Investigating context influence in character Level LSTM methods for Japanese Auto Punctuation

Qi Chen, Jianmin Wu, Tianhuang Su

EBG, Baidu Inc, Shenzhen, 518052

{chenqi21, wujianmin01, sutianhuang}@baidu.com

1. Introduction

Automatic speech recognition (ASR) systems for Latin languages generate a stream of words. For Asia language, ASR systems usually just produce a sequence of characters, like Chinese and Japanese. Both kinds of the above mentioned systems don't get punctuated directly. However, in many situations, people do need punctuation marks to help them understand better and faster. For instance, during the process of meeting minutes transcription and simultaneous interpretation, the accurate punctuation for ASR output can save lot efforts and help people concentrate on linguistic proofread.

So far, plenty of works have been done in auto punctuation. Before deep learning comes to trend, the most popular methods can be addressed as: 1. Introduce punctuations during N-gram language model training and predict it in the same ways as words. 2. Combine lexical features and prosodic features (pause duration, etc.) [12], and do classification with machine learning algorithms. 3. Treat the task as a sequence-labeling problem and solve it with conditional ransom fields (CRFs)[2].

Considering deep learning's high performance in other tasks, many researchers starts to apply it on auto punctuation. CNN based method that proposed by Che. X et al. [6] made use of context formatted pre-trained Word Vectors as input. They treated the punctuation task as a 4-class classification problem.

Tilk et al. [3] proposed a two-stage RNN model using long short-term memory (LSTM) [10] units, their first stage use only lexical features in large corpus, the second stage use both lexical feature and pause duration information. To include both the past and future observation, Xu K. et al. [4] adopted a multilayer Bidirectional LSTM (Bi-LSTM) framework in use of word Embedding features, and punctuate Chinese sentences with 3-class labels. Ballesteros M et al. [7] used character-based embedding and LSTM model to do punctuation prediction on full range of punctuation marks across languages. Tilk O et al. [5] also improved their model by introducing an Attention Mechanism [9] to their Bi-GRU models.

Almost all above models are trained on languages that do not need word segmentation. [4] predicts Chinese punctuations but with segmented corpus processed by Mecab[1]. Inspired by [7,13], we base our models on character embedding for Japanese punctuation. Since Japanese does not have word boundary and have introduced many foreign words, using word level embedding can cause huge vocabulary and sparse model. In addition, we investigated different ways of utilizing context information. Our results shows using local attention information in Bi-LSTM can elevate the performance. The model has been deployed in our IME product (Simeji voice input), and serves our users.

2. Methods

All of our models take character embedding as input, and they are based on single layer LSTM [10] to explore the different ways of applying lexical context information. The task can be formulated as follows.

Given a character sequence with length T:

$$\vec{x} = [x_1, x_2, \ldots, x_T] \quad (1)$$

Transform to embedding with size N:

$$\vec{X} = [X_1, X_2, \ldots, X_T], X_i \in R^N \quad (2)$$

Predict label y_i after character x_i among K=5 labels, with model f_M:

$$y_i = arg \max_{k=1:K}(\sigma(\overrightarrow{W}^T f_M(\vec{X}) + \vec{b})) \quad (3)$$

Where $\sigma(\vec{z})$ is the softmax function:

$$\sigma(\vec{z})_k = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} \quad (4)$$

$$y_i \in \{N, C, P, Q, E\} \quad (5)$$

Here N is the label for no punctuation mark, C for comma “、”, P for period “。”, Q for question “？”, and E for exclamation “！”. In following sections, we would describe the models f_M in detail.

2.1. Future Context LSTM

The first model we use is a unidirectional LSTM with future context window. Unidirectional LSTM can only make use of past information, so we add a future window to include the future information. The model can be described as:

$$context(X_t) = [X_t, X_{t+1}, \ldots, X_{t+w}] \quad (6)$$

$$\vec{h}_t, \vec{C}_t = LSTM(context(X_t), \vec{h}_{t-1}, \vec{C}_{t-1}) \quad (7)$$

$$f_{M1}(\vec{X}) = [\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_T] \quad (8)$$

Here the hidden state of LSTM cell is $\vec{h}_t$ at time t, the cell state is $\vec{C}_t$, and $context(X_t)$ is the context feature with window size w. Padding should be done when the sliding future window exceeds the sequence length.

2.2. Bi-LSTM

Bi-LSTM can utilize both past information and future information at a specific time. We introduce the Bi-LSTM model for comparing the context influence. The model description is as follows:

$$\vec{h}_t, \vec{C}_t = LSTM(X_t, \vec{h}_{t-1}, \vec{C}_{t-1}) \quad (9)$$

$$\overleftarrow{h}_t, \overleftarrow{C}_t = LSTM(X_t, \overleftarrow{h}_{t-1}, \overleftarrow{C}_{t-1}) \quad (10)$$

$$\tilde{h}_t = [\vec{h}_t, \overleftarrow{h}_t] \quad (11)$$

$$f_{M2}(\vec{X}) = [\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_T] \quad (12)$$

$\vec{h}_t$ is the hidden state of the forward LSTM layer at time step t, $\overleftarrow{h}_t$ is computed the same way as $\vec{h}_t$ except the input X is processed in reverse order. We use two direction hidden feature $\tilde{h}_t$ for label prediction.

