Transformersを用いた古典中国語(漢文)文切りモデルの製作

安岡孝一 (京都大学人文科学研究所附属東アジア人文情報学研究センター)

古典中国語 (漢文) は，単語と単語の間に区切りがなく，文と文の間にも区切りがない．これが，白文と呼ばれる古典中国語の書写形態であり，傍目には，漢字が連続的に並んでいるだけである．それゆえ，白文に対する言語処理は，単語と単語の間を区切る「単語切り」(word tokenization) と，文と文の間を区切る「文切り」(sentence segmentation) から始まる．本稿では，Transformers を用いた古典中国語の「文切り」手法と，それに伴う古典中国語「文切り」モデルの製作について述べる．これに加え，Transformers を用いた古典中国語の「単語切り」を，品詞付与と同時に行う手法についても述べる．

Sentence Segmentation of Classical Chinese Texts Using Transformers and BERT/RoBERTa Models

Koichi Yasuoka (Kyoto University)

Classical Chinese texts do not have any spaces or punctuations between words or between sentences. They consist of continuous strings of Chinese characters from the start to the end. The analysis of classical Chinese texts has to begin with finding out word-boundaries and sentence-boundaries, i.e. word tokenization and sentence segmentation. In this paper we discuss the sentence segmentation of classical Chinese texts using Transformers, which is designed for natural language processing with pre-trained language models such as BERT and RoBERTa. In the appendix we discuss the word tokenization (and Part-Of-Speech tagging) of classical Chinese texts using Transformers.

1　はじめに

筆者が班長を務める京都大学人文科学研究所共同研究班「古典中国語のコーパスの研究」(班員: Christian Wittern，守岡知彦，池田巧，山崎直樹，二階堂善弘，鈴木慎吾，師茂樹，白須裕之，藤田一乘) では，古典中国語 (漢文) の文法解析に精力を傾注しており，これまでに形態素解析・依存文法解析・直接構成鎖解析などの手法を研究してきた [1]．これらの古典中国語解析手法のうち，形態素解析は高い精度が得られている [2] が，依存文法解析や直接構成鎖解析については，白文を複数の文に分ける「文切り」がうまくいかず，結果として解析精度が下がってしまっていた [3]．

時おりしも，北京師範大学から『古詩文斷句』[4] が発表された．『古詩文斷句』は，古典中国語 BERT モデルによって文切りをおこなうもので，高い精度を誇っているらしい．ただし，このモデルは公開されておらず，デモ用 WWW ページ[a)]のみ利用できる．API も公開されていないため，われわれの解析モジュールに組み込むのは難がある．何とか，われわれ自身の手で，文切りモデルを製作できないだろうか．

2　系列ラベリングとしての文切り

白文「憲問恥子曰邦有道穀邦無道穀恥也克伐怨欲不行焉可以為仁矣子曰可以為難矣仁則吾不知也」を，「憲問恥」「子曰」「邦有道穀」「邦無道穀恥也」「克伐怨欲不行焉」「可以為仁矣」「子曰」「可以為難矣」「仁則吾不知也」の 9 つの文 [5] に，文切りすることを考えてみよう．

[6] のアルゴリズムは，古典中国語における文切りを，ニューラルネット上での系列ラベリング[b)]として解く，という点が特徴的である．具体的には，文頭の文字に対するラベルを B，文末の文字に対するラベルを E とし，B・M・E3・E2・E を用いて各文を表す (図 1)．ただし，1 文字のみで構成される文は，ラベルを S とする．なお，2 文字で構成される文のラベルは，B・E となる．3 文字で構成される文のラベルは，B・E2・E となる．4 文字で構成される文のラベルは，B・E3・E2・E となる．

Transformers [7] には，`run_ner.py` という系列ラベリング学習用プログラムが準備されている．これと，1 文字＝ 1 語であるような古典中国語モデルを

[a)] `https://seg.shenshen.wiki`
[b)] 品詞付与や固有表現抽出に用いられる．

憲 問 恥 子 曰 邦 有 道 穀 邦 無 道 穀 恥 也 克 伐 怨 欲 不 行 焉 可 以 為 仁 矣 子 曰 可 以 為 難 矣 仁 則 吾 不 知 也
B E2 E B E B E3 E2 E B M M E3 E2 E B M M M E3 E2 E B M E3 E2 E B E B M E3 E2 E B M M E3 E2 E

