TR-SLT-0002

Support Vector Machine を用いた Chunking による中国語解析

Chinese Analyzer with Support Vector Machines

吉田 辰巳 Tatsumi YOSHIDA
大竹 清敬 Kiyonori OHTAKE
坂本 仁 Masashi SAKAMOTO

2002.2.22

中国語構文木コーパスの情報を Support Vector Machine で学習し，中国語の形態素解析と Base Phrase Chunking を行った．形態素辞書を用いた他の解析器と比較するなどして，SVM による中国語解析の特徴を調査し，考察した．

(株) 国際電気通信基礎技術研究所
音声言語コミュニケーション研究所
〒619–0288 京都府相楽郡精華町光台二丁目 2 番地 2 TEL: 0774-95-1301

Advanced Telecommunication Research Institute International
Spoken Language Translation Research Laboratories
2-2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-0288, Japan
Telephone: +81-774-95-1301
Fax: +81-774-95-1301

目次

1 はじめに

最近では，タグ付けされた中国語の構文木コーパスが利用可能になってきている．中国語は，文字種が多く，単語の分かち書きを行わないため，解析の困難な言語である．そのため，中国語解析器の構築には多大な労力を必要とする．そこで，本報告ではタグづきコーパスからの機械学習による中国語解析器を検討する．学習手法としては，近年注目を浴びている Support Vector Machine (SVM) を採用した．

2 中国語解析

2.1 中国語の形態素解析

形態素解析は，文の最小単位である形態素を認定し，品詞を付与する一連の処理である．多くの自然言語処理が形態素解析の結果に基づいて行われるため，極めて重要な処理と言える．

中国語は形態素解析の困難な言語である．それは，英語のような単語の分かち書きを行わないことや，基本的にすべての単語を漢字のみで表記することが，主な原因である．そのため，精度の良い中国語の形態素解析器を構築するためには，多大な労力を必要とする．

2.2 中国語構文木コーパス

中国語構文木コーパス (PCT) は， University of Pennsylvania (UPENN) の The Chinese Treebank Project [1]により作成されたタグ付きコーパスである．中国国営通信社である新華社通信 (Xinhua News Agency) の 1994 年から 1998 年の 325 記事 (約 10 万語) に対して，単語分割，品詞付与，構造付与を人手で行っている．

本稿では， LDC2000T48 を用いて実験を行った．本実験で用いた PCT のタグセットを，巻末の付録Aに示す．

3 SVM による Chunking

3.1 Support Vector Machine

SVM は，統計的学習理論に基づく 2 クラスのパターン認識手法である．従来の手法と比べて多くの面で優位性を示し，文字認識や画像認識など，様々な分野で応用されている．

SVM は，線形分離可能なパターン分布においては，2 次計画法を解くことによって識別境界のマージンを最大化し，最適な識別関数を得ることができる．線形分離が困難な場合にも，カーネルトリックと呼ばれる計算技術によって識別関数を非線形へと拡張することが可能である．また，入力次元数に依存しない極めて高い汎化能力を持つことが明らかになっている．

[1] http://idc.upenn.edu/ctb

3.2 SVM による Chunking

自然言語処理における Chunking とは，与えられた文を適当な解析単位に分割し，各要素にタグを付与する一連の処理を指す．例えば，単語の分かち書きや形態素解析，文節区切りなどが Chunking に含まれる．

工藤らは，SVM に基づく汎用的な Chunker として YamCha[2]を提案している．YamCha は Chunking を Tagging (タグ付与) と見なすため，一般的な Tagger として用いることが可能である．

SVM は 2 値分類器なので，タグ表現のような多値の分類問題を扱うためには，何らかの拡張を行う必要がある．YamCha では，“pairwise classification” と呼ばれる手法を採用している．これは，K クラスの分類問題に対して，クラス 2 つの組み合わせを分類する $K(K-1)/2$ 種類の分類器を作成し，それらの多数決でクラスを決定する手法である．

YamCha を用いた英文の Base Phrase Chunking 実験 [1] や，日本語の係り受け解析実験 [2] では，従来の学習モデルと比べて高い解析結果を示している．

4 YamCha による形態素解析

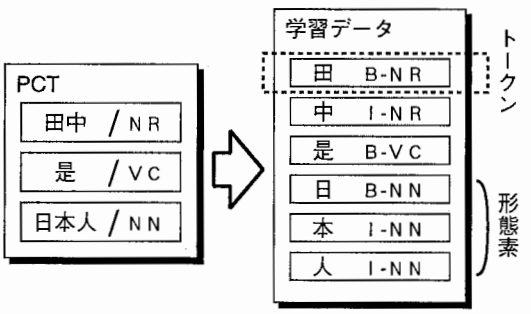
SVM を用いた Chunking により，中国語の形態素解析実験を行った．ここでは形態素を Chunk と見なし，1 文字が Chunk を構成する 1 要素に相当する．