2.3 Bi-LSTM with local Attention

In [4], the author proved that deeper Bi-LSTM is useful for predicting comma and period while modeling the context of output punctuation labels with CRF would harm the performance, since auto punctuation is a highly imbalanced task and there exist weak relationships among the labels. This is explicable: the appearance of a comma in a sentence is not much relevant to other commas or periods. LSTM methods can make use of context information, but there also exists the label interference during the optimizing process due to the information flow between different time steps. The loss of one label can affect the other time step label prediction.

Inspired by this, we want make better use of the word context while decoupling the label context. Attention mechanism [9] can help address this problem. Our Bi-LSTM with local attention model is defined as:

$$s_t = \sum_{i=t-\frac{l_a+1}{2}}^{t+\frac{l_a+1}{2}} a(\tilde{h}_t, \tilde{h}_i)\tilde{h}_i \quad (13)$$

$$e(\tilde{h}_t, \tilde{h}_i) = v_a^T \tanh(W_a^T \tilde{h}_t + U_a^T \tilde{h}_i) \quad (14)$$

$$a(\tilde{h}_t, \tilde{h}_i) = \frac{e(\tilde{h}_t, \tilde{h}_i)}{\sum_{k=t-\frac{l_a-1}{2}}^{t+\frac{l_a-1}{2}} e(\tilde{h}_t, \tilde{h}_k)} \quad (15)$$

$$f_{M3}(\vec{X}) = [s_1, s_2, \ldots, s_T] \quad (16)$$

We use the current Bi-LSTM hidden state $\tilde{h}_t$ and all the hidden states fall in the window centered at t (window size l_a) to calculate their similarities $e(\tilde{h}_t, \tilde{h}_i)$. Then the alignment score $a(\tilde{h}_t, \tilde{h}_i)$ is calculated by softmax function. The finally context vector is the weighted average over the in-window hidden states. If the sliding window crosses the sentence boundary, we choose to ignore the outside part. The detailed structure of this model is presented in Figure 1. This mechanism can automatically search the most relevant features within the window size l_a for label prediction at time step t. It’s called local attention and was proposed in [11] for encoder decoder problem. Compared to global attention, it requires less memory and more computationally efficient.

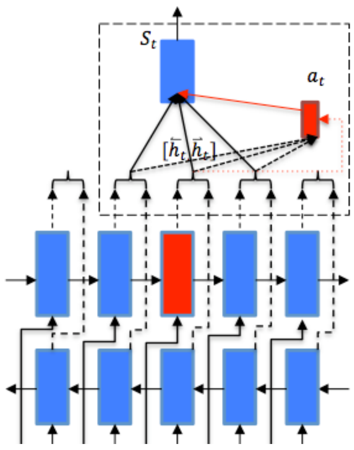

Figure1: Bi-LSTM with local Attention

[5] also used attention mechanism, but in a different way. They forward Bi-GRU hidden state to another GRU, and calculate the attention based on previous GRU hidden state and all Bi-GRU hidden states. Using another GRU cause their output labels still have interference with each other in time order.

3. Experiments

3.1 Data Preparation

Before training, we extract the high frequency characters as our vocabulary dictionary (size 6797). We add "[Alpha]", "Num" to our vocabulary, replace all English words with "Alpha" and all numbers with "Num" during preprocessing. This can avoid English words or numbers been split by mistake during prediction process. We also use "Unk" to present all unknown characters in sentences. Then we extract the character–label sequence pairs to form our data set.

Dimension	Train Set	Test Set
No. of sentences	23592151	10000
Average length	27.5	27.5
Characters	650009658	275484
Comma	29237800	12313
Period	18791497	7927
Question	1598499	697
Exclamation	3205905	1381

Table 1: Data Set details

We use the punctuated twitter data as train set and test set. The description of our data sets is detailed in Table 1. The train set is composed by 23.5M sentences, and the test set has10k sentences. The average length of sentences in both sets is 27.5. In our data sets, comma holds 4.5%, period holds 2.89%, question holds 0.25%, and exclamation holds 0.49%.

3.2 Evaluation metrics

We use conventional Precision, Recall, and F1-score as model evaluation metrics. For over-all metric, we removed the null options and focus on evaluating punctuation marks as [6] did. We also calculate precision, recall and F1-score for comma, period, question and exclamation respectively in similar way.

