Ancient Chinese Punctuation via In-Context Learning

Jie Huang
NANJING UNIVERSITY
Jiangsu, China
huangjie@smail.nju.edu.cn

Abstract

EvaHan2024 focuses on sentence punctuation in ancient Chinese. Xunzi large language base model, which is specifically trained for ancient Chinese processing, is advised in the campaign. In general, we adopted the in-context learning (ICL) paradigm for this task and designed a post-processing scheme to ensure the standardability of final results. When constructing ICL prompts, we did feature extraction by LLM QA and selected demonstrations based on non-parametric metrics. We used Xunzi in two stages and neither did further training, so the model was generic and other fundamental abilities remained unaffected. Moreover, newly acquired training data can be directly utilized after identical feature extraction, showcasing the scalability of our system. As for the result, we achieved an F1-score of 67.7% on a complex test dataset consisting of multiple types of documents and 77.98% on Zuozhuan data.

Keywords: EvaHan2024, large language model, in-context learning

1. Introduction

Ancient Chinese texts typically consist of only characters without punctuation marks. So researchers in the field of ancient Chinese face the challenge of dealing with large amounts of unpunctuated text. Employing LLMs to do sentence punctuation will save significant manpower and facilitate subsequent research.

The prediction pipeline of our system is shown in figure 1. We take unpunctuated text as input and induce LLM to generate text with proper punctuation by in-context learning(ICL). To construct ICL prompt for each test input separately, we obtain the document category and sentence POS tag sequence by LLM QA, and then select texts with high similarity to the test input from training set. In addition, the generation mechanism of LLMs does not guarantee that the model will give fully standard result, which means the characters may not exactly consistent with the input. For the completeness of our system, we present an efficient and general post-processing scheme based on sequence matching implemented by dynamic programming.

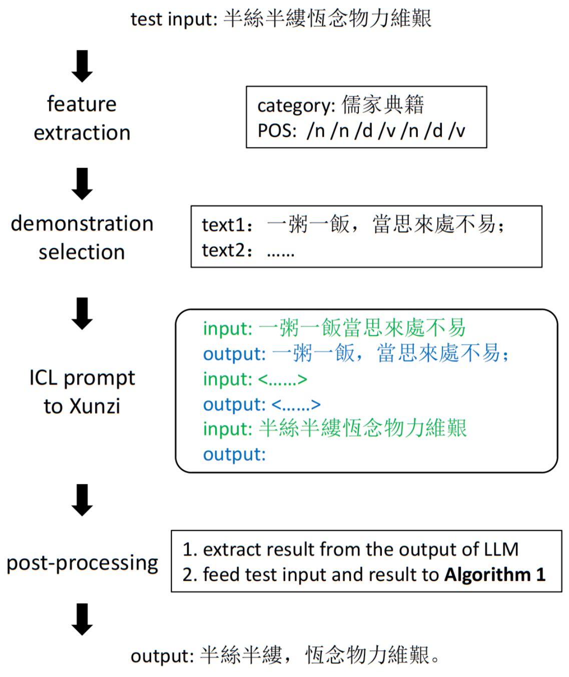
test input: 半絲半縷恆念物力維艱
feature extraction
category: 儒家典籍
POS: /n /n /d /v /n /d /v
demonstration selection
text1：一粥一飯，當思來處不易；
text2：……
ICL prompt to Xunzi
input: 一粥一飯當思來處不易
output: 一粥一飯，當思來處不易；
input: <……>
output: <……>
input: 半絲半縷恆念物力維艱
output:
post-processing
1. extract result from the output of LLM
2. feed test input and result to Algorithm 1
output: 半絲半縷，恆念物力維艱。

Figure 1: Prediction pipeline

2. Method

2.1. In-context learning

In-context learning(Dong et al., 2023) is a paradigm that allows LLM to learn tasks given only a few examples, which means, we can concatenate some pairs of input and output from the training set before the test input to help the general LLM perform a specific task better. The choice of examples may affect the performance significantly(Liu et al., 2022). In our system, we choose texts based on similarity. Concretely, we first obtained the category of the documents by LLM QA. Then among the documents of the same type as test input, we compute a similarity score between the POS tagging sequences of test and training text, which are also obtained by LLM QA. Texts with high scores will be used to construct the prompt for Xunzi.

