數位典藏與數位人文 3 期（2019 年 4 月）
Journal of Digital Archives and Digital Humanities No. 3, pp. 1-19 (April, 2019)
DOI:10.6853/DADH.201904_3.0001

# 一種自動標點的方法與實現

釋賢超[1,*]、方愷齊[2]、釋賢迥[3]、釋賢菊[4]、釋賢碉[5]、
釋賢繼[6]、釋賢大[7]、釋賢奉[8]、宋延淳[9]

## 摘　要

中文古籍通常沒有標點符號，給現代人閱讀和理解帶來極大困難。為古籍添加現代標點是古籍整理和研究的基礎，也是相當繁重的工作。借助人工智能（artificial intelligence, AI）實現古籍的自動標點具有現實意義。我們應用深度學習（deep learning, DL）在自然語言處理（natural language processing, NLP）領域的最新工具，在超過 5 千萬個漢字和約 1 千萬個標點組成的訓練集上，使用長短時記憶（long short-term memory, LSTM）和卷積神經網路（convolutional neural network, CNN）兩種模型進行訓練。然後在六種不同朝代佛教古籍文本的測試集上，實現了最高 94.3% 的標點正確率，可以為古文標注七種現代標點（逗號、句號、問號、嘆號、頓號、分號、冒號）。

**關鍵詞：**自動標點、古籍、長短時記憶、深度學習、自然語言處理

---

投稿日期：2019 年 1 月 20 日；通過日期：2019 年 3 月 28 日。
[1] 北京市海淀區龍泉寺藏經辦公室法師。
[2] 彩雲科技工程師。
[3] 北京市海淀區龍泉寺藏經辦公室沙彌。
[4] 北京市海淀區龍泉寺藏經辦公室沙彌。
[5] 北京市海淀區龍泉寺藏經辦公室法師。
[6] 北京市海淀區龍泉寺藏經辦公室法師。
[7] 北京市海淀區龍泉寺藏經辦公室法師。
[8] 北京市海淀區龍泉寺藏經辦公室法師。
[9] 西安須曼那科技發展有限公司高級工程師。
[*] 通訊作者：釋賢超，Email: xianchao@longquan.org

# 壹、概述

## 一、研究現狀

自動標點是指在非人工干預的情況下，根據特定算法給沒有現代標點的古籍文本自動標注現代中文標點的技術。這項研究的提出只有十餘年的時間。人工設計的規則庫（黃建年，2009）和條件隨機場方法（張開旭、夏雲慶、宇航，2009）都曾用於自動標點。利用循環神經網路（recurrent neural network, RNN）對古文進行自動斷句的工作，如採用基於門控循環單元（gated recurrent unit, GRU）的雙向循環神經網路（王博立、史曉東、蘇勁松，2017），或採用條件隨機場下的雙向長短時記憶神經網路（bidirectional long short-term memory with conditional random fields, Bi-LSTM-CRF）（Han, Wang, Zhang, Fu, & Liu, 2018），但是未實現添加現代標點。英文自動標點的研究有不少文獻報導（Kolář & Lamel, 2012），但與中文標點有所區別。

## 二、技術範疇

深度學習（deep learning, DL）已廣泛應用於自然語言處理（natural language processing, NLP），如機器翻譯、文本分類、機器問答、自動摘要等。自動標點也屬於 NLP 的應用範疇。本領域研究採用的主要模型架構有：基於循環神經網路的長短時記憶（long short-term memory, LSTM）（Sutskever, Vinyals, & Le, 2014），基於卷積神經網路（convolutional neural network, CNN）的序列到序列（sequence to sequence, Seq2Seq）模型（Gehring, Auli, Grangier, Yarats, & Dauphin, 2017）。

## 三、概念術語

本文中「斷句「和「標點」的含義有所區別。

### （一）斷句

斷句是指只標注句號，表示停頓。

### （二）標點

標點是指標注句號、逗號等多種現代標點符號，可以分為標號和點號兩類。

#### 1. 點號

點號表示語句中的語氣停頓，常見的點號有：句號（。）、逗號（，）、問號（？）、感嘆號（！）、頓號（、）、分號（；）和冒號（：）。

#### 2. 標號

標號表示語句中的特定成分，常見的標號有：雙引號（“ ”或『 』）、單引號（‘ ’或「 」）和書名號（《》）。

## 貳、模型原理

標點標注問題可以分為兩類：一是點號標注，二是標號標注。

### 一、點號標注

點號的特點是，在同一個位置上至多只能出現一個點號，即兩個點號不可同時出現在同一個位置。所以標注點號可以看作是由一長度為 $n$ 的文本序列 $T = (t_0, ..., t_{n-1})$ 生成一個具有同等字元長度的點號序列 $C = (c_0, ..., c_{n-1})$，點號序列中的第 $i$ 個元素 $c_{i-1}$ 所表示的是文本序列中第 $i$ 個元素 $t_{i-1}$ 後是否存在某種點號。如果存在，那麼 $c_{i-1}$ 便是七種點號中的某一種；如果不存在（即此處不斷句），便用一個自定義的常量表示（可看作表示不斷句的第八種點號）。從文本序列生成點號序列是一個典型的 N vs. N 問題（即輸入序列與輸出序列等長），採用 RNN 的經典結構及其變種（如 LSTM 模型）進行處理是一種比較合理的選擇。

