Tag Insertion Using Neural Networks

Liangyou Li[*], Alexander Raginsky[+]
[*] ADAPT Centre, Dublin City University
[+] Pangeanic, Spain

July 2016

This document describes a neural network architecture for inserting tags in a source sentence into its target sentence. We treat the tag insertion problem as a sequence labeling task and use recurrent neural networks to take the source sentence into consideration and at the same time label the target sentence. Experiments are conducted on a data set provided by Pangeanic.

1 Problem Definition

During a translation task, translators translate a source sentence into a target sentence. Source sentences frequently contain tags to define their styles, formation etc.. In the translation, translators need to insert those tags in appropriate positions.

1.1 Tags

There are two kinds of tags: self closing tag and open tag. An example of both kinds of tags is shown in Table 1.

Open Tag	
Source:	Once the problem has been eliminated, press the button **<T>** 7 **</T>** to restart the programme.
Target:	Cuando se haya solucionado el problema, pulse la tecla **<T>** 7 **</T>** para reiniciar el programa.
Self-closing Tag	
Source:	Turn the selector dial to **<T/>** and then in the new selected programme.
Target:	Gire el selector de programas hasta la posición **<T/>** y, seguidamente, hasta el nuevo programa seleccionado.

Table 1: Example of tags

1.2 Sequence Labeling

We treat the tag insertion problem as a sequence labeling task and use the standard BIO labeling format. Table 2 shows a BIO format for the example in Table 1.

There are cases where multiple tags exist in a single sentence. We treat each tag independently and create BIO instances for each of them and ignore the type of tags. Table 3 shows an example.

Open Tag:	
Source:	Once/**O** the/**O** problem/**O** has/**O** been/**O** eliminated,/**O** press/**O** the/**O** button/**O** 7/**B-T** to/**O** restart/**O** the/**O** programme./**O**
Target:	Cuando/**O** se/**O** haya/**O** solucionado/**O** el/**O** problema,/**O** pulse/**O** la/**O** tecla/**O** 7/**B-T** para/**O** reiniciar/**O** el/**O** programa./**O**
Self-closing Tag:	
Source:	Turn/**O** the/**O** selector/**O** dial/**O** to/**B-T** and/**I-T** then/**O** in/**O** the/**O** new/**O** selected/**O** programme./**O**
Target:	Gire/**O** el/**O** selector/**O** de/**O** programas/**O** hasta/**O** la/**O** posición/**B-T** y,/**I-T** seguidamente,/**O** hasta/**O** el/**O** nuevo/**O** programa/**O** seleccionado./**O**

Table 2: Example of BIO format

Source:	<T1> a b <T2> c d e f </T2><T1/> g h
Target:	<T1><T2> c d e f </T2> a b <T1/> g h
BIO instance T1:	
Source:	a/**B-T** b/**I-T** c/**I-T** d/**I-T** e/**I-T** f/**I-T** g/**O** h/**O**
Target:	c/**B-T** d/**I-T** e/**I-T** f/**I-T** a/**I-T** b/**I-T** g/**O** h/**O**
BIO instance T2:	
Source:	a/**O** b/**O** c/**B-T** d/**I-T** e/**I-T** f/**I-T** g/**O** h/**O**
Target:	c/**B-T** d/**I-T** e/**I-T** f/**I-T** a/**O** b/**O** g/**O** h/**O**

Table 3: Example of creating multiple BIO instances when multiple tags exist

2 Neural Network Architecture

The probability of a sequence of labels $y = y_1 \cdots y_m$ over a sequence of target words $t = t_1 \cdots t_m$ is given by:

$$p(y|s, sl, tw) = \prod_i p(y_i|s, sl, tw_1 \cdots tw_i)$$

where s are source words, sl are source labels, y are target labels, $tw_i = t_{i+w} \cdots t_i \cdots t_{i+w}$ is a sequence of target words in a fixed size $2w+1$ of window. So, when predicting each y_i, we takes context target words into consideration

Let $x_i = [t_{i-w}, \cdots, t_i, \cdots, t_{i+w}]$ is a concatenation of vectors of target words (Mesnil et al., 2013). $zx = zx_1 \cdots zx_n$ are representations of source words, $zy = zy_1 \cdots zy_n$ are representations of source labels. Each zx_j is a vector whose value while be learned during training. Each zy_j is a one-hot vector over all labels.

Figure 1 shows an architecture used in this document. Three Long Short-Term Memory (LSTM) networks are used. Two of them form a bi-directional LSTM (Bahdanau et al., 2014) used to learn new representations $[fh_i, bh_i]$ of source words and labels. An attention model is then used to compress source information into a vector a_i by a weighted summation $\oplus$ over $[fh_i, bh_i]$ (Bahdanau et al., 2014). At final, another LSTM is used to predict a label y_i for a target word t_i.

