing by Hakkani-Tr, D., Tr, G., Celikyilmaz, A., Chen, Y. N., Gao, J., Deng, L., Wang, Y. Y. (2016). We also benefit a lot from the course material of Current Topics in Artificial Intelligence-Machine Learning in Natural Language Processing at UCLA CS Department by Prof. Kai-Wei Chang.

2 Introduction

The slot filling task is a subtask of Spoken Language Understanding (SLU) and it can also be addressed as a standard sequence labeling or sequence discrimination task. Typical dialogue managers rely on a slot filling functional module to extract additional parameters or information required to determine the appropriate action to take when facing a detected user intent. We can frame the slot filling task using an example in the Airline Travel Information System (ATIS) benchmark which we used for experiments. The following figure shows the a typical sentence in ATIS dataset and its annotation of slot/concept, named entity, intent as well as domain. The last two rows are key tasks other than slot filling in SLU, which are domain detection and intent determination. We can see that the slot filling is quite similar to the Named Entity Recognition (NER) task, following the IOB tagging representation, which is represented in a format of in/out/begin, except that slot filling is more domain and intent specific.

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*							

Figure 1: ATIS utterance example IOB representation

Modern methods to solve slot filling problem includes generative models such as Hidden Markov Model (HMM), discriminative models such as Conditional Random Field (CRF), etc.

With the popularity of RNNs in many other natural language processing tasks such as language modeling and machine translation, RNN was also introduced to handle the slot filling problem and achieved the state-of-the-art performance (Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ..., Zweig, G., 2015).

In Mesnil et al. paper, they tried two basic types of RNNs: Elman-type and Jordan-type. An Elman network is a three-layer network (arranged horizontally as x, y, and z in the illustration), with the addition of a set of "context units", which is the u in the illustration below. The hidden layer is connected to these context units fixed with a weight of one. At each time step, the input is feedforward and then a learning rule is applied. The fixed back connections save a copy of the previous values of the hidden units in the context units, since they propagate over the connections before the learning rule is applied. Thus, the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multilayer perceptron. Precisely, dynamics of the Elman-type RNN can be represented mathematically by

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

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:

$$y(t) = g(Wh(t)), g(z_m) = \frac{e^{z_m}}{\sum_{i=1}^{k} e^{Z_k}}$$

The only difference between these two RNN architectures is the context nodes of Jordan-type RNN are fed from the output layer instead of from the hidden layer. The softmax layer can store and convey more information than the sigmoid layer as it contains more nodes, and the experiment results also confirm that the Jordan-RNN can offer slightly higher performance than the Elman-RNN.

However, RNN models usually require to be trained with a large amount of labeled data to achieve the expected performance. Inspired by the paper written by Peters et al. (2017), besides word embedding, we use a pre-trained language model to help encode the context of each token in the word sequence. Basically, a language model assigns a probability to the whole sequence:

$$p(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, ..., t_{k-1})$$

We also use word embedding, where words or phrases from the vocabulary are mapped to vectors of real numbers. The word embedding can learn semantic and syntactic information of the words, i.e. similar words are close to each other in this space and dissimilar words far apart. These can be learned either using large amount of text like Wikipedia or specifically for a given problem. And previous work (Mesnil, G., He, X., Deng, L., Bengio, Y., 2013) confirms that with pre-trained fine-tuning word vectors from SENNA brings out better performance. In this paper, we follow the ideas of RNN models combining with word embeddings, and further explore the sequence label performance of bidirectional LSTM and language models.

3 RNN with Language Model Embedding

Overview

In light of the success of RNNs in language modeling and many other natural language processing tasks, RNNs were introduced to solve slot filling problem which unsurprisingly achieved the state-of-the-art performance (Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ..., Zweig, G., 2015). But every coin has two sides. RNNs are usually data thirsty which means they need to be trained with a large amount of data to achieve the expected performance. This section will talk about how to alleviate this shortcoming of RNNs with pre-trained language model embedding.

Baseline LSTM Model

Our baseline model is basically the same structure as described in Mesnil et al. (2015) paper. But we replaced the simple Jordan and Elman versions of

RNN with a modern two-layer LSTM.

$$
\begin{aligned}
f_k &= \sigma(W_f \cdot [h_{k-1}, x_k] + b_f) \\
i_k &= \sigma(W_i \cdot [h_{k-1}, x_k] + b_i) \\
o_k &= \sigma(W_o \cdot [h_{k-1}, x_k] + b_o) \\
\widetilde{C_k} &= \tanh(W_c \cdot [h_{k-1}, x_k] + b_c) \\
C_k &= f_k * C_{k-1} + i_k * \widetilde{C_k} \\
h_k &= o_k * \tanh(C_k)
\end{aligned}
$$

We will briefly refer LSTM as $h_k = H(h_{k-1}, x_k)$ in the subsequent content. Our baseline LSTM model can be described as below.

$$
\begin{aligned}
x_k &= w_k = E(t_k) \\
h_k &= H(h_{k-1}, x_k) \\
y_k &= softmax(W_y \cdot h_k + b_y)
\end{aligned}
$$

where $E()$ represents word embedding.