經典的 RNN 由公式 (1)、(2) 表示，

$$h_i = f(\mathbf{W}^{ht} t_i + \mathbf{W}^{hh} h_{i-1}) \quad (1)$$

$$c_i = g(\mathbf{W}^{ch} h_i) \quad (2)$$

其中 $t_0, ..., t_{n-1}$ 為輸入序列，$c_0, ..., c_{n-1}$ 為輸出序列，$h_0, ..., h_{n-1}$ 為隱狀態，$f$、$g$ 是激活函數，用於控制輸出的大小，常見的激活函數有 tanh、sigmoid 等等；每個步驟的 $\mathbf{W}^{ht}$、$\mathbf{W}^{hh}$、$\mathbf{W}^{ch}$ 都是全局共享（即不隨 $i$ 的變化而改變）的線性變換矩陣。自動標點任務中，$g$ 是一個 softmax 激活函數，可以將隱狀態歸一化為輸出點號的概率。文本序列中每個元素的輸出都是八個概率值（總和為 1），對應七種點號和「不斷句」的概率。

RNN 的經典結構可以用圖 1 表示。RNN 的特點是每個時刻接受一個輸入向量，產生一個輸出向量，輸入與輸出之間存在至少一個隱藏層，每個時刻的輸出向量取決於當下的隱藏層狀態，每個時刻的隱藏層狀態則受上一時刻的隱藏層狀態和當前輸入向量的共同影響。RNN 可以處理任意長度的文本序列。由於 RNN 中每個時刻的狀態都受到上一時刻的影響，無法通過並行處理來加快訓練或推理的過程，所以運行效率比較低。因為梯度消失和梯度爆炸的問題，使用訓練 RNN 一般不夠穩定。

LSTM 是一種時間遞迴神經網路。基於 LSTM 的 RNN 引入了在相鄰時間步之間的隱狀態傳遞（即後文所提到的細胞狀態 $P$），同時加入的

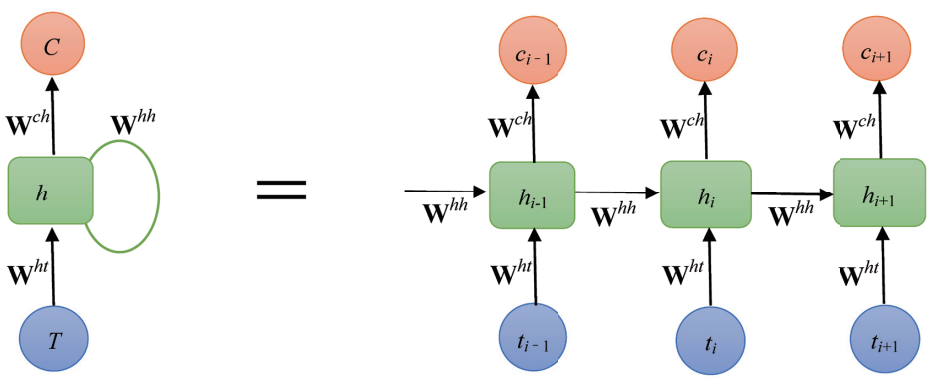


圖 1　RNN 經典結構示意圖

資料來源：作者自製。

門控單元和 tanh 激活函數能很好地調整輸入輸出的大小，所以 LSTM 可以比較好地解決古典 RNN 方法難以應對的長時信息訓練問題。例如，最近 Khandelwal、He、Qi 與 Jurafsky（2018）發現 LSTM 在生成過程中平均可以利用到當前時間步前 200 步的信息。LSTM 的原理是由上一時刻的隱狀態和當前輸入值，分別計算遺忘門 $f$（forget gate）、輸入門 $r$（input/remember gate）和輸出門 $o$（output gate）的值，控制細胞狀態 $P$ 的信息遺忘、記憶和輸出，從而獲得當前時刻輸出值（Hochreiter & Schmidhuber, 1997）。計算過程分五步，用公式 (3) 至公式 (8) 表示。