図 1: 系列ラベリングとしての文切りアルゴリズム

表 1: 古典中国語 (漢文) 文切りモデルの評価 (seqeval)

	評価 (evaluation)				テスト (predict)			
	accuracy	F1	precision	recall	accuracy	F1	precision	recall
RoBERTa-Classical-Chinese (base)	92.39	90.30	89.96	90.65	91.11	88.81	88.68	88.94
RoBERTa-Classical-Chinese (large)	92.36	90.15	90.03	90.28	90.60	88.24	87.97	88.51
SikuBERT [9]	90.64	87.85	87.64	88.06	89.05	86.30	85.72	86.89
SikuRoBERTa [9]	90.23	87.50	86.97	88.04	89.39	86.75	86.00	87.52
AnchiBERT [10]	87.39	83.78	83.30	84.27	85.22	81.71	81.48	81.93
GuwenBERT (large) [8]	86.17	82.62	82.24	83.00	84.56	80.96	80.46	81.46
GuwenBERT (base) [8]	84.87	80.73	80.30	81.16	83.00	79.06	78.75	79.37
『古詩文斷句』[4]	85.27	83.66	80.95	86.55	86.81	85.03	82.83	87.34

組み合わせれば，系列ラベリングとして文切りを実装[c]できる．われわれは，北京理工大学が製作した古典中国語モデル GuwenBERT [8] を拡張する形で，新たな古典中国語モデル RoBERTa-Classical-Chinese を製作し，文切りモデルの学習に用いることにした．

3 評価と考察

Transformers 4.9.2 と RoBERTa-Classical-Chinese を用いて，古典中国語 (漢文) の文切りモデルを製作した．訓練データには Universal Dependencies 2.8.1 の `lzh_kyoto-ud-train.conllu` を，評価データには `lzh_kyoto-ud-dev.conllu` を，テストデータには `lzh_kyoto-ud-test.conllu` を，それぞれ用いた．図 2 のスクリプトを用いて訓練 (文切りモデル構築) に要した時間は，3 epochs・1 GPU (NVIDIA GeForce RTX 2080) で，各モデル 5〜10 分程度だった．なお，比較のため，[8, 9, 10] の古典中国語モデルに対しても，文切りモデルを製作した．『古詩文斷句』に対しては，評価データとテストデータを手作業で入力した．

seqeval [d]による評価結果を，表 1 に示す．われわれの RoBERTa-Classical-Chinese による文切りモデルの精度が，かなり高いと言える．白文「憲問恥子曰邦有道穀邦無道穀恥也克伐怨欲不行焉可以為仁矣子曰可以為難矣仁則吾不知也」に対し，各モデルで文切りした結果を図 3 に示す．RoBERTa-Classical-Chinese は，この白文を完璧に文切りしている．

GuwenBERT の結果は，未知語 ([UNK]，図 3 ではゲタ記号「〓」で示す) がかなり多い．この原因としては，GuwenBERT のボキャブラリー (base・large ともに 23292 語) が簡化字で占められており，繁體字が未知語となってしまうためである．これに対し，われわれの RoBERTa-Classical-Chinese では，3026 語の繁體字 (および日本の常用漢字) を拡張して，ボキャブラリーを 26318 語としており，結果として未知語が発生していない．

一方で，SikuBERT・SikuRoBERTa・AnchiBERT のボキャブラリーは，いずれも 21128 語であり，四庫全書の異なり字数 28803 字 [9] より少ない．ボキャブラリーの中身を見てみると，「しにはとんとんワークケートを」[e]を含んでいるなど，BERT-base-Chinese [11] のボキャブラリーそのものである．つまり，SikuBERT・SikuRoBERTa・AnchiBERT のボキャブラリーは，古典中国語向けチューニングがなされていない．

これらの点を [8, 9, 10] の製作者にも伝えたところ，南京農業大学のグループが SikuBERT・SikuRoBERTa を改良し，ボキャブラリーに 8663 語を追加[f]したモ

[c] `run_ner.py` は，固有表現抽出 (named entity recognition) 用のプログラムであり，文切りへの適用には改造が必要だった．この改造は Transformers 4.6.0 以降，公式に組み込まれている．

