古典中国語の形態素解析と係り受け解析

安岡孝一*・安岡素子*

1　はじめに

2019 年 5 月リリースの Universal Dependencies (UD) 2.4 において、われわれの研究グループは、四書 (孟子・論語・大学・中庸) の依存構造コーパス (11176 文、55026 語、56768 字) を製作[1]し、UD における開発の一翼を担うことにした[2]。開発は現在も継続しており、2022 年 11 月リリースの UD 2.11 は、『孟子』『論語』『禮記』『十八史略』『楚辭』『唐詩三百首』『摩訶般若波羅蜜大明呪經』『金剛般若波羅蜜經』『佛説阿彌陀經』の依存構造コーパス (63079 文、310594 語、324415 字) を収録[3]している。

これら依存構造コーパスの開発過程において、われわれは、UD と Transformers[4]による古典中国語 (漢文) 形態素解析・係り受け解析モデルを製作した。具体的には、事前学習モデル roberta-classical-chinese-base-char[5]に対して、系列ラベリングによるファインチューニングをおこない、形態素解析と係り受け解析を同時におこなう手法を開発した。本稿では、この手法の概要を紹介する。また、各手法を読者が実際に試せるよう、Google Colaboratory[6]でのプログラミング例も、合わせて示すことにする。

2　Universal Dependencies の概要

UD は、書写言語における品詞・形態素属性・依存構造 (係り受け関係) を、言語に関わらず記述する手法である[7]。句構造を考慮せずに係り受け関係を記述することで、言語横断性を高めており、全ての文法構造を単語間のリンクで記述するのが特徴である。

依存構造解析それ自体は、Tesnière の構造的統語論[8]に源を発し、Мельчук の有向グラフ記述[9]によって、一応の完成を見た手法である。その最大の特長は、いわゆる動詞中心主義によって言語横断的な記述が可能だという点にあり、Мельчук 依存文法をコンピュー

*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

[1]Koichi Yasuoka: Universal Dependencies Treebank of the Four Books in Classical Chinese, DADH2019: 10th International Conference of Digital Archives and Digital Humanities (December 2019), pp.20-28.

[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, 藤田一乘: 古典中国語 (漢文) Universal Dependencies とその応用, 情報処理学会論文誌, Vol.63, No.2 (2022 年 2 月), pp.355-363.

[3]`https://github.com/UniversalDependencies/UD_Classical_Chinese-Kyoto`

[4]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.

[5]`https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-base-char`

[6]`https://colab.research.google.com`

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

[8]Lucien Tesnière: Éléments de Syntaxe Structurale, Paris: C. Klincksieck (1959).

[9]Igor A. Mel'čuk: Dependency Syntax: Theory and Practice, New York: State University of New York Press (1988).

表 1: CoNLL-U の各フィールド

1. ID: 単語ごとに付与されたインデックスで、文ごとに 1 から始まる整数。縮約語に対しては、単語の範囲を示すのも可。
2. FORM: 語、または、句読記号。
3. LEMMA: 基底形、語幹。
4. UPOS: UD で規定された言語普遍な品詞タグ (表 2)。
5. XPOS: 言語固有の品詞タグ。
6. FEATS: UD で規定された言語普遍な形態素属性のリスト。言語固有の拡張も可。
7. HEAD: 当該の単語の係り受け元 ID。係り受け元が無い場合は 0 とする。
8. DEPREL: UD で規定された言語普遍な係り受けタグ (表 3)。HEAD が 0 の場合は root とする。言語固有の拡張も可。
9. DEPS: 複数の係り受け元を持つ場合、全ての HEAD:DEPREL ペア。
10. MISC: その他のアノテーション。

表 2: UD 品詞タグ (UPOS)

Open class words	**Closed class words**	**Other**
ADJ 形容詞	ADP 側置詞	PUNCT 句読点
ADV 副詞	AUX 助動詞	SYM 記号
INTJ 感嘆詞	CCONJ 並列接続詞	X その他
NOUN 名詞	DET 限定詞	
PROPN 固有名詞	NUM 数詞	
VERB 動詞	PART 接辞	
	PRON 代名詞	
	SCONJ 従属接続詞	

表 3: UD 係り受けタグ (DEPREL)

	Nominals	**Clauses**	**Modifier Words**	**Function Words**
Core arguments	nsubj 主語 obj 目的語 iobj 間接目的語	csubj 節主語 ccomp 節目的語 xcomp 節補語		
Non-core dependents	obl 斜格補語 vocative 呼称語 expl 形式語 dislocated 外置語	advcl 連用修飾節	advmod 連用修飾語 discourse 談話要素	aux 動詞補助成分 cop 繋辞 mark 標識
Nominal dependents	nmod 体言による連体修飾語 appos 同格 nummod 数量による修飾語	acl 連体修飾節	amod 用言による連体修飾語	det 決定語 clf 類別語 case 格表示

Coordination	**MWE**	**Loose**	**Special**	**Other**
conj 接続 cc 接続語	fixed 固着 flat 並列 compound 複合	list 細目 parataxis 隣接表現	orphan 親なし goeswith 泣き別れ reparandum 言い損じ	punct 句読点 root 親 dep 未定義