（一）根據前一時刻的隱狀態 $h_{i-1}$ 和當前輸入 $t_i$，計算遺忘門值 $f_i$、輸入門值 $r_i$、輸出門值 $o_i$：

$$f_i = \sigma(W_f \times [h_{i-1}, t_i] + b_f) \tag{3}$$

$$r_i = \sigma(W_r \times [h_{i-1}, t_i] + b_r) \tag{4}$$

$$o_i = \sigma(W_o \times [h_{i-1}, t_i] + b_o) \tag{5}$$

（二）根據前一時刻的隱狀態 $h_{i-1}$ 和當前輸入 $t_i$，計算臨時細胞狀態 $\tilde{P}_i$；

$$\tilde{P}_i = tanh(W_P \times [h_{i-1}, t_i] + b_P) \tag{6}$$

（三）根據輸入門值 $r_i$、遺忘門值 $f_i$、臨時細胞狀態 $\tilde{P}_i$ 和上一刻細胞狀態 $P_{i-1}$，計算當前時刻細胞狀態 $P_i$；

$$P_i = f_i \times P_{i-1} + r_i \times \tilde{P}_{i-1} \tag{7}$$

（四）根據當前時刻細胞狀態 $P_i$ 和輸出門值 $o_i$，得到隱狀態 $h_i$。

$$h_i = o_i \times tanh(P_i) \tag{8}$$

（五）從隱狀態 $h_i$ 獲得當前輸出 $c_i$ 的方法，同公式 (2)。

其中，σ 是激活函數。每個步驟的 $W_f$、$W_r$、$W_o$、$b_f$、$b_r$、$b_o$ 都是全局共享的線性變換矩陣。

本文使用了多層雙向 LSTM（Bi-LSTM）網路，每一層由兩個不共享參數的 LSTM 組成，一個 LSTM 處理正向序列 $T = (t_0, ..., t_{n-1})$，另一個 LSTM 處理反向序列 T_rev = $(t_{n-1}, ..., t_0)$。第 $t_i$ 個元素輸出的隱狀態是由這兩個 LSTM 網路在此第 $i$ 步的輸出相加得到。

## 二、標號標注

標號的特點是成對出現。另外，標號有可能與點號出現在同一個位置，也有可能多個標號出現在同一個位置。比如，下面這個句子：

我問他做什麼，他說：「我在讀《史記》。」

在「說」和「我」之間出現了冒號（：）和引號（「）兩個標點，其中一個點號，一個是標號。在「記」的後面則存在書名號（》）、句號（。）和引號（」）三個標點，其中一個是點號，兩個是標號。

當存在標號的情況下，文本序列 T 的長度通常不等於標號序列 C 的長度，$t_i$ 與 $c_i$ 之間也不能一一對應，這是一個 N vs. M 問題（輸入序列與輸出序列長度不等），採用 Seq2Seq 模型進行處理是比較合適的。其結構示意圖見圖 2。

Seq2Seq 的原理是將輸入序列編碼為一個背景向量（context vector），再從背景向量解碼為目標序列，從而實現不定長序列的轉換（Sutskever et al., 2014）。編碼和解碼的過程通常採用 RNN 模塊實現，其中用於編碼的稱之為編碼器（encoder），用於解碼的模塊被稱為解碼器（decoder）。

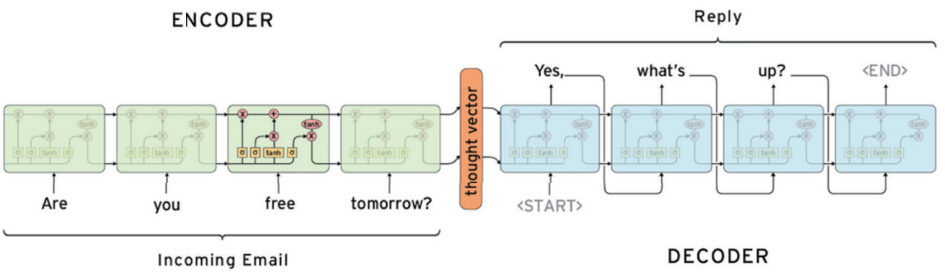


圖 2　Seq2Seq 結構示意圖

資料來源：Ivanov (2016)。

代表性的工作有 Facebook 發布的 fairseq 模型（https://github.com/pytorch/fairseq），因其採用 CNN 模塊實現編碼和解碼的過程，訓練和推理速度都大幅優於 RNN 模塊。

在編碼器—解碼器（encoder–decoder）結構中，輸入序列的信息被全部壓縮在一個背景向量之中。不過背景向量的長度是有限的，背景向量的長度是提升序列轉換準確度的主要瓶頸。為解決這個問題，允許背景向量在產生目標序列的時候可以隨著時刻的不同而發生變化，改進後的結構被稱為注意力機制（attention）（Luong, Pham, & Manning, 2015）。代表性的工作有 Google 提出的 Transformer 模型架構（Vaswani et al., 2017），在諸多 NLP 任務中實現了最佳結果。

N vs. N 問題是 N vs. M 問題的一個特例，所以 Seq2Seq 也可以用於標注點號。下面便運用了基於 CNN 模塊的 fairseq 開源架構來標注點號，作為 LSTM 模型的對照。