LSTM Model with Pre-trained Language Model Embedding

Nowadays, using pre-trained word embedding such as Word2Vec or GloVe is quite popular in some NLP tasks. Instead of initializing the word embedding with random values, pre-trained word embedding can capture useful semantic and syntactic information learned from another large dataset. However, for many NLP tasks such as slot filling, besides the meaning of a word, it's also significant to represent the word in context. The state-of-the-art method (our baseline model) relies on the RNN model to encode the token sequences into a context sensitive representation, which requires additional labeled data (Peters, M. E., Ammar, W., Bhagavatula, C., Power, R., 2017). Inspired by this paper in 2017, we implemented a LSTM model with language model embeddings pre-trained on One Billion Word Benchmark (Jozefowicz, Rafal and Vinyals, Oriol and Schuster, Mike and Shazeer, Noam and Wu, Yonghui, 2016). One Billion Word Benchmark contains one billion words and a vocabulary of about 800K words. We will refer to this pre-trained language model as CNN-BIG-LSTM in the later content. As illustrated in Figure 2, the input sentence will be fed into both the GloVe and the language model and the result embeddings will then be concatenated as the new input embedding for the downstream 2-layer LSTM model.

$$
\begin{aligned}
w_k &= E(t_k) \\
l_k &= LM(t_1, t_2, t_3, ..., t_k) \\
x_k &= [w_k, l_k] \\
h_k &= H(h_{k-1}, x_k) \\
y_k &= softmax(W_y \cdot h_k + b_y)
\end{aligned}
$$

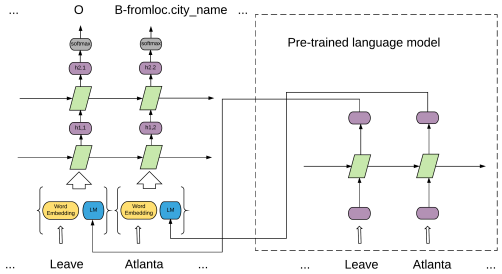

Figure 2: LSTM Model with Pre-trained Language Model Embedding

Other Models Implemented

Besides the models we talked above, we also implemented bidirectional LSTM with GloVe word embedding which is described as below:

$$
\begin{aligned}
w_k &= GloVe(t_k) \\
l_k &= LM(t_1, t_2, t_3, ..., t_k) \\
x_k &= [w_k, l_k] \\
\overrightarrow{h_k} &= \overrightarrow{H}(\overrightarrow{h_{k-1}}, x_k) \\
\overleftarrow{h_k} &= \overleftarrow{H}(\overleftarrow{h_{k+1}}, x_k) \\
y_k &= softmax(\overrightarrow{W_y} \cdot \overrightarrow{h_k} + \overleftarrow{W_y} \cdot \overleftarrow{h_k} + b_y)
\end{aligned}
$$

We choose to use bi-directional recurrent neural network (bi-RNN) because combining information of the succeeding words is proved to be important for slot filling task (Vu, T. N., Gupta, P., Adel, H., and Schutze, H., 2016). In bi-directional RNNs, words from both previous and future time step are regarded to predict the semantic tag of the target word. And GloVe can also provide a lot of useful addition semantic and syntactic information. We experimented all the combinations of these settings and next section will talk about our experiment result.

4 Experiments

Data

We report results on the widely used Airline Travel Information System (ATIS) dataset (C. Hemphill, J. Godfrey, and G. Doddington, 1990). Words in the dataset are all labeled in a IOB format as shown in Figure 3:

WORD	LABEL
what	O
flights	O
leave	O
atlanta	B-fromloc.city_name
at	O
about	B-depart_time.time_relative
DIGIT	B-depart_time.time
in	O
the	O
afternoon	B-depart_time.period_of_day
and	O
arrive	O
in	O
san	B-toloc.city_name
francisco	I-toloc.city_name

Figure 3: Labeled data example in IOB format, where O denotes an artificial class for any words that do not have real slots

We have 3983 labeled sentences as training data, 995 labeled sentences as validation data, and 893 labeled sentences as testing data.