[d] `https://pypi.org/project/seqeval`

[e] 「仕事探しにはどんどんワークゲートを」から漢字と濁点を除去した文字列だと考えられる．現代中国語や古典中国語の BERT/RoBERTa モデルに，なぜこんなボキャブラリーが含まれているのか，正直なところ筆者には謎である．

[f] 「しにはとんとんワークケートを」を含め，BERT-base-Chinese 由来の 21128 語を残したまま，新たに漢字 8663 字種を 1 文字＝1 語の形で追加し，ボキャブラリーを 29791 語に拡張している．

```
#! /bin/sh
BERT=KoichiYasuoka/roberta-classical-chinese-base-char

pip3 install transformers==4.9.2 datasets seqeval
test -d UD_Classical_Chinese-Kyoto ||
 git clone -b r2.8 https://github.com/universaldependencies/UD_Classical_Chinese-Kyoto
test -f run_ner.py ||
 curl -LO https://raw.githubusercontent.com/huggingface/transformers/v‘pip3 list |
 sed -n ’s/^transformers *\([^ ]*\) *$/\1/p’‘/examples/pytorch/token-classification/run_ner.py

python3 -c ’
for d in ["train","dev","test"]:
  with open("UD_Classical_Chinese-Kyoto/lzh_kyoto-ud-"+d+".conllu","r",encoding="utf-8") as f:
    r=f.read()
  with open(d+".json","w",encoding="utf-8") as f:
    tokens=[]
    tags=[]
    i=0
    for s in r.split("\n"):
      t=s.split("\t")
      if len(t)==10 and not s.startswith("#"):
        for c in t[1]:
          tokens.append(c)
          i+=1
      else:
        if i==1:
          tags.append("S")
        elif i==2:
          tags+=["B","E"]
        elif i==3:
          tags+=["B","E2","E"]
        elif i>3:
          tags+=["B"]+["M"]*(i-4)+["E3","E2","E"]
        i=0
        if len(tokens)>80:
          print("{\"tokens\":[\""+"\",\"".join(tokens)+"\"],\"tags\":[\""+"\",\"".join(tags)+"\"]}",file=f)
          tokens=[]
          tags=[]
    if len(tokens)>0:
      print("{\"tokens\":[\""+"\",\"".join(tokens)+"\"],\"tags\":[\""+"\",\"".join(tags)+"\"]}",file=f)
’
python3 run_ner.py --model_name_or_path $BERT --seed 1234 --train_file train.json --validation_file dev.json \
 --test_file test.json --output_dir ‘basename $BERT‘.danku --do_train --do_eval --do_predict
```

図 2: RoBERTa-Classical-Chinese (base) による文切りモデル構築用シェルスクリプト

憲 問 恥 子 曰 邦 有 道 穀 邦 無 道 穀 恥 也 克 伐 怨 欲 不 行 焉 可 以 為 仁 矣 子 曰 可 以 為 難 矣 仁 則 吾 不 知 也
B E2 E B E B E3 E2 E B M M E3 E2 E B M M M E3 E2 E B M E3 E2 E B E B M E3 E2 E B M M E3 E2 E

(a) RoBERTa-Classical-Chinese (base・large)

憲 問 恥 子 曰 邦 有 道 穀 邦 無 道 穀 恥 也 克 伐 怨 欲 不 行 焉 可 以 為 仁 矣 子 曰 可 以 為 難 矣 仁 則 吾 不 知 也
B E2 E B E B E3 E2 E B E2 E B E2 E B M M M E3 E2 E B M E3 E2 E B E B M E3 E2 E B M M E3 E2 E

(b) SikuBERT・SikuRoBERTa・AnchiBERT

〓 〓 〓 子 曰 邦 有 道 穀 邦 〓 道 穀 〓 也 克 伐 怨 欲 不 行 焉 可 以 〓 仁 矣 子 曰 可 以 〓 〓 矣 仁 〓 吾 不 知 也
B E2 E B E B E3 E2 E B M M E3 E2 E B M M M E3 E2 E B M E3 E2 E B E B M E3 E2 E B M M E3 E2 E

(c) GuwenBERT (base・large)