## 參、數據集

用於標點模型訓練的數據集，來自通過網路獲取的《中華電子佛典集成》（Chinese Buddhist Electronic Text Association [CBETA], http://cbeta.org）和《全唐文》（https://zh.wikisource.org/wiki/ 全唐文），以及通過軟體獲得的《佛光大藏經》的文本內容。訓練數據集總量約 5,400 萬字的古文和 1,190 萬個標點（只含點號，不含標號）。

測試數據集則是從 CBETA 中抽取了六段來自不同朝代未加現代標點

的文本。每段有 600 至 1,000 左右個漢字。測試數據集偏小，是因為製作測試數據的工作量較大。測試數據集都經過人工干預，以便允許存在多種合理結果。

數據集的構建主要步驟：

一、標點文獻採集，主要通過開源的古籍數據收集和自行標點來完成的；

二、電腦分類整理，通過軟體程序設計對各類文獻進行分類、篩選和存儲；

三、標點校對審核，主要對電腦分選過的數據再次進行人工核查；

四、數據格式標準化，則是通過電腦將人工核查確認的數據通過軟體程序設計進一步處理，將數據中多餘的符號標注等全部去掉，只保留標點和文字，並轉換成規範的自定義數據格式。

本數據集可以用於點號標注，不能用於標號標注。

## 肆、模型比較

### 一、模型訓練

這裡使用的 LSTM 模型，是帶有殘差網路（He, Zhang, Ren, & Sun, 2016）的六層 Bi-LSTM 架構。該 LSTM 架構的隱狀態單元數為 512。計算過程是：首先，通過編碼模塊將每個漢字映射為一個長度為 512 的字向量。隨後，將字向量分別輸入到兩個方向相反的 LSTM 網路（即 Bi-LSTM），把每個字上得到的兩個輸出進行求和，再作為下一層的 Bi-LSTM 網路的輸入。這裡使用 0.1 的隨機失活（dropout）對每層 Bi-LSTM 網路的輸入進行處理（Zaremba, Sutskever, & Vinyals, 2014），隨後進行一個大小調整（scale）的操作以調節隱狀態的大小。殘差網路的實現方法是將每層 Bi-LSTM 的輸入直接加到本層 Bi-LSTM 的輸出上。在對最後一層 Bi-LSTM 網路的輸出進行大小調整後，送給一個線性分類器，映射到每個標點上。再經過一個 softmax 運算，就可以得到每個標點的出現概率。以上模型是在基於 Pytorch 的語言模型框架 fairseq 上實現的。計算流程如圖 3 所示。

| | 是 | 故 | 大 | 慧 | 聖 | 智 | 三 | 相 | 當 | 勤 | 修 | 學 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| , | 0.00 | **0.97** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | **0.49** | 0.00 | 0.00 | 0.00 | 0.00 |
| 。 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | **0.95** |
| 、 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| : | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ? | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ; | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| ! | 0.00 | 0.00 | 0.00 | **1.00** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.05 |
| C | **1.00** | 0.03 | **1.00** | 0.00 | **1.00** | **1.00** | **1.00** | **0.51** | **1.00** | **1.00** | **1.00** | 0.00 |

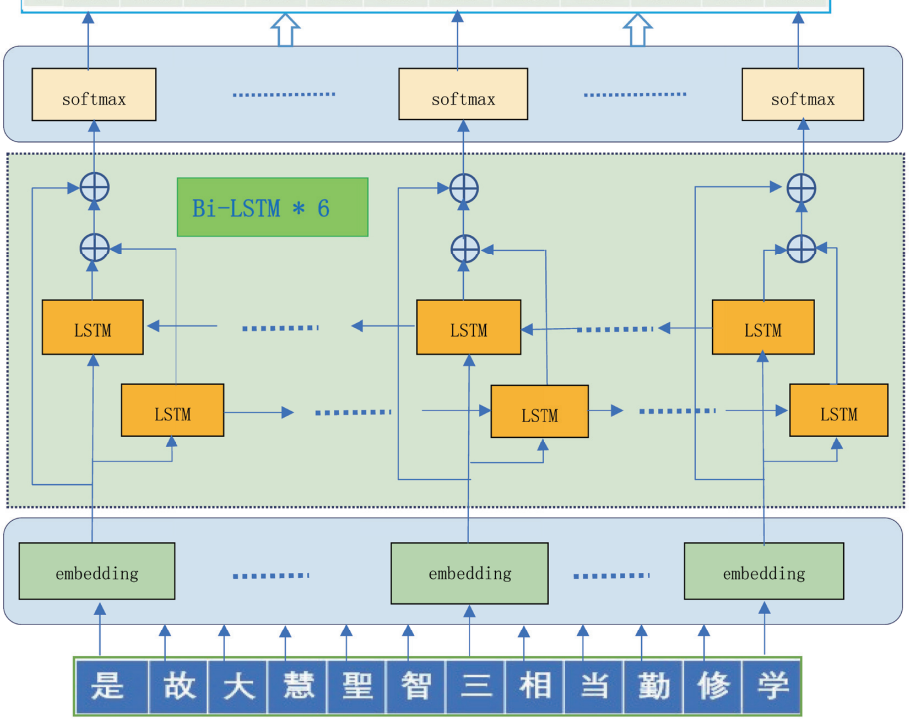


圖 3　LSTM 模型計算流程圖

資料來源：作者自製。

其中，概率表格中的 “C” 是表示不加標點的自定義常量。最後的標點結果是「是故，大慧！聖智三相當勤修學。」不過，在第八個位置上（即「相」與「當」之間），不加標點和出現逗號的概率相當接近，說明「是故，大慧！聖智三相，當勤修學。」也是一種可以接受的標點結果。