Model training

We first reimplemented the recurrent neural network architecture based on Mesnil et al. paper (Mesnil, G., He, X., Deng, L., Bengio, Y., 2013) with TensorFlow as our baseline model. Based on our baseline model, we experimented with 4 settings, and report F1 scores for each of the followings:

- **Baseline:** A forward RNN with LSTM cell model, where the GloVe and the language model embedding are not included, was trained with different sizes of training data.
- **LSTM + LM:** A forward RNN with LSTM cell model, where only the language model embedding is included, was trained with different sizes of training data.
- **Bi-LSTM + LM:** An bidirectional RNN with LSTM cell model, where only the language model embedding is included, was trained with different sizes of training data.
- **Bi-LSTM + LM + Glove**: An bidirectional RNN with LSTM cell model, where both the GloVe and the language model embedding are included, was trained with different sizes of training data.

For all experiments, during the training, we assign the pre-trained word embeddings to words in our dataset that can be found in the GloVe dictionary and randomly initiate the embeddings to the rest. We used the RMSPropOptimizer with a constant learning rate = 0.001 to reduce the cross entropy loss function. We also include dropout regularization to avoid the over fitting. We end training after 5 epochs, because the purpose of our experiments is to reduce the dataset size for training, and the F1 results after epochs is sufficient to justify. For each epoch, we shuffle the training sequence.

Results

In the experiment, we prepare four different sizes of training datasets, which are one dataset consists of 3983 sentences, one with 2983 sentences, one with 1983 sentences, and one with 983 sentences. We compare the performance of all of our models using these four datasets. The results listed in Figure 4 show that within each dataset, the performance is improved consistently by adding language model, bidirectional LSTM, and GloVe representation. Although, apparently, the overall performance decreases with the reducing dataset size, the results listed at the bottom of the Figure 4 reveal that adding language model significantly improve the performance when we have very small dataset. Simply after 5 epochs training, the Bi-LSTM + LM + GloVe model trained with the smallest dataset outperforms the baseline model trained with the largest dataset.

Moreover, we train our model with more epochs to converge. We also include two more datasets: one with 483 sentences and the other one with 183 sentences. The results shown in Figure 5 indicates that our Bi-LSTM + LM + GloVe model still has strong performance on small dataset after 40 training epochs, and our observation of the significant improvement of performance is consistent with previous experiment when training with small dataset.

DATA SIZE	MODEL	F1-SCORE
3983	Baseline	88.31
	LSTM + LM	89.45
	Bi-LSTM + LM	92.67
	Bi-LSTM + LM + GloVe	93.43
2983	Baseline	87.26
	LSTM + LM	89.91
	Bi-LSTM + LM	91.28
	Bi-LSTM + LM + GloVe	92.11
1983	Baseline	82.16
	LSTM + LM	87.98
	Bi-LSTM + LM	91.18
	Bi-LSTM + LM + GloVe	92.74
983	Baseline	67.59
	LSTM + LM	87.10
	Bi-LSTM + LM	87.86
	Bi-LSTM + LM + GloVe	91.07

Figure 4: F1-score (after 5 epochs) comparison between different model settings trained with different data size

DATA SIZE	MODEL	F1-SCORE
983	Baseline	91.67
	LSTM + LM + GloVe	92.28
	Bi-LSTM + LM + GloVe	94.59
483	Baseline	90.19
	LSTM + LM + GloVe	92.05
	Bi-LSTM + LM + GloVe	93.51
183	Baseline	83.38
	LSTM + LM + GloVe	90.09
	Bi-LSTM + LM + GloVe	92.61

Figure 5: F1-score (after 40 epochs) comparison between different model settings trained with different data size

5 Conclusion and Future Outlook

In this paper, we presented a bi-directional LSTM model with pre-trained word embedding GloVe as well as pre-trained language model embedding for slot filling task. This model significantly improves the recognition performance under the situation where we do not have enough training data. In other words, with smaller training dataset, we can still achieve almost the same performance of a LSTM model without pre-trained language model embedding.

One possible future work is to extend the traditional forward language model to a bidirectional language model, inspired by our experimental result that bi-directional LSTM outperforms single direction LSTM. Also to explore what other kind of general knowledge can be learned on a large unlabeled corpus and embedded for a specific task use.

References

C. Hemphill, J. Godfrey, and G. Doddington. 1990. The atis spoken language systems pilot corpus. *in Proc. of the DARPA speech and natural language workshop*.

Hakkani-Tr, D., Tr, G., Celikyilmaz, A., Chen, Y. N., Gao, J., Deng, L., Wang, Y. Y. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. *Interspeech*, pages 715–719.

Jozefowicz, Rafal and Vinyals, Oriol and Schuster, Mike and Shazeer, Noam and Wu, Yonghui. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv*, 1602.02410.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., ..., Zweig, G. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.

Mesnil, G., He, X., Deng, L., Bengio, Y. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. *Interspeech*, pages 3371–3775.

Peters, M. E., Ammar, W., Bhagavatula, C., Power, R. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv*, 1705.00108.

Vu, T. N., Gupta, P., Adel, H., and Schutze, H. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. *In Proc. ICASSP*.