タ向けに洗練した UD においても、言語に関わらない記述、という特長が前面に押し出されている。UD における文法構造記述は、句構造を考慮せず、全てを単語間のリンクとして表現する。これにより、言語横断的な文法構造記述を可能としている。

UD 係り受けコーパスの交換用フォーマットとして、CoNLL-U と呼ばれるタブ区切りテキスト (文字コードは UTF-8[10]) が規定されている。CoNLL-U の各行は各単語に対応しており、表 1 に示す 10 個のタブ区切りフィールドで構成される。ID・FORM・LEMMA は、単語そのものに関するフィールドである。UPOS・XPOS・FEATS は、単語の品詞と形態素属性に関するフィールドである。HEAD・DEPREL・DEPS は、単語の係り受けに関するフィールドである。

UD における係り受け関係は、単語間の有向グラフを HEAD と DEPREL で記述する。HEAD は、その単語に入る有向枝のリンク元 ID を示しており、DEPREL は、その有向枝における係り受けタグである。ただし、HEAD が 0 の場合、その枝に入るリンク元は存在しない。リンクの本数は単語の個数に等しく、各リンクのリンク先は、全て互いに異なっている。すなわち、各単語から出るリンクは複数の可能性があるが、各単語に入るリンクは 1 つだけである。なお、リンクはループしない。

UD の係り受けリンクは、Мельчук 依存文法の後裔にあたり、いわゆる動詞中心主義である。動詞をリンク元として、主語や目的語へとリンクする。修飾関係においては、被修飾語から修飾語へとリンクする。ただし、側置詞 (前置詞や後置詞) を体言の修飾語だとみなす点[11]が、Мельчук とは異なっている。また、コピュラ文においては動詞中心主義を採らず、補語をリンク元として、主語や繋辞へとリンクする。

UD を古典中国語へ適用するに際して、われわれは係り受けタグを拡張 (discourse:sp、nsubj:pass、nsubj:outer、csubj:outer、obl:tmod、obl:lmod、compound:redup、flat:vv)

```
# text = 未聞好學者也
1   未   未   ADV   _   Polarity=Neg  2  advmod        _   SpaceAfter=No
2   聞   聞   VERB  _   _             0  root          _   SpaceAfter=No
3   好   好   VERB  _   _             5  acl           _   SpaceAfter=No
4   學   學   NOUN  _   _             3  obj           _   SpaceAfter=No
5   者   者   PART  _   _             2  obj           _   SpaceAfter=No
6   也   也   PART  _   _             2  discourse:sp  _   SpaceAfter=No
```

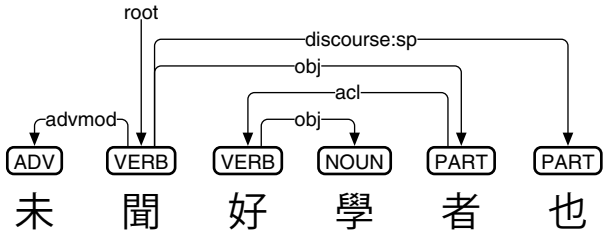


図 1: 古典中国語 CoNLL-U と deplacy による可視化

[10]ISO/IEC 10646-1:1993/Amd.2:1996 Information Technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane, Amendment 2: UCS Transformation Format 8 (UTF-8), International Organization for Standardization, Genève (October 15, 1996).

[11]Joakim Nivre: Towards a Universal Grammar for Natural Language Processing, CICLing 2015: 16th International Conference on Intelligent Text Processing and Computational Linguistics (April 2015), pp.3-16.

した[12]。特に discourse:sp は文助詞 (sentence particle) を表しており[13]、古典中国語における動賓終構造 (predicate-object-final structure) の終を、記述するために用いている。例として「未聞好學者也」の CoNLL-U と、deplacy[14]による可視化を図 1 に示す。「聞」から「者」へ obj が、「聞」から「也」へ discourse:sp が繋がれており、「聞-者-也」という動賓終構造を、UD で記述できている。

3 BERT/RoBERTa モデルと穴埋めゲーム

Google AI Language が発表した BERT[15]は、大量の文章を機械に丸暗記させる際にどのような手法が有効なのかを、端的に示した事前学習モデルである。単純に文章を丸暗記するのではなく、文章中に現れる文と文の繋がり、あるいは、文中に現れるトークン (単語もしくは単語より短い文字列) とトークンの繋がりを、効果的に学習できるような工夫がなされている。

トークンとトークンの繋がりを学習するために用いられるのが、穴埋めゲームである。「This [MASK] a pen.」の [MASK] に対し、「This is a pen.」という答を得るような穴埋めゲーム (図 2) を考えてみよう。bert-base-cased[16]においては、各トークンの入出力は文字コード (UTF-8) でおこなわれるが、内部的には 768 次元のベクトル (数値の列) で表現される。入力側においては、各入力トークンに対応するベクトルをそのまま用いるが、出力側においては、各出力トークンに近い[17]ベクトルになるように学習をおこなう。「This [MASK] a pen.」の例で言えば、「This」「a」「pen」「.」は入力ベクトルをほぼ素通ししつ