進一步分析可以發現，排除第八個位置，有八個位置上不加標點，最大概率平均值為 1；有三個位置有標點結果，最大概率平均值為 0.97，略低於不加標點的情況。從整體來看，不加標點時的最大概率平均值通常都

是大於有標點時的最大概率平均值。換句話說，標點模型不加標點的概率閾值應該高於加標點的概率閾值。所以，當不加標點的概率和加標點的概率完全相同的情況，應該更傾向於在這個位置上選擇加標點的可能。

這裡使用的 CNN 端到端模型為 fairseq 自帶的 CNN「英—德」（fconv_wmt_en_de）翻譯示例。

## 二、模型測試

測試數據集分別選取南北朝、隋朝、唐朝、宋朝、遼朝和明朝等不同時期的佛教經典文獻，分別用 LSTM 和 CNN 兩種模型進行訓練和測試。表 1 和圖 4 是兩個標點模型測試結果的對比。

這裡對常規的正確率算法進行了調整，以便按照更嚴格的標準評價自動標點的實際性能。考慮到一個總字數為 $M$ 的段落，每個字後面的位置要麼沒有標點，要麼是七種標點之一，因此設定每個位置的分值是 1 分，則滿分應為 $M$。如果該位置的標點正確，則得 1 分；否則，得 0 分。每個位置的得分總和稱之為常規分 $Q$，$Q/M$ 稱之為常規正確率。如果文本中一個標點都不加，根據以上算法，也能得到一定分數，將其叫做背景分 $N$。背景分的存在使得常規正確率的數值偏高，就算一個標點都不加，常規正備率也能達到 80% 至 90%，這樣就不能明顯區分不同標點結果的差異。所以，這裡採用扣除背景分影響的改進正確率 $q$ 作為評價指標，其計算公式是：

$$q = \frac{Q - N}{M - N} \tag{9}$$

**表 1　兩種模型標點正確率比較**

| 朝代 | 字數 | 有效標點數 | LSTM 模型正確率（%） | CNN 模型正確率（%） |
|---|---|---|---|---|
| 南北朝 | 1,019 | 190.5 | 74.0 | 57.2 |
| 隋朝 | 689 | 103.0 | 90.3 | 53.4 |
| 唐朝 | 1,020 | 194.0 | 94.3 | 69.1 |
| 宋朝 | 1,020 | 180.5 | 75.3 | 51.8 |
| 遼朝 | 694 | 127.0 | 75.2 | 43.3 |
| 明朝 | 1,014 | 169.0 | 65.7 | 46.7 |

資料來源：作者自製。

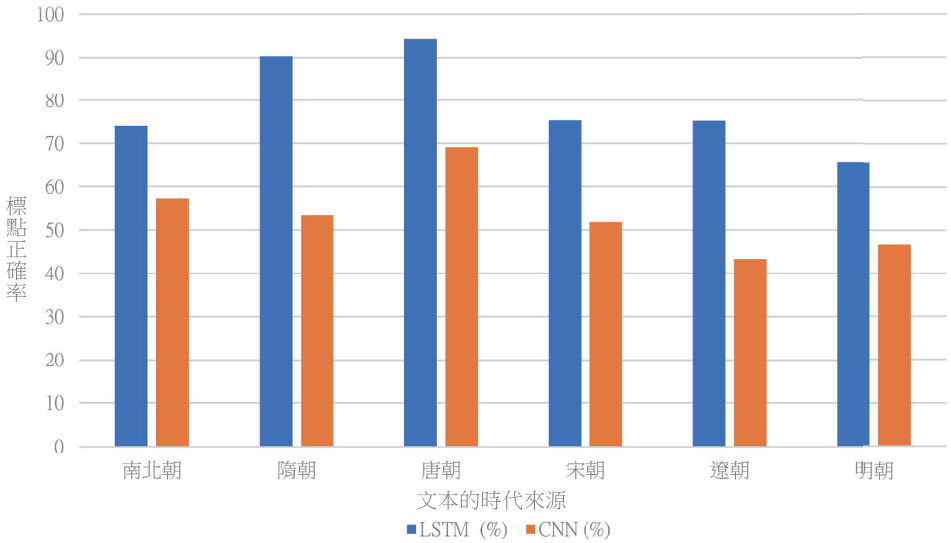


**圖 4　兩種模型標點正確率比較**

資料來源：作者自製。

比如，當總字數 $M$ 是 1,000，背景分 $N$ 是 800，如果常規分 $Q$ 是 900，那麼常規正確率 $Q/M$ 是 90%，改進正確率 $q$ 則只有 50%。