Regarding classification, we predefined some common text categories to provide as options to the LLM. By limiting the scope of the answer, the validity of classification results is basically guaranteed.

Regarding POS tagging, due to the long output sequence, illegal tags and unlabeled results are prone to occur. In order to reduce this situation, we randomly selected a labeled result which contains most of the labels from the training set as a reference and add it to the QA prompt. The pos tagging result is in the form of " 半絲/n 半縷/n 恆/d 念/v 物力/n 維/d 艱/v", and we compute similarity score between the label sequences like [n,n,d,v,n,d,v].

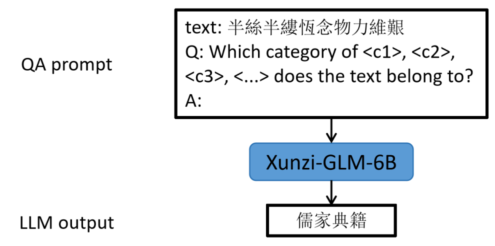
QA prompt
text: 半絲半縷恆念物力維艱
Q: Which category of <c1>, <c2>, <c3>, <...> does the text belong to?
A:
Xunzi-GLM-6B
LLM output
儒家典籍

Figure 2: classification model

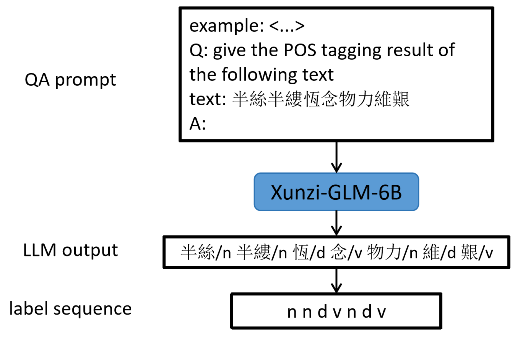
QA prompt
example: <...>
Q: give the POS tagging result of the following text
text: 半絲半縷恆念物力維艱
A:
Xunzi-GLM-6B
LLM output
半絲/n 半縷/n 恆/d 念/v 物力/n 維/d 艱/v
label sequence
n n d v n d v

Figure 3: POS tagging model

For brevity, we use <c1>, <c2>, <c3> in the diagram instead of concrete categories, which are shown in table 1. The randomly selected example in the QA prompt for POS tagging is alse omitted. In addition, since Xunzi is specifically finetuned for ancient Chinese, the actual QA prompts are all completely in Chinese, and the same is true for ICL prompts.

Since we hope the demonstration have strong structural similarity to the test input, at least locally, a score based on the longest common substring(LCS) is adopted. We add the latter term to give a relatively higher score to shorter sequences when the substring length is the same, for less redundant or confusing information, which may be significantly helpful for short text punctuation prediction.

$$l = LCS(POS_{train}, POS_{test})$$
$$s = l + l/LEN(POS_{train})$$

After we have determined the demostrations, the ICL prompt can be constructed as presented in figure1 and fed to the LLM. We remove the punctuation marks in the text to build demonstration inputs and use the original text as output. Then the test input is attached and we expect a well-punctuated output from the LLM, to be specific, Xunzi-GLM-6B in our system.

2.2. Post processing

The results given by the large model are not always completely standardized, such as the mixed use of traditional and simplified Chinese characters, missing and wrong characters, etc. However, we believe that most of the characters in the generated results are still consistent with the input. So we can try to correct the results to ensure that the final results are fully standardized while retaining effective punctuation as much as possible.

As shown in figure 4, we designed a general and efficient scheme. Firstly, the character-by-character alignment results of input (unpunctuated raw text) and output (prediction from LLM) are obtained by the dynamic programming matrix of the longest common subsequence algorithm. Then the aligned parts are kept and the remaining parts are filled with characters in the input and punctuations in the output. The error types can be counted while the final results are obtained. See the pseudocode in Algorithm 1 for details.