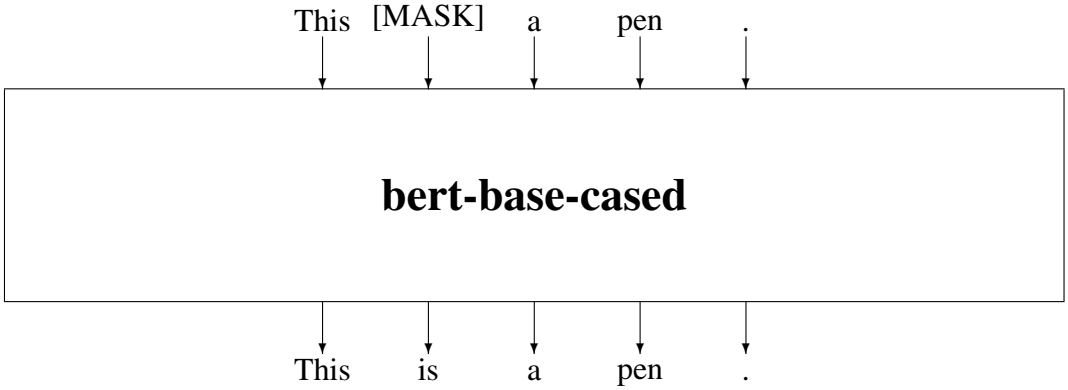


図 2: bert-base-cased における穴埋めゲーム

[12] https://universaldependencies.org/lzh/#syntax

[13] Herman Leung, Rafaël Poiret, Tak-sum Wong, Xinying Chen, Kim Gerdes and John Lee: Developing Universal Dependencies for Mandarin Chinese, 12th Workshop on Asian Language Resources (December 2016), pp.20-29.

[14] 安岡孝一: Universal Dependencies にもとづく多言語係り受け可視化ツール deplacy, 人文科学とコンピュータシンポジウム「じんもんこん 2020」論文集 (2020 年 12 月), pp.95-100.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova: BERT: Pre-training of Deep Bidirectional Transformers for Language, 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.

[16] https://huggingface.co/bert-base-cased

[17] ここでいう「近い」は、コサイン類似度 (2 つのベクトルの内積を、ベクトルのユークリッド・ノルムで割った値) が 1 に近い、という意味である。

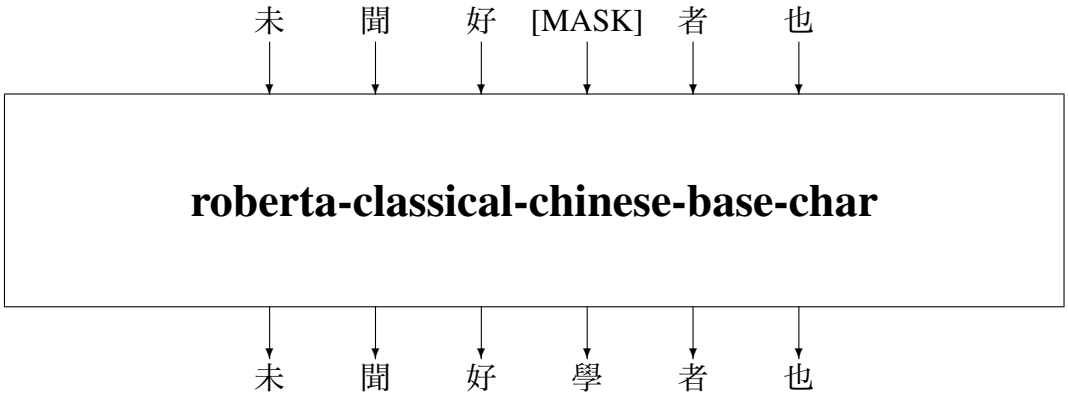


図 3: roberta-classical-chinese-base-char における穴埋めゲーム

つ、「is」に近い出力ベクトルが得られるよう、内部の記憶素子をいじるわけである。このような穴埋めゲームを大量の文に対しておこなうことで、トークンとトークンの繋がりを学習できる、というのが、BERT の工夫の一つ[18]である。

BERT を多言語拡張する際に問題となるのは、モデルの入出力に文という単位を仮定している点である。古典中国語の白文においては、文という単位は必ずしも明確ではない。他の言語においても、引用文を含む文や、SNS での「つぶやき」など、文境界がハッキリしない例は山ほどある。この問題を解決すべく、RoBERTa[19]は、文という単位を仮定せず、トークンの並びを用いて、穴埋めゲームのみに特化した学習をおこなう。これにより、文という単位に捉われることなく、大量の文章を丸暗記できる。

われわれの roberta-classical-chinese-base-char は、古典中国語における漢字をトークンとみなし、漢字単位での穴埋めゲーム (図 3) を学習したモデルである。各漢字 (約 26000 字種) の入出力は UTF-8 でおこなうが、内部的には 768 次元のベクトルで表現する。Google Colaboratory 上の Transformers と roberta-classical-chinese-base-char を用いて、「未聞好 [MASK] 者也」を穴埋めするプログラムと、その結果を図 4 に示す。「學」の確率が 13.5% となっており、「学」の 15.4%に負けているものの、これらは異体字関係にあることから、

```
!pip install transformers
from transformers import pipeline
fmp=pipeline("fill-mask","KoichiYasuoka/roberta-classical-chinese-base-char")
prd=fmp("未聞好[MASK]者也")
print("\n".join("{:8} {:.03f}".format(t["token_str"],t["score"]) for t in prd))
```

```
学      0.154
學      0.135
事      0.031
勇      0.027
之      0.025
```