憲問恥 ○ 子曰 ○ 邦有道 ○ 穀 ○ 邦無道 ○ 穀 ○ 恥也 ○ 克 ○ 伐 ○ 怨 ○ 欲不行焉 ○ 可以為仁矣 ○ 子曰 ○ 可以為難矣 ○ 仁則吾不知也 ○

(d)『古詩文斷句』

図 3: 文切り結果の比較

表 2: 改良版の SikuBERT・SikuRoBERTa による文切りモデルの評価 (seqeval)

	評価 (evaluation)				テスト (predict)			
	accuracy	F1	precision	recall	accuracy	F1	precision	recall
SikuBERT (2021.9.13 版)	90.47	87.84	87.42	88.26	89.33	86.75	86.42	87.08
SikuRoBERTa (2021.9.22 版)	91.48	89.04	88.45	89.63	90.37	87.94	87.16	88.74

表 3: RoBERTa-Classical-Chinese による古典中国語の文切りモデル・単語切りモデルの URL

文切り (base)	`https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-base-sentence-segmentation`
文切り (large)	`https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-large-sentence-segmentation`
単語切り (base)	`https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-base-upos`
単語切り (large)	`https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-large-upos`

デルを再発表した．改良版 SikuBERT・SikuRoBERTa に対し，文切りモデルを製作・評価したところ，表 2 の結果となった．SikuBERT の精度はほぼ「横這い」だが，SikuRoBERTa の精度は明らかに上がっている．この結果からも，ボキャブラリーの古典中国語向けチューニングは，解析精度の向上に寄与するといえるだろう．

4 おわりに

Transformers と RoBERTa-Classical-Chinese による古典中国語の文切りモデルを，HuggingFace ハブから公開した．表 3 に URL を示す．また，これらの文切りモデルを，古典中国語係り受け解析器 SuPar-Kanbun [g)]の文切りモジュール (`Danku=True` 時に起動) として組み込んだ．これらに加え，同様の手法を UPOS (Universal Part-Of-Speech)[12] に適用し，古典中国語の品詞付与と単語切りを同時におこなうモデル (付録参照) も公開した．ぜひ使ってみてほしい．

参考文献

[1] 安岡孝一: 漢文の形態素解析・依存文法解析・直接構成鎖解析, 東方學報, 第 94 册 (2019 年 12 月), pp.330-322.

[2] 安岡孝一, ウィッテルン クリスティアン, 守岡知彦, 池田巧, 山崎直樹, 二階堂善弘, 鈴木慎吾, 師茂樹: 古典中国語 (漢文) の形態素解析とその応用, 情報処理学会論文誌, Vol.59, No.2 (2018 年 2 月), pp.323-331.

[3] 安岡孝一: 四書を学んだ MeCab + UDPipe はセンター試験の漢文を読めるのか, 東洋学へのコンピュータ利用, 第 30 回研究セミナー (2019 年 3 月), pp.3-110.

[4] 胡韌奮, 李紳, 諸雨辰: 基于深層語言模型的古漢語知識表示及自動断句研究, 中文信息学報, Vol.35, No.4 (2021 年 4 月), pp.8-15.

[5] 服部宇之吉, 三島毅, 重野安繹, 竹添進一郎, 星野恆, 小柳司氣太, 安井小太郎, 島田鈞一, 岡田正之, 井上哲次郎: 漢文大系, 冨山房 (1909〜1916 年).

[6] 王博立，史暁東，蘇勁松: 一種基于循環神経網絡的古文断句方法, 北京大学学報 (自然科学版), Vol.53, No.2 (2017 年 3 月), pp.255-261.

[7] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush: Transformers: State-of-the-Art Natural Language Processing, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (October 2020): System Demonstrations, pp.38-45.

[8] 闔覃, 遲澤聞: 基于継続訓練的古漢語語言模型, 第 19 届中国計算語言学大会 “古聯杯” 古籍文献命名実体識別 (2020 年 10 月).

[9] 王東波, 劉暢, 朱子赫, 劉江峰, 胡昊天, 沈思, 李斌: SikuBERT 与 SikuRoBERTa: 面向数字人文的《四庫全書》預訓練模型構建及応用研究, 図書館論壇 (網絡首発 2021 年 8 月 20 日).

[10] Huishuang Tian, Kexin Yang, Dayiheng Liu and Jiancheng Lv: AnchiBERT: A Pre-Trained Model for Ancient Chinese Language Understanding and Generation, IJCNN 2021: International Joint Conference on Neural Network (July 2021), [#630].