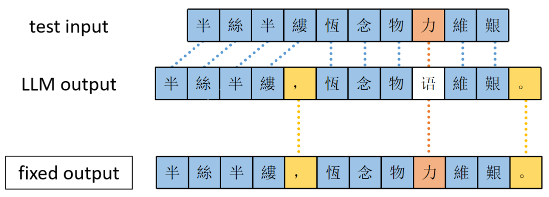
test input
半 絲 半 縷 恆 念 物 力 維 艱
LLM output
半 絲 半 縷 ， 恆 念 物 语 維 艱 。
fixed output
半 絲 半 縷 ， 恆 念 物 力 維 艱 。

Figure 4: Post processing

3. Experiments

Both the classification and POS tagging of the training set documents can be done in advance and the results are saved. In the testing phase, our pipeline is: 1. do classification and POS tagging for the test input; 2. select samples from training documents of same type; 3. construct ICL prompt for the LLM and get the prediction; 4. do post-processing and output the final result.

The experiment was carried out on two NVIDIA RTX A6000 with 48G memory. We have two test dataset. The first dataset consists of several documents of different categories, and we will refer to as test A. The second dataset is Zuozhuan, which is a history book, and we will refer to as test B.

Totally, a prediction of 310,207 tokens is finished in 6600 seconds. With an average of 47 tokens per second, the computational efficiency is acceptable.

Algorithm 1 Post-processing

```
Input: The test input, unpunctuated raw text, $R$; The prediction from llm, $H$; Punctuation mark list, $P$; Alignment result $A = [(i_1, j_1), \ldots, (i_n, j_n)]$
Output: A fixed result $F$ similar to $H$ but consistent with $R$;
 1: $pi = pj = 0$
 2: $F = empty\ string$
 3: for $m = 1 to n$ do
 4:     $i, j = A[m]$
 5:     while 1 do
 6:         $ri = R[pi], hj = H[pj]$
 7:         if $pi = i$ then
 8:             if $pj = j$ then
 9:                 $F = F + hj,\ pi++,\ pj++$
10:                 break
11:             else if $hj$ in $P$ then
12:                 $F = F + hj,\ pj++$
13:             else // redundant character
14:                 $pj++$
15:             end if
16:         else
17:             if $pj = j$ then // missing character
18:                 $F = F + ri,\ pi++$
19:             else if $hj$ in $P$ then
20:                 $F = F + hj,\ pj++$
21:             else // wrong character
22:                 $F = F + ri, pi++, pj++$
23:             end if
24:         end if
25:     end while
26: end for
27: while $pi < len(R)$ do
28:     $F = F + R[pi],\ pi++$
29: end while
30: while $pj < len(H)$ do
31:     $hj = H[pj]$
32:     if $hj$ in $P$ then
33:         $F = F + hj$
34:     end if
35:     $pj++$
36: end while
        return $F$
```

3.1. Data Information

We predefined 14 categories on the training dataset, and the number of documents and characters for each category are shown in the table 1. The following six categories were involved in the testing stage and star-marked in the table: Confucianism(儒家典籍), Buddhist sutra(佛教經文), Prose(散文雜記), History(歷史), Geography(地理), Agronomy(農學).

In fact, the categories may not cover all documents, and the accuracy of classification results given by LLM is difficult to verify due to the lack of expert knowledge. We believe that when the LLM gives the same classification label to two documents, at least for the model, some features of the pair are consistent, and it is more likely to be useful for our task.

type	docs	tokens
Confucianism*	14	5945471
Novel	25	4111359
Medical	35	3529444
History*	29	3343991
Criticism	36	1657880
Drama	3	1264962
Prose*	26	1030393
Taoist sutra	68	887175
Buddhist sutra*	8	427919
Geography*	4	370990
Poetry	5	220044
Astrology	3	189229
Art of war	6	166254
Agronomy*	4	34117

Table 1: statistical information of training data

The statistical information of the two test datasets is shown in table 2. Test A consists of several short books of different types, while Test B is a single long history book.

	docs	tokens
test A	6	50722
test B	1	199879

Table 2: statistical information of test data

3.2. Results

We get evaluation results of segmentation and punctuation. For segmentation, we treat all punctuation marks as a segment mark to compute the metrics. The baseline model is Xunzi-Qwen-7B-Chat.