図 4: 「未聞好 [MASK] 者也」に対する穴埋めゲームプログラムと結果

[18]BERT のもう一つの工夫は、文と文の繋がりを学習する隣接文ゲームだが、本稿では扱わない。
[19]https://arxiv.org/abs/1907.11692

十分に学習できていると言っていいだろう。

BERT/RoBERTa モデルは、出力部を付け替えることにより、様々な用途に応用可能である。出力ベクトルを品詞情報だとみなせば、品詞付与に使える。係り受けタグだとみなせば、係り受け解析に使える。UD においては、形態素解析や係り受け解析への応用が重要であり、以下、古典中国語に対する実装を示していくことにする。

4　系列ラベリングを用いた古典中国語の形態素解析

roberta-classical-chinese-base-ud-goeswith[20]は、roberba-classical-chinese-base-char の出力部を付け替えて、UD の UPOS や DEPREL を出力するよう、ファインチューニングをおこなったモデル[21]である。BERT/RoBERTa モデルにおける系列ラベリングは、各トークンにラベルを付与することができるので、ラベルとして表 2 の UPOS を付与すれば、形態素解析 (品詞付与) モデルとして使用できる (図 5)[22]。

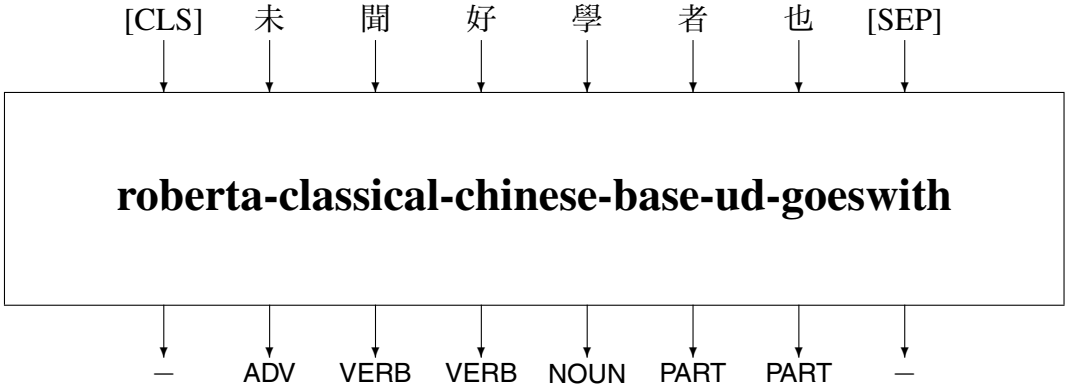


図 5: 系列ラベリングを用いた古典中国語の UPOS 付与

Google Colaboratory 上の Transformers と roberta-classical-chinese-base-ud-goeswith を用いて、「未聞好學者也」に UPOS を付与するプログラムと、その結果を図 6 に示す。漢字 1 文字が 1 トークンとして扱われており、それぞれに UPOS が付与されている。

しかしながら、古典中国語における単語は、漢字 1 文字とは限らない。複数の漢字から成る単語 (たとえば「孟子」) も存在する。そのような単語に対し roberta-classical-chinese-base-ud-goeswith では、2 文字目以降の漢字に X という UPOS 品詞を付与することで、複数の漢字から成る漢語を表現している。Google Colaboratory 上の Transformers と roberta-classical-chinese-base-ud-goeswith を用いて、「孟子見梁惠王」に UPOS を付与するプログラムと、その結果を図 7 に示す。図 7 では「孟子」が 2 文字の漢語であり、「見」「梁」「惠」「王」は 1 文字の漢語とみなされている。

[20]https://huggingface.co/KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith
[21]系列ラベリングの実装には、Transformers の RobertaForTokenClassification を用いた。
[22]文頭を表す [CLS] と文末を表す [SEP] も使用しているが、これらは次章の係り受け解析と整合性を取るためである。

```
!pip install deplacy transformers
class SeqL(object):
  def __init__(self,bert):
    from transformers import pipeline
    self.tagger=pipeline(task="ner",model=bert)
  def __call__(self,text):
    w=[(t["start"],t["end"],t["entity"]) for t in self.tagger(text)]
    u="# text = "+text.replace("\n"," ")+"\n"
    for i,(s,e,p) in enumerate(w,1):
      m,q="_" if i<len(w) and e<w[i][0] else "SpaceAfter=No",p.split("|")
      f="_" if p.find("=")<0 else "|".join(t for t in q if t.find("=")>0)
      u+="\t".join([str(i),text[s:e],"_",q[0],"_",f,"_","_","_",m])+"\n"
    return u+"\n"
nlp=SeqL("KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith")
doc=nlp("未聞好學者也")
import deplacy
deplacy.serve(doc,port=None)
```