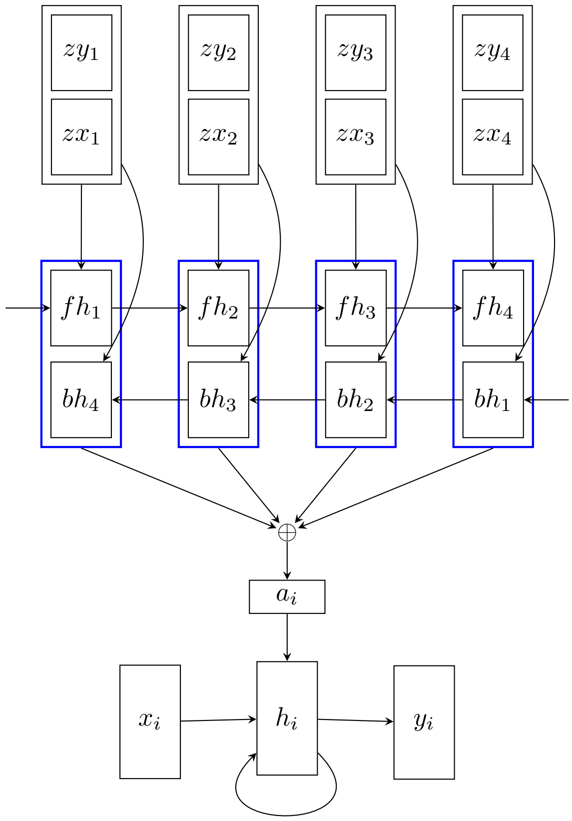

Figure 1: Neural network architecture

2.1 Long Short-Term Memory

Let x_i are input vectors to LSTM. Given previous hidden vector h_{i-1} and memory vector c_{i-1}, the next hidden h_i and memory c_i can be calculated by:

$$\begin{aligned}
I_i &= sigmoid(W_I x_i + U_I h_{i-1} + b_I) \\
F_i &= sigmoid(W_F x_i + U_F h_{i-1} + b_F) \\
\overline{C}_i &= tanh(W_C x_i + U_C h_{i-1} + b_C) \\
c_i &= I_i * \overline{C}_i + F_i * C_{i-1} \\
O_i &= sigmoid(W_O x_i + U_O h_{i-1} + b_O) \\
h_i &= O_i * tanh(c_i)
\end{aligned}$$

where I_i, F_i, and O_i are input, forget, and output gates, respectively. Since we usually focus on h_i, so we define $h_i = \text{LSTM}(x_i, h_{i-1})$

- **Source** LSTMs: Let $z_j = [zx_j, zy_j]$ is a concatenation of zx_j and zy_j. Then:

$$\begin{aligned}
fh_j &= \text{LSTM}(z_j, fh_{j-1}) \\
bh_j &= \text{LSTM}(z_{j+1}, fh_{j-1})
\end{aligned}$$

- **Target** LSTM:

$$h_i = \text{LSTM}(x_i, h_{i-1}, a_i)$$

2.2 Attention Model

Let $zh_j = [fh_j, bh_{n-j+1}]$ is a concatenation of source hiddens fh_j and bh_{n-j+1}, n is the length of the source sentence. The attention vector a_i is a weighted summation over zh:

$$a_i = \sum_{j=1}^{n} \alpha_{ij} * zh_j$$

where α_i is vector representing a probability distribution:

$$\begin{aligned} \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \\ e_{ij} &= W_e tanh(zh_j + p_i) + b_e \\ p_i &= W_h h_{i-1} + W_x x_i + W_a a_{i-1} \end{aligned}$$

3 Experiments

Data is provided by Pangeanic. Table 4 shows statistics of our data set.

EN–ES	Train	Valid	Test
Open Tag	5,146	1,000	1,000
Self-closing Tag	1,242	200	200

Table 4: The number of sentence pairs in our English–Spanish data

All words are placed by universal POS tags. Table 5 shows evaluation scores of sequence labeling.

		Precision	Recall	F1
Open Tag	Valid	54.54	59.62	56.96
	Test	53.17	60.22	56.48
Self-closing Tag	Valid	53.42	51.50	52.44
	Test	59.19	53.30	56.07
All Tag	Valid	48.58	53.77	51.04
	Test	49.45	55.89	52.47

Table 5: Evaluation results

4 Clues of Future Improvement

- Take previous target labels into consideration
- words + POS as inputs

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Mesnil, G., He, X., Deng, L., and Bengio, Y. (2013). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775.