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (June 2019), Human Language Technologies, Vol.1, pp.4171-4186.

[12] Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, Daniel Zeman: Universal Dependencies, Computational Linguistics, Vol.47, No.2 (June 2021), pp.255-308.

[13] Yuanhe Tian, Yang Song, Xiang Ao, Fei Xia, Xiaojun Quan, Tong Zhang, Yonggang Wang: Joint Chinese Word Segmentation and Part-of-speech Tagging via Two-way Attentions of Auto-analyzed Knowledge, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (July 2020), pp.8286-8296.

[14] Tolmachev Arseny, 河原大輔, 黒橋禎夫: 大規模な自動解析データが形態素解析器をどこまで小さくできるか, 言語処理学会第 25 回年次大会発表論文集 (2019 年 3 月), pp.209-212.

[15] Pattarawat Chormai, Ponrawee Prasertsom, Jin Cheevaprawatdomrong and Attapol T. Rutherford: Syllable-based Neural Thai Word Segmentation, Proceedings of the 28th International Conference on Computational Linguistics (December 2020), pp.4619-4637.

[g)] `https://pypi.org/project/suparkanbun`

付録　Transformers を用いた古典中国語 (漢文) 単語切りモデルの製作

古典中国語の単語切り (word tokenization) モデルを，Transformers [7] の系列ラベリングで実装してみよう．例として「或五十歩而後止」という文を,「或」「五十」「歩」「而」「後」「止」の 6 つの単語に，単語切りすることを考える．

古典中国語 BERT/RoBERTa モデルは，漢字に対しては，1 文字＝ 1 語という形で設計されている．われわれの RoBERTa-Classical-Chinese も，モデルの設計は GuwenBERT [8] を引き継いでおり，1 文字＝ 1 語である．しかしながら，古典中国語の単語は 1 文字とは限らず，2 文字以上の単語も存在する．特に数詞と固有名詞は，2 文字以上の単語が多い．

[13] のアルゴリズムは，1 文字＝ 1 語の現代中国語 BERT モデル上で，現代中国語の単語切りと品詞付与を同時におこなう．これにより，2 文字以上の単語に対する解析精度が良くなるらしいのだが，では，これを，古典中国語に応用できないだろうか．すなわち，1 文字の単語に対しては，UPOS (Universal Part-Of-Speech)[12] をそのまま品詞付与し，一方，2 文字以上の単語に対しては，1 文字目は「B-」を，2 文字目以降は「I-」を付けた UPOS を付与する.「或五十歩而後止」の例で言えば,「五十」だけが 2 文字の単語なので,「五」に B-NUM を,「十」に I-NUM を付与し，それ以外の単語には UPOS をそのまま付与する (図 4).

UPOS 品詞付与による古典中国語 (漢文) 単語切りモデルを，Transformers 4.9.2 の `run_ner.py` による系列ラベリングで製作した．訓練データには Universal Dependencies 2.8.1 の `lzh_kyoto-ud-train.conllu` を，評価データには `lzh_kyoto-ud-dev.conllu` を，テストデータには `lzh_kyoto-ud-test.conllu` を，それぞれ用いた．図 5 のスクリプトを用いて訓練 (単語切りモデル構築) に要した時間は，3 epochs・1 GPU (NVIDIA GeForce RTX 2080) で，各モデル 5〜10 分程度だった．なお，比較のため，改良版 SikuBERT・SikuRoBERTa その他の古典中国語モデルに対しても，UPOS 品詞付与＋単語切りモデルを製作した．

seqeval による評価結果を，表 4 に示す．われわれの RoBERTa-Classical-Chinese による UPOS 品詞付与＋単語切りモデルの解析精度が高く，また，base モデル (12 層・768 隠れ層・12 ヘッド) より large モデル (24 層・1024 隠れ層・16 ヘッド) の方が，良い結果となっている．

Transformers と RoBERTa-Classical-Chinese による古典中国語の UPOS 品詞付与＋単語切りモデルを，HuggingFace ハブから公開した (表 3)．また，この手法を，現代中国語 [13]・日本語 [14]・タイ語 [15] にも適用 (表 5) し，多言語係り受け解析器 esupar [h] の UPOS 品詞付与＋単語切りモジュールとして組み込んだ．ぜひ使ってみてほしい．