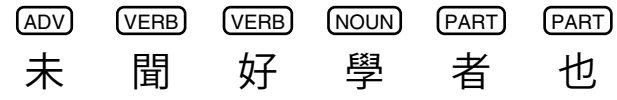


図 6: roberta-classical-chinese-base-ud-goeswith による UPOS 付与

```
!pip install deplacy transformers
class SeqL(object):
  def __init__(self,bert):
    from transformers import pipeline
    self.tagger=pipeline(task="ner",model=bert)
  def __call__(self,text):
    w=[(t["start"],t["end"],t["entity"]) for t in self.tagger(text)]
    u="# text = "+text.replace("\n"," ")+"\n"
    for i,(s,e,p) in enumerate(w,1):
      m,q="_" if i<len(w) and e<w[i][0] else "SpaceAfter=No",p.split("|")
      f="_" if p.find("=")<0 else "|".join(t for t in q if t.find("=")>0)
      u+="\t".join([str(i),text[s:e],"_",q[0],"_",f,"_","_","_",m])+"\n"
    return u+"\n"
nlp=SeqL("KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith")
doc=nlp("孟子見梁惠王")
import deplacy
deplacy.serve(doc,port=None)
```

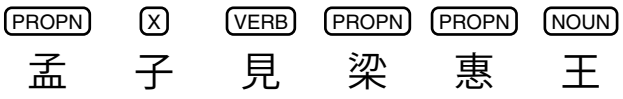


図 7: 複数の漢字から成る漢語に X を用いた UPOS 付与

5　系列ラベリングを用いた古典中国語の係り受け解析

系列ラベリングを用いて、係り受け解析をおこなう手法を考えてみよう。UD を有向グラフとして見た場合、係り受け解析は隣接行列の導出だと考える[23]ことができる。たとえば、図 1 の「未聞好學者也」の有向グラフであれば

$$\begin{bmatrix} – & – & – & – & – & – \\ \text{advmod} & \text{root} & – & – & \text{obj} & \text{discourse:sp} \\ – & – & – & \text{obj} & – & – \\ – & – & – & – & – & – \\ – & – & \text{acl} & – & – & – \\ – & – & – & – & – & – \end{bmatrix}$$

という 6×6 の隣接行列を導出すれば良い。この隣接行列において、各列は各トークンに入る有向枝の係り受けタグ、各行は各トークンから出る有向枝の係り受けタグである。なお、対角成分では root を示すことにする。

Google Colaboratory 上の Transformers と roberta-classical-chinese-base-ud-goeswith を用いて、「未聞好學者也」の隣接行列のロジット[24]を求めるプログラムと、その結果を図 9 に示す。図 9 の例では、各列ごとにロジット最大の要素を選ぶだけで、「未聞好學者也」の有向グラフを正しく導出できる。ただし、実際の係り受け解析においては、ループが起こらないよう、Chu-Liu-Edmonds アルゴリズム[25]を適用する (図 8)。

```
!pip install transformers ufal.chu_liu_edmonds deplacy
from transformers import pipeline
nlp=pipeline(model="KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith",
  task="universal-dependencies",trust_remote_code=True)
doc=nlp("未聞好學者也")
import deplacy
deplacy.serve(doc,port=None)
```

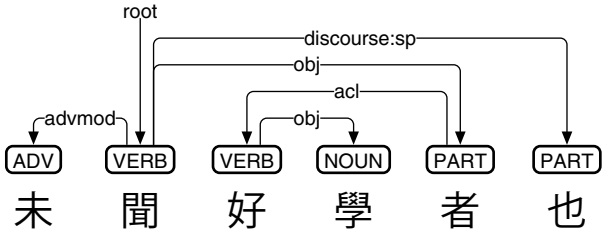


図 8: roberta-classical-chinese-base-ud-goeswith による係り受け解析

roberta-classical-chinese-base-goeswith の内部では、隣接行列の各行ごとに系列ラベリングをおこなっている (図 10)。この際、隣接行列のどの行に着目しているかを表すため、該当するトークンを [MASK] で隠し、[SEP] の後に該当トークンを移す、という処理をおこなっている。たとえば「未聞好學者也」の「聞」に対応する行であれば、図 10(b) に示すように、入力側の「聞」を [MASK] して、[SEP] の後に「聞」を移している。

[23]Dan Kondratyuk and Milan Straka: 75 Languages, 1 Model: Parsing Universal Dependencies Universally, Proceedings of the 2019 Conference on Empirical Methods in Natural Language (November 2019), pp.2779-2795.

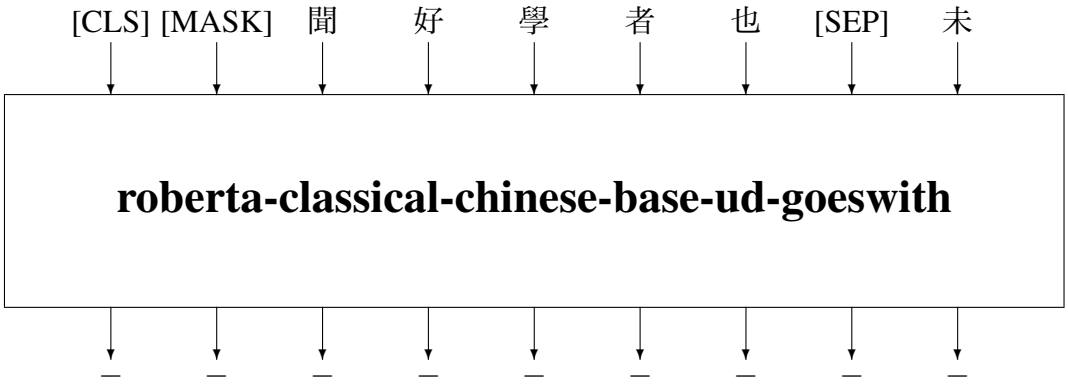
[24]ラベルの生起確率を p とおくとき、ロジット (対数オッド) は $\log \frac{p}{1-p}$ である。