3.3 Results

Table 2 and Table 3 show our experiments results in detail. FC-LSTM-w represents the Future Context LSTM model and the adjacent number indicates the future window size. Bi-LSTM-A stands for Bi-LSTM with local attention model (attention window size 5). For comma prediction, Bi-LSTM-A method outperforms all the others. It improves 1.74% F1-score compared to Bi-LSTM, and 5.72% F1-score to FC-LSTM-w4. For question and exclamation, FC-LSTM-w2 performs best with highest recall and F1-score, but relatively lower precision. There is a reciprocal relationship among the results of question, exclamation and period, since they all indicate sentence ends and are always falsely transformed from one to the other. For end of sentence punctuations, we just need past information since they are the ends of the sentence. So FC-LSTM performs best. The future window can help distinguish them from comma. But for comma, the better representation of both directional can achieve better results, so Bi-LSTM-A performs best by introducing feature importance in attention window.

In addition, our new model prefers to transform questions and exclamations to periods, which decreases the period precision. In practice, it's acceptable. To explain the decrease of recall, we note that most question and exclamation sentences are longer than 10 while our attention window size is 5, after attention calculation, it cannot capture the information that is too far away from the center word. Experiments on attention window size can be explored in further research.

MODELS	COMMA			PERIOD			QUESTION		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
FC-LSTM-w1	73.64	53.59	62.04	83.54	96.28	89.46	69.46	47.63	56.51
FC-LSTM-w2	74.16	59.39	65.96	**83.77**	96.20	89.56	70.58	**49.21**	**57.99**
FC-LSTM-w4	74.22	61.88	67.49	83.61	96.39	89.55	69.12	47.20	56.10
Bi-LSTM	75.15	66.34	70.47	82.93	97.57	**89.65**	71.72	44.76	55.12
Bi-LSTM-A	**77.77**	**67.38**	**72.21**	79.17	**99.90**	88.34	**100.0**	0.14	0.29

Table 2: Comma, Period, Question results on Twitter Test set.

MODELS	EXCLAMATION			NO-PUNC			PUNC-AVER.		
	Pr.	Re.	F1	Pr.	Re.	F1	Pr.	Re.	F1
FC-LSTM-w1	54.40	15.21	23.77	97.77	99.06	98.41	77.91	66.19	71.58
FC-LSTM-w2	54.11	**16.22**	**24.96**	98.04	98.99	98.51	78.06	69.48	73.52
FC-LSTM-w4	54.31	15.06	23.58	98.16	98.95	98.55	77.96	70.78	74.20
Bi-LSTM	61.83	10.79	18.37	98.37	98.93	98.65	78.40	**73.32**	**75.77**
Bi-LSTM-A	**100.0**	0.14	0.29	**98.42**	**99.06**	**98.74**	**78.46**	72.69	75.47

Table 3: Exclamation, PUNC-AVER, No-Punc results on Twitter test set.

4. Conclusion

In this paper, we compared different ways of introducing context information in auto punctuation tasks. Our Bi-LSTM with local attention model performs best among all the models. Future study can be conducted on improve the attention method, for example, the similarity calculation, attention center word position selection.

5. References

[1] Kudo T. Mecab: Yet another part-of-speech and morphological analyzer[J]. http://mecab. sourceforge. net/, 2005.

[2]Y.Liu,A.Stolcke,E.Shriberg,andM.Harper,"Using conditional random fields for sentence boundary detection in speech," in Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005, pp. 451–458.

[3]Tilk O, Alumäe T. LSTM for punctuation restoration in speech transcripts[C]//Sixteenth annual conference of the international speech communication association. 2015.

[4] Xu K, Xie L, Yao K. Investigating LSTM for punctuation prediction[C]//Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium on. IEEE, 2016: 1-5.

[5]Tilk O, Alumäe T. Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration[C]//INTERSPEECH 2016:3047-3051

[6] Che X, Wang C, Yang H, et al. Punctuation Prediction for Unsegmented Transcript Based on Word Vector[C]//LREC. 2016.

[7] Ballesteros M, Wanner L. A neural network architecture for multilingual punctuation generation[C]// Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing; 2016 Nov. 1-5; Austin (TX, USA).[place unknown]: ACL; 2016. p. 1048-53. ACL (Association for Computational Linguistics), 2016.

[8]Ueffing N, Bisani M, Vozila P. Improved models for automatic punctuation prediction for spoken and written text[C]//INTERSPEECH. 2013: 3097-3101.

[9]D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine trans-lation by jointly learning to align and translate," ICLR2015, arXiv:1409.0473, 2015.

[10]Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[11]Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation[J]. arXiv preprint arXiv:1508.04025, 2015.

[12] J.-H. Kim and P. C. Woodland, "The use of prosody in a combined system for punctuation generation and speech recognition." in INTERSPEECH, 2001,pp. 2757–2760.

[13]Ballesteros M, Dyer C, Smith N A. Improved transition-based parsing by modeling characters instead of words with LSTMs[J]. arXiv preprint arXiv:1508.00657, 2015.