Chunker には YamCha を，正解データとしては PCT をそれぞれ用いている．

4.1 学習データ

YamCha で扱う学習データは，複数のトークンと複数のカラムから構成される．ここでは，1 トークンが 1 文字に対応している．

第 1 カラム目には，形態素の要素である 1 文字を記述し，第 2 カラム目には，YamCha で推定するタグを記述する．このタグには，形態素の区切り位置を示す情報と，形態素に付与する品詞情報の両方が含まれている．



[2]Yet Another Multipurpose CHunk Annotator
http://cl.aist-nara.ac.jp/ taku-ku/software/yamcha/

形態素の区切り位置は，IOB2 タグによって表現した．これは，Chunk の先頭にあるトークンに B タグを付与し，Chunk に含まれる先頭以外のトークンに I タグを付与するルールである．Chunk 外のトークンには O タグを付与するが，本実験ではすべてのトークンが何らかの Chunk に含まれるため用いない．

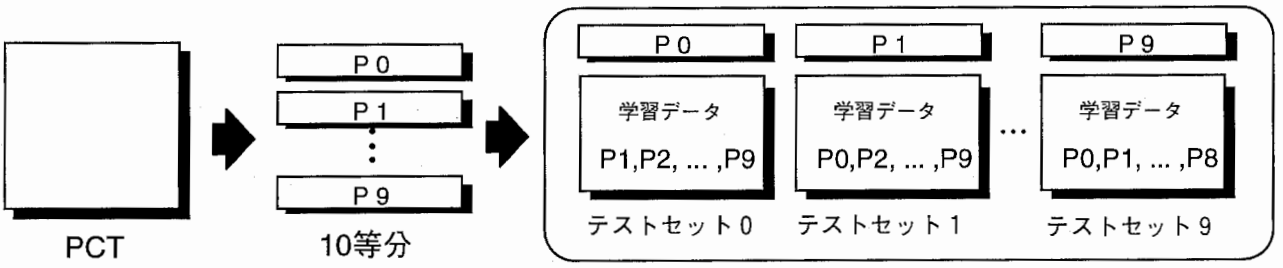
また，品詞情報は PCT のタグセットを用いた．品詞が “-NONE-” の形態素は構文構造上形式的に配置され，実体を持たないため，学習の対象外とする．

PCT のデータから YamCha の学習データへ変換するために，Perl でフィルタを作成した．これを巻末の付録Bに示す．

4.2　open test

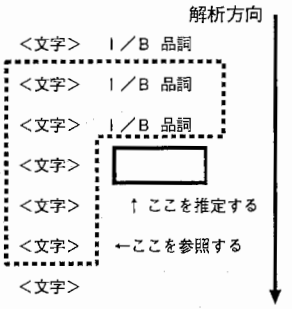
(4.2.1)　実験方法

PCT 全体を母集団とする 10-fold cross-validation の open test を行った．



PCT の 325 記事全体を無作為に 10 等分し，1 割をテストデータ，残りの 9 割を学習データとする．この方法で 10 組の異なるテストセットを作成した．

まず，YamCha に学習データを与え，SVM の学習モデルを作成する．その際，推定するトークンと，前方 2 トークン，後方 2 トークンの合計 5 トークンにおける文字情報を素性として学習した．また，前方 2 トークンの推定タグを動的素性とする．これは，解析中に得られたタグを動的に参照して次のタグを推定するための素性である．その他，すべての学習条件は YamCha の標準設定に従っている．



次に，得られた学習モデルを用いてテストデータを解析した．YamCha は，文字情報のみから，IOB タグと品詞タグを推定する．その結果を PCT と比較し，正解数を求めた．

以上の学習と解析を 10 組のテストセットすべてについて行い，正解率の平均を算出した．

(4.2.2) 実験結果

単語分割と品詞付与の正解率を表1に示す．誤り品詞の内訳は，表2に示す通りである．この表では，全誤り語のうち 16.3% が，VV と認定すべき品詞を NN と認定したために起こったことを表している．この実験では，NN-VV 間や，NN-NR 間の誤りが多いことがわかる．