[25]H. N. Gabow, Z. Galil, T. Spencer and R. E. Tarjan: Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs, Combinatorica, Vol.6, No.2 (June 1986), pp.109-122.

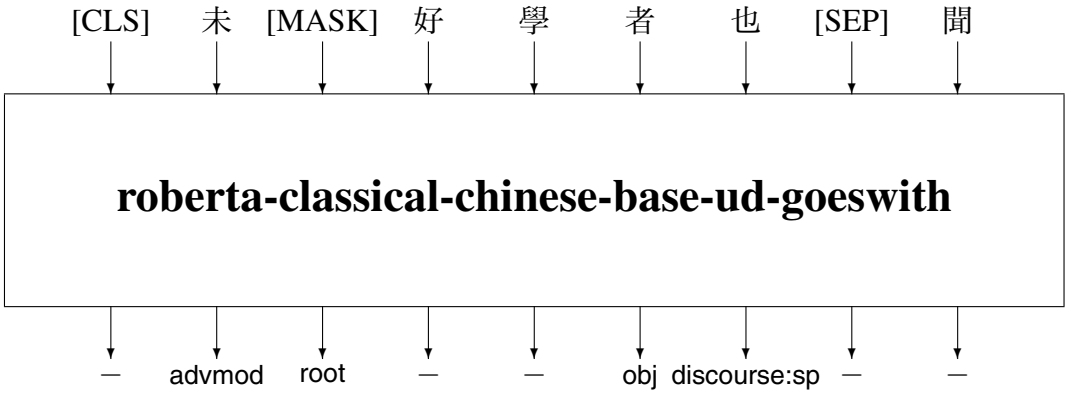
```
!pip install transformers
import torch,numpy
from transformers import AutoTokenizer,AutoModelForTokenClassification
brt="KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith"
txt="未聞好學者也"
tkz=AutoTokenizer.from_pretrained(brt)
mdl=AutoModelForTokenClassification.from_pretrained(brt)
v,l=tkz(txt,return_offsets_mapping=True),mdl.config.id2label
w,u=v["input_ids"],[txt[s:e] for s,e in v["offset_mapping"] if s<e]
x=[w[:i]+[tkz.mask_token_id]+w[i+1:]+[j] for i,j in enumerate(w[1:-1],1)]
with torch.no_grad():
  m=mdl(input_ids=torch.tensor(x)).logits.numpy()[:,1:-2,:]
r=[1 if i==0 else -1 if l[i].endswith("|root") else 0 for i in range(len(l))]
m+=numpy.where(numpy.add.outer(numpy.identity(m.shape[0]),r)==0,0,numpy.nan)
d,p=numpy.nanmax(m,axis=2),numpy.nanargmax(m,axis=2)
print(" ".join(x.rjust(12-len(x)) for x in u))
for i,j in enumerate(u):
  print("\n"+" ".join("{:12.3f}".format(x) for x in d[i])," ",j)
  print(" ".join(l[x].split("|")[-1][:12].rjust(12) for x in p[i]))
```

未	聞	好	學	者	也	
2.501	3.073	2.156	2.156	1.241	3.359	未
root	goeswith	conj	obj	conj	discourse:sp	
13.925	14.172	3.748	5.088	12.398	12.621	聞
advmod	root	ccomp	obj	obj	discourse:sp	
1.403	2.656	4.458	11.748	3.088	4.822	好
advmod	advmod	root	obj	mark	discourse:sp	
1.364	2.630	6.094	2.734	1.881	3.548	學
flat	amod	amod	root	case	discourse:sp	
1.233	3.927	12.372	4.860	1.477	5.199	者
advmod	amod	acl	nmod	root	discourse:sp	
1.628	1.754	1.413	2.450	1.019	1.965	也
flat	amod	amod	obj	goeswith	root	