或五十歩而後止
PRON B-NUM I-NUM NOUN CCONJ NOUN VERB

図 4: 「B-」「I-」を併用した UPOS 品詞付与

[h] `https://pypi.org/project/esupar`

表 4: 古典中国語 UPOS 品詞付与＋単語切りモデルの評価 (seqeval)

	評価 (evaluation)				テスト (predict)			
	accuracy	F1	precision	recall	accuracy	F1	precision	recall
RoBERTa-Classical-Chinese (base)	94.25	90.78	90.77	90.78	94.55	91.62	91.65	91.60
RoBERTa-Classical-Chinese (large)	94.53	91.17	91.22	91.12	94.93	92.04	92.10	91.97
SikuBERT (2021.9.13 版)	93.80	90.14	90.08	90.19	94.11	91.01	91.07	90.94
SikuRoBERTa (2021.9.22 版)	94.15	90.45	90.48	90.42	94.29	91.20	91.25	91.14
AnchiBERT [10]	92.73	88.56	88.63	88.49	92.61	88.92	89.04	88.80
GuwenBERT (large) [8]	88.17	82.08	82.41	81.74	88.85	83.25	83.77	82.74
GuwenBERT (base) [8]	87.18	80.64	80.98	80.30	87.67	81.71	82.29	81.14

```
#! /bin/sh
BERT=KoichiYasuoka/roberta-classical-chinese-base-char

pip3 install transformers==4.9.2 datasets seqeval
test -d UD_Classical_Chinese-Kyoto ||
 git clone -b r2.8 https://github.com/universaldependencies/UD_Classical_Chinese-Kyoto
test -f run_ner.py ||
 curl -LO https://raw.githubusercontent.com/huggingface/transformers/v‘pip3 list |
 sed -n ’s/^transformers *\([^ ]*\) *$/\1/p’‘/examples/pytorch/token-classification/run_ner.py

python3 -c ’
for d in ["train","dev","test"]:
  with open("UD_Classical_Chinese-Kyoto/lzh_kyoto-ud-"+d+".conllu","r",encoding="utf-8") as f:
    r=f.read()
  with open(d+".json","w",encoding="utf-8") as f:
    tokens=[]
    tags=[]
    for s in r.split("\n"):
      t=s.split("\t")
      if len(t)==10 and not s.startswith("#"):
        for c in t[1]:
          tokens.append(c)
        if len(t[1])==1:
          tags.append(t[3])
        else:
          tags.extend(["B-"+t[3]]+["I-"+t[3]]*(len(t[1])-1))
      elif len(tokens)>80:
        print("{\"tokens\":[\""+"\",\"".join(tokens)+"\"],\"tags\":[\""+"\",\"".join(tags)+"\"]}",file=f)
        tokens=[]
        tags=[]
    if len(tokens)>0:
      print("{\"tokens\":[\""+"\",\"".join(tokens)+"\"],\"tags\":[\""+"\",\"".join(tags)+"\"]}",file=f)
’
python3 run_ner.py --model_name_or_path $BERT --seed 1234 --train_file train.json --validation_file dev.json \
 --test_file test.json --output_dir ‘basename $BERT‘.upos --do_train --do_eval --do_predict
```

図 5: RoBERTa-Classical-Chinese (base) による UPOS 品詞付与＋単語切りモデル構築用シェルスクリプト

表 5: 現代中国語・日本語・タイ語 UPOS 品詞付与＋単語切りモデルの URL

現代中国語		https://huggingface.co/KoichiYasuoka/chinese-bert-wwm-ext-upos
		https://huggingface.co/KoichiYasuoka/chinese-roberta-base-upos
		https://huggingface.co/KoichiYasuoka/chinese-roberta-large-upos
日本語	国語研短単位	https://huggingface.co/KoichiYasuoka/bert-base-japanese-upos
		https://huggingface.co/KoichiYasuoka/bert-large-japanese-upos
	国語研長単位	https://huggingface.co/KoichiYasuoka/bert-base-japanese-luw-upos
		https://huggingface.co/KoichiYasuoka/bert-large-japanese-luw-upos
タイ語		https://huggingface.co/KoichiYasuoka/roberta-base-thai-syllable-upos