還考慮到每個位置可以存在多種合理的標點方式，比如當逗號或句號都合理的時候，那麼採用逗號或句號都可得到 1 分。在某些情況下，為了表示有些標點只是勉強合理，則設定為 0.5 分。

## 三、結果討論

結果表明，LSTM 模型的正確率在 65.7% 至 94.3% 之間，CNN 的正確率在 43.3% 至 69.1% 之間。LSTM 模型整體優於 CNN 模型，但二者趨勢基本一致。正確率最高的是唐代，隋朝次之，明代的正確率最低。不同朝代文獻標點的正確率差異，應該來自於訓練集數據分布的不平衡和品質的不統一。

實驗表明，LSTM 模型可以處理更長的文本序列，對 GPU 顯存要求更低，但因缺乏並行處理機制，計算速度更慢。CNN 模型可以進行並行處理，計算速度更快，但不能處理較長的文本序列，需要更大的GPU顯存。

不同架構的模型對於標點訓練的適應性及模型參數的優化策略等，有待進一步研究。

以上工作都是針對點號的標注，標號的標注仍需探索。

## 伍、結論

根據現代標點的特性，將自動標點轉化為 NLP 的 N vs.N 和 N vs. M 這兩個典型問題。規範化流程構建的數據集，為模型訓練和測試打下基礎。測試結果顯示，自動標點模型取得了最高 94.3% 的標點正確度。訓練集的結構調整和品質提升是改進標點模型性能的必要準備，使之適應各個歷史時期以及各種古籍類別。擴展測試集的數據規模，有助於提高標點性能評價的準確性和穩定性。

## 致謝

自動標點的訓練集和測試集取自《中華電子佛典集成》和《佛光大藏經》，工作也得到中華電子佛典協會和佛光山編藏處的鼓勵和支持，在此深表感謝。

## 參考文獻

王博立、史曉東、蘇勁松（2017）。一種基於循環神經網絡的古文斷句方法。*北京大學學報（自然科學版）*，*53*，255-261。doi:10.13209/j.0479-8023.2017.032

張開旭、夏雲慶、宇航（2009）。基於條件隨機場的古漢語自動斷句與標點方法。*清華大學學報（自然科學版）*，*49*，1733-1736。doi:10.16511/j.cnki.qhdxxb.2009.10.027

黃建年（2009）。*農業古籍的電腦斷句標點與分詞標引研究*（未出版之博士論文）。南京農業大學科學技術史系，南京，中國。

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. Retrieved from https://arxiv.org/abs/1705.03122

Han, X., Wang, H., Zhang, S., Fu, Q., & Liu, J. S. (2018). Sentence segmentation for classical Chinese based on LSTM with radical embedding. Retrieved from https://arxiv.org/abs/1810.03479

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778). doi:10.1109/CVPR.2016.90

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735-1780. doi:10.1162/neco.1997.9.8.1735

Ivanov, N. (2016). Tensorflow seq2seq chatbot. Retrieved from https://github.com/nicolas-ivanov/tf_seq2seq_chatbot

Khandelwal, U., He, H., Qi, P., & Jurafsky, D. (2018). Sharp nearby, fuzzy far away: How neural language models use context. Retrieved from https://arxiv.org/abs/1805.04623

Kolář, J., & Lamel, L. (2012). *Development and evaluation of automatic punctuation for French and English speech-to-text*. Paper presented at the 13th Annual Conference of the International Speech Communication Association 2012 (INTERSPEECH 2012). Portland, OR. Retrieved from https://www.isca-speech.org/archive/archive_papers/interspeech_2012/i12_1376.pdf

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. Retrieved from https://arxiv.

org/abs/1508.04025

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 3104-3112). San Diego, CA: Neural Information Processing Systems.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ...Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett. (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 5998-6008). San Diego, CA: Neural Information Processing Systems.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. Retrieved from https://arxiv.org/abs/1409.2329

# A Method and Implementation of Automatic Punctuation

Xianchao Shi[1,*], Kaiqi Fang[2], Xianjiong Shi[3], Xianju Shi[4], Xiandiao Shi[5], Xianji Shi[6], Xianda Shi[7], Xianfeng Shi[8], Yanchun Song[9]

## Abstract

Ancient Chinese scriptures usually have no punctuation marks, which makes it difficult for modern people to read and understand. Adding modern punctuation to ancient scriptures is the basis for the collation and research of ancient scriptures, however, it is a very tedious process. Therefore, it is of practical significance to realize automatic punctuation of ancient scriptures by means of artificial intelligence (AI). We apply the latest tool of deep learning (DL) in the field of natural language processing (NLP) to train the two models of long short-term memory (LSTM) and convolution neural network (CNN) on a training set of more than 50 million Chinese characters and approximately 10 million punctuations. Then, on the test set of Buddhist texts from six different dynasties, the highest punctuation accuracy of 94.3% was achieved. At present, the system can mark seven kinds of modern punctuations (comma, period, question mark, exclamation mark, dunhao, semicolon, colon) for ancient texts.