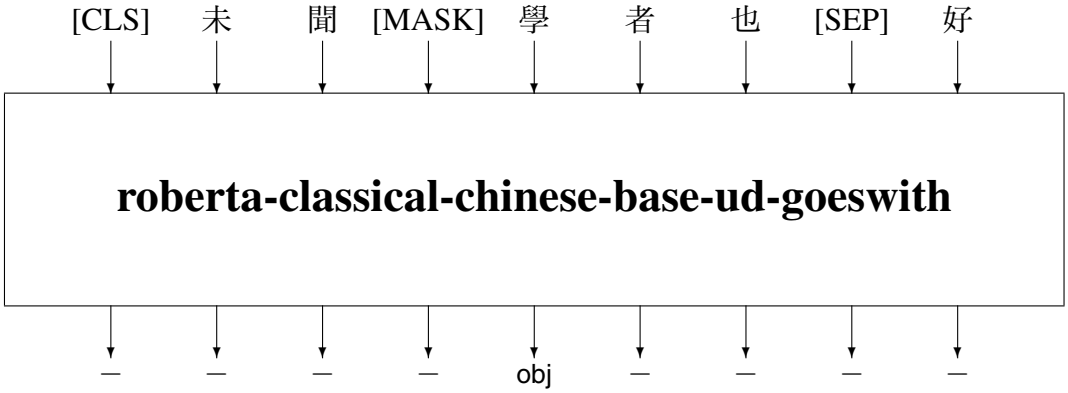
図 9: 「未聞好學者也」に対する隣接行列ロジットの導出



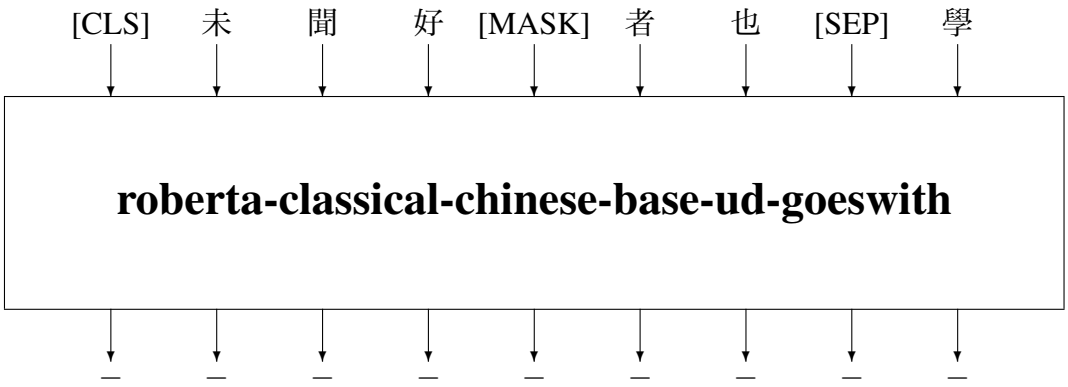
(a)「未」から出る有向枝



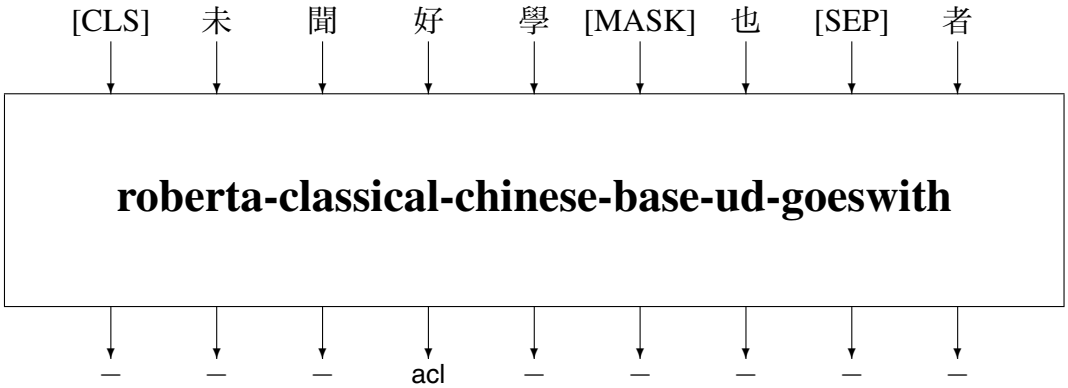
(b)「聞」から出る有向枝 (root を含む)



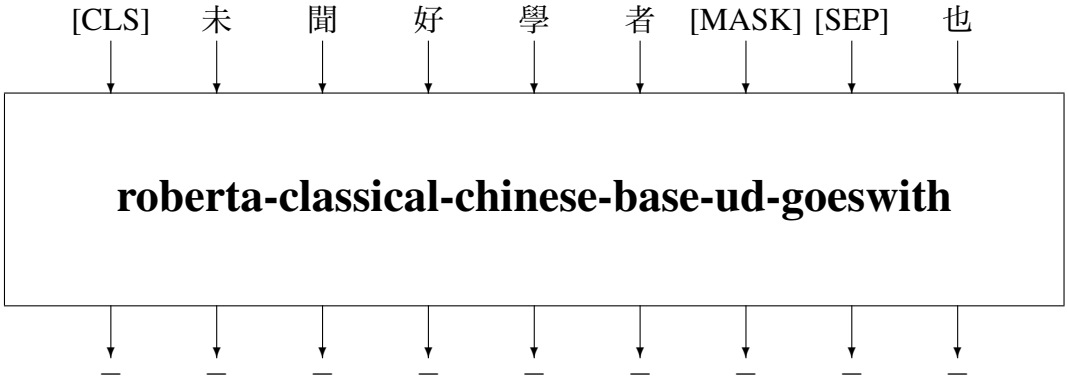
(c)「好」から出る有向枝



(d)「學」から出る有向枝



(e)「者」から出る有向枝



(f)「也」から出る有向枝

図 10: 系列ラベリングを用いた「未聞好學者也」の隣接行列学習

```
!pip install transformers ufal.chu_liu_edmonds deplacy
from transformers import pipeline
nlp=pipeline(task="universal-dependencies",trust_remote_code=True,
  model="KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith")
doc=nlp("孟子見梁惠王")
import deplacy
deplacy.serve(doc,port=None)
```