	Precision	Recall	F1-Score
SEG	90.53%	66.12%	76.42%
PUNC	73.52%	52.22%	61.06%

Table 3: Test A, baseline(Xunzi-Qwen-7B-Chat)

	Precision	Recall	F1-Score
SEG	95.28%	87.17%	91.04%
PUNC	79.25%	72.09%	75.50%

Table 4: Test B, baseline(Xunzi-Qwen-7B-Chat)

Our system adopted Xunzi-GLM-6B as base model since it tends to generate relatively standard results. To verify the effectiveness of our strategy for selecting demonstrations, we conducted the following two sets of experiments for comparison. To ensure the standardability of the prediction, the outputs were all post-processed.

For experiment 1, when building the ICL prompt, we used BM2.5 to retrieve highly related text among documents of the same type. BM2.5 is a statistical method based on word frequency, which means it may have better efficiency but will ignore the sequence information of texts. From the results, the model works well with ICL and the simply constructed prompts on Test A, while has no positive effect on Test B, which seems to be easier from the baseline.

	Precision	Recall	F1-Score
SEG	91.54%	73.54%	81.56%
PUNC	74.52%	58.44%	65.51%

Table 5: Exp 1, Test A, Xunzi-GLM-6B, with ICL prompt build by classification (LLM QA) and retrieval (BM2.5)

	Precision	Recall	F1-Score
SEG	95.53%	86.56%	90.82%
PUNC	79.72%	72.18%	75.76%

Table 6: Exp 1, Test B, Xunzi-GLM-6B, with ICL prompt built by classification (LLM QA) and retrieval(BM2.5)

For experiment 2, We used exactly the same system as described in the former section. We further exploited the ability of the LLM to obtain POS tag sequences and designed a similarity metric for demonstration selection. From the results, we can see that with the higher level feature, the performance of the system is improved on both test sets.

	Precision	Recall	F1-Score
SEG	89.65%	79.49%	84.27%
PUNC	72.87%	63.25%	67.72%

Table 7: Test A, Exp 2, Xunzi-GLM-6B, with ICL prompt built by classification (LLM QA), POS tagging (LLM QA) and similarity

	Precision	Recall	F1-Score
SEG	95.38%	89.68%	92.44%
PUNC	80.44%	75.67%	77.98%

Table 8: Test B, Exp 2, Xunzi-GLM-6B, with ICL prompt built by classification (LLM QA), POS tagging (LLM QA) and similarity

Finally, we add a note on the role of post-processing in the system. We processed the output line by line. In Exp 2, Test B contains 3319 lines, of which 2828 lines were standard, and the remaining 491 lines were output after post-processing, accounting for about 15%. The proportion was even higher for Test A, with 293 standard lines among a total of 401 lines, 27% of the output were post-processed.

4. Discussion

For the detailed results, the performance of the system on different types of text does vary significantly. In test A, we achieved an F1 score of 61.61% for the Buddhist sutra(佛教经文) type, which is well below average(67.72%) and Test B(77.98%). But this type also shows the most significant improvement over baseline(55.73%). This indicates that the large language model itself is less capable of handling this type of text, which is related to the obscure language and unusual expression of Buddhist texts.

In general, we adopted ICL framework in our system and selected texts that are similar to the test input as demonstrations. We first narrow the range with categories and then perform fine-grained matching by POS sequences. The combination of content and structure features achieved good results. However, ICL demonstrations can also be selected based on other criterias, such as annotation difficulty(Drozdov et al., 2023) or the proportion of different punctuation marks in the text(Levy et al., 2023). Moreover, the order of examples can also affect the generation results(Lu et al., 2022).

Our system made use of the large language model's own knowledge and general ability to compensate for the lack of external domain knowledge. We used the features obtained by LLM QA to construct prompts for the same model, which is kind of accommodation to the model. In experiments with small models, we worry about error propagation between two stages, but for larger models, this potential consistency may tend to have a positive impact as the model becomes more powerful.

5. References

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Andrew Drozdov, Honglei Zhuang, Zhuyun Dai, Zhen Qin, Razieh Rahimi, Xuanhui Wang, Dana Alon, Mohit Iyyer, Andrew McCallum, Donald

Metzler, and Kai Hui. 2023. PaRaDe: Passage ranking using demonstrations with LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14242–14252, Singapore. Association for Computational Linguistics.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422, Toronto, Canada. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.