**Keywords:** automatic punctuation, ancient scriptures, long short-term memory (LSTM), deep learning, natural language processing

Manuscript received: January 20, 2019; Accepted: March 28, 2019
[1] Venerable, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[2] Engineer, ColorfulClouds Tech.
[3] Novice Monk, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[4] Novice Monk, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[5] Venerable, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[6] Venerable, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[7] Venerable, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[8] Venerable, Buddhist Canon Office, Longquan Monastery, Haidian Beijing.
[9] Senior Engineer, Xi'an Xumana Technology Development Co., Ltd.
[*] Email: xianchao@longquan.org

## 1. Introduction

Automatic punctuation refers to the research for marking modern Chinese punctuation on ancient scriptures without manual intervention. This research was initiated about ten years ago. Both the human-designed rules (Huang, 2009) and the conditional random field method (Zhang, Xia, & Yu, 2009) have been used for automatic punctuation. There are cases of using recurrent neural network to realize automatic segmentation of sentences on ancient texts. For example, one is the bi-directional recurrent neural networks based on Gated Recurrent Unit (Wang, Shi, & Su, 2017), the other is bi-directional long short-term memory neural networks under conditional random fields (Bi-LSTM-CRF) (Han, Wang, Zhang, Fu, & Liu, 2018). However, they did not realize automatic punctuation. There are many papers on automatic punctuation for English texts (Kolář & Lamel, 2012), however, the research on Chinese punctuation is a totally different story.

Deep Learning technique has been widely used in natural language processing (NLP), such as machine translation, text categorization, machine question and answer, and automatic summarization. Automatic punctuation belongs to the application of NLP. The main model architectures used in this field are long short-term memory (LSTM) model based on recurrent neural network (RNN) (Hochreiter, & Schmidhuber, 1997; Khandelwal, He, Qi, & Jurafsky, 2018; Sutskever, Vinyals, & Le, 2014) and sequence to sequence (Seq2Seq) model based on convolutional neural network (CNN) (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017; Luong, Pham, & Manning, 2015; Vaswani, et al., 2017).

In this paper, the principle of automatic punctuation, train set, model testing and evaluation are introduced. Finally, the improvement and further application of punctuation are put forward.

## 2. Principle

The LSTM model used in this paper is a six-layer Bi-LSTM with residual connection (He, Zhang, Ren, & Sun, 2016). The number of hidden units of the LSTM is 512. The calculation process is shown in figure 1.

First, each Chinese character is mapped into a word vector of length 512 by an encoding module. Subsequently, the word vectors are fed into the two LSTM networks (i.e., Bi-LSTM), and the two outputs obtained on each word are summed