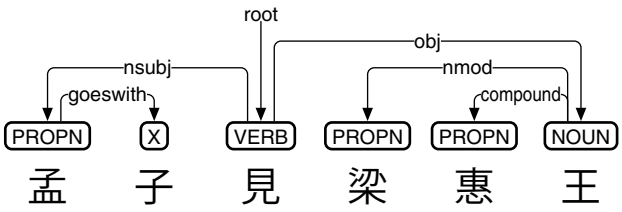


図 11: 複数の漢字から成る漢語での単語内リンク goeswith

ただし、古典中国語における単語は、漢字 1 文字とは限らず、複数の漢字から成る単語も存在する。そのような単語であっても、roberta-classical-chinese-base-ud-goeswith は、1 文字を 1 トークンとして扱うしかない。そこで、複数の漢字からなる漢語に対しては、2 文字目以降の漢字に goeswith(泣き別れ) リンクを導入し、1 文字目の漢字と繋いでおくことにした。Google Colaboratory 上の Transformers と roberta-classical-chinese-base-ud-goeswith を用いて、「孟子見梁惠王」の係り受け解析をおこなうプログラムと、その結果を図 11 に示す。

なお、図 11 は、あくまで内部処理を示したものであって、複数の漢字から成る漢語がバラバラになっており、必ずしも正しい UD とは言えない。そこで、goeswith を解消するオプションとして、`aggregation_strategy="simple"`を準備した。図 12 に Google Colaboratory での使用例を示す。「孟」と「子」を繋いでいた goeswith が解消されて、「孟子」が 1 語となっているのがわかる。

```
!pip install transformers ufal.chu_liu_edmonds deplacy
from transformers import pipeline
nlp=pipeline(task="universal-dependencies",trust_remote_code=True,
  model="KoichiYasuoka/roberta-classical-chinese-base-ud-goeswith",
  aggregation_strategy="simple")
doc=nlp("孟子見梁惠王")
import deplacy
deplacy.serve(doc,port=None)
```

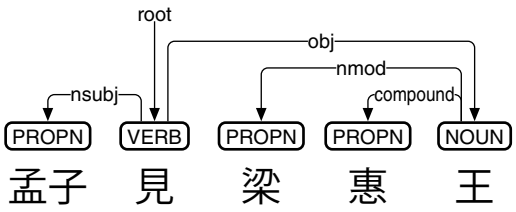


図 12: 「孟子見梁惠王」に対する係り受け解析

6 おわりに

古典中国語 UD にもとづく形態素解析・係り受け解析をおこなうべく、Transformers の系列ラベリング RobertaForTokenClassification を用いて、roberta-classical-chinese-base-ud-goeswith を製作した。このモデルでは、漢字 1 文字を 1 トークンとして扱っており、複数の漢字から成る漢語に対しては、X と goeswith を内部処理に用いた。これにより、形態素解析と係り受け解析を同時におこなう系列ラベリング手法が、古典中国語に対して実装できた。

なお、系列ラベリングは応用範囲が広く、たとえば古典中国語の文切り[26]にも応用できる。図 13 の例では、文頭の漢字に対するラベルを B、文末の漢字に対するラベルを E として、B・M・E3・E2・E を用いて[27]各文を系列ラベリングしている。ぜひ、色々と試してみてほしい。

```
!pip install deplacy transformers
class SeqL(object):
  def __init__(self,bert):
    from transformers import pipeline
    self.tagger=pipeline(task="ner",model=bert)
  def __call__(self,text):
    w=[(t["start"],t["end"],t["entity"]) for t in self.tagger(text)]
    u="# text = "+text.replace("\n"," ")+"\n"
    for i,(s,e,p) in enumerate(w,1):
      m,q="_" if i<len(w) and e<w[i][0] else "SpaceAfter=No",p.split("|")
      f="_" if p.find("=")<0 else "|".join(t for t in q if t.find("=")>0)
      u+="\t".join([str(i),text[s:e],"_",q[0],"_",f,"_","_","_",m])+"\n"
    return u+"\n"
nlp=SeqL("KoichiYasuoka/roberta-classical-chinese-base-sentence-segmentation")
doc=nlp("子曰學而時習之不亦説乎有朋自遠方來不亦樂乎人不知而不慍不亦君子乎")
import deplacy
deplacy.serve(doc,port=None)
```

図 13: roberta-classical-chinese-base-sentence-segmentation による古典中国語の文切り

[26]安岡孝一: Transformers を用いた古典中国語 (漢文) 文切りモデルの製作, 人文科学とコンピュータシンポジウム「じんもんこん 2021」論文集 (2021 年 12 月), pp.104-109.

[27]王博立, 史晓东, 苏劲松: 一种基于循环神经网络的古文断句方法, 北京大学学报(自然科学版), Vol.53, No.2 (2017 年 3 月), pp.255-261.