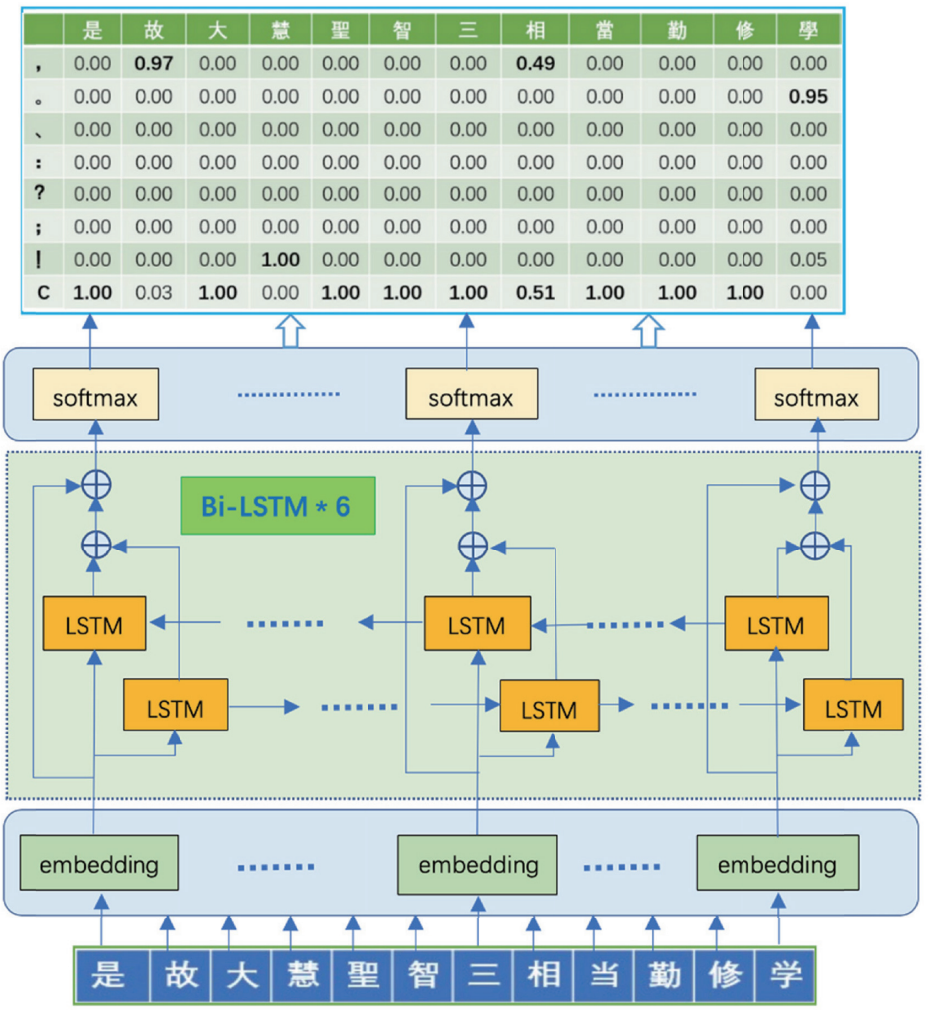


Figure 1. The Bi-LSTM model structure

Source: This study.

and used as inputs to the next layer of Bi-LSTM network. Here, the input of each layer of the Bi-LSTM network (Zaremba, Sutskever, & Vinyals, 2014) is processed, using a random dropout of probability 0.1, followed by a scale operation to adjust the value of the hidden state. The residual network is implemented by directly adding the input of the each layer of Bi-LSTM to the output of this layer of Bi-LSTM. After scaling the output of the last layer of the Bi-LSTM network, it is sent to a linear classifier that maps into each punctuation. After a softmax operation, the probability of occurrence of each punctuation can be obtained. The above model is implemented on fairseq, which is based on the Pytorch framework.

The CNN model used in this paper is the English-Germany translation example (fconv_wmt_en_de) that comes with fairseq.

## 2.1 Train Set

The train set used for punctuation model training comes from Chinese Buddhist Electronic Text Association (CBETA) and Quantangwen (i.e., *A Complete Collection of the Prose Works from Tang and Five Dynasties*) obtained through the network, as well as the text content of the Foguangzang obtained through software. The training dataset has a total of about 54 million characters and 11.9 million punctuations.

## 2.2 Model Evaluation

The test set is consisted of Buddhist texts selected from different periods in the Southern and Northern Dynasties, Sui Dynasty, Tang Dynasty, Song Dynasty, Liao Dynasty and Ming Dynasty. Each text has about 600–1000 Chinese characters. The test datasets were manually intervened to allow for a variety of reasonable results. LSTM and CNN models were used for training and testing. Table 1 is the comparison of the correct rate of punctuation between the two models.

The test results show that the accuracy rate of the LSTM model is between 65.7% and 94.3%, and the accuracy rate of CNN is between 43.3% and 69.1%. The LSTM model is superior to the CNN model overall, but the trends are basically the same. The highest accuracy rate was in the Tang Dynasty, followed by the Sui Dynasty, and the Ming Dynasty had the lowest accuracy rate. The difference in the correct rate of punctuation in different dynasty literatures

Table 1. Comparison of the correct rate of punctuation between the two models

| Dynasty | number of words | effective number of punctuation | correct rate for LSTM (%) | correct rate for CNN (%) |
|---|---|---|---|---|
| North and South | 1,019 | 190.5 | 74.0 | 57.2 |
| Sui Dynasty | 689 | 103.0 | 90.3 | 53.4 |
| Tang Dynasty | 1,020 | 194.0 | 94.3 | 69.1 |
| Song Dynasty | 1,020 | 180.5 | 75.3 | 51.8 |
| Liao Dynasty | 694 | 127.0 | 75.2 | 43.3 |
| Ming Dynasty | 1,014 | 169.0 | 65.7 | 46.7 |

Source: This study.

should come from the imbalance of the distribution of data in the train set and the inconsistency of quality. The adaptability of models with different architectures for punctuation training and optimization strategies for model parameters are for further study.

## 3. Conclusions

Given the characteristics of modern punctuation, automatic punctuation research can be transformed into two typical problems of N vs. N and N vs. M in natural language processing. The dataset built through standardized processes laid the foundation for model training and testing. The test results show that the automatic punctuation model achieved a maximum of 94.3% punctuation accuracy. In order to adapt to various historical periods and various ancient scriptures, the structural adjustment and quality improvement of the training set is necessary to improve the performance of the punctuation model. Extending the data size of the test set helps improve the accuracy and stability of automatic punctuation.