表 1: open test の正解率

	正解数 / 推定数 = 正解率
単語分割	163195 / 171011 = 95.43 %
品詞付与	151413 / 171011 = 88.54 %

表 2: 誤りパターンの上位 5 種

正解 -⟩ 認定	出現率
VV -⟩ NN	16.3 %
NR -⟩ NN	11.1 %
NN -⟩ VV	10.3 %
JJ -⟩ NN	7.5 %
NN -⟩ NR	6.0 %

PCT 9 割の学習と 1 割の解析に要した処理時間などを表3に示す．基本的に，CPU : Pentium III 600 MHz，メモリ : 256 MB，OS : Linux の計算機を用いた．ただし，YamCha では学習後にモデルデータをコンパイルする必要があり，その際に大量のメモリを必要とする．この実験では，約 700 MB のメモリを必要としたため，コンパイル作業だけは，CPU : Pentium III 733 MHz，メモリ : 960 MB の計算機を使用した．

表 3: 形態素解析の処理時間

タグ	: 64 種類
トークン数	: 約 16 万
学習	: 約 6 時間
コンパイル	: 約 5 分
解析	: 約 35 分

4.3 closed test

PCT 全体を学習し，無作為に抽出した 1 割の記事郡を解析する closed test を行った．結果は，表 4，表 5の通りである．形態素分割，品詞付与とも 99% を越える高い精度を得ている．open test で目立った NN-VV 間の誤りは減少し， NN-NR が大部分を占めている．

表 4: closed test の正解率

	正解数 / 推定数 = 正解率
単語分割	18370 / 18394 = 99.87 %
品詞付与	18281 / 18394 = 99.39 %

表 5: 誤りパターンの上位 5 種

正解 -⟩ 認定	出現率
NN -⟩ NR	43.3 %
NR -⟩ NN	7.9 %
VV -⟩ NN	5.3 %
NT -⟩ NN	4.4 %
PU -⟩ NN	4.4 %

4.4 未知語の性質

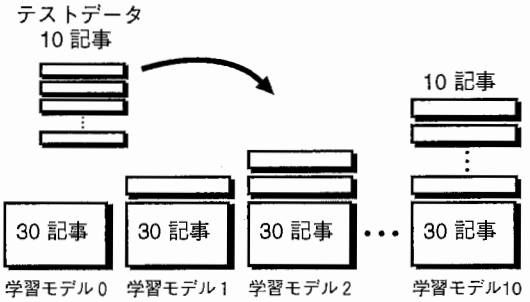
解析するテストデータに含まれる語のうち，学習データに含まれていないものを未知語と定義し，その性質を調べた．

未知語の占める割合は表 6の通りである．全体の 1 割ほどが未知語で，その半数ほどを誤って認定していた．また，全ての誤り語の 1/4 ほどが未知語であった．

表 6: 未知語の占める割合

	open test
未知語／全語	9.91 %
誤り未知語／未知語	48.79 %
誤り未知語／誤り語	24.44 %

未知語の性質をより詳しく調べるため，以下の実験を行った．まず，学習データとして 30 記事，テストデータとして 10 記事を，いずれも無作為に PCT から選択する．テストデータから順に 1 記事ずつ学習データに加えていき，それぞれの学習モデルでテストデータ 10 記事を解析した．



結果は，図1の通りである．未知語の量が増加するに連れて，ほぼ線形に正解率が低下していることがわかる．

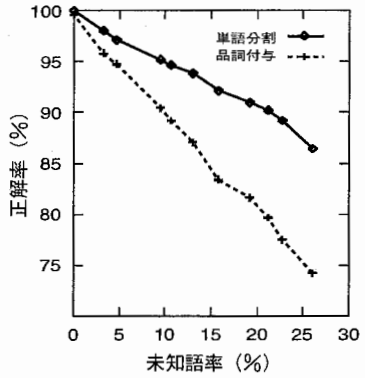


図 1: 未知語率と解析精度

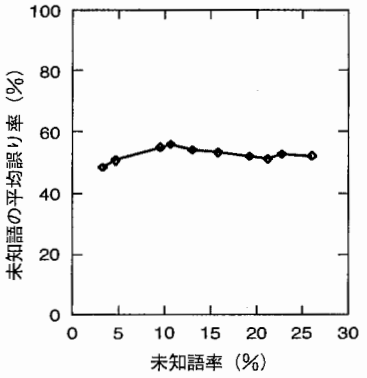


図 2: 未知語率と未知語誤り率

次に，未知語の量と，未知語全体に占める誤り語の割合との関係を調査した．結果は図2の通りである．未知語の量によらず，未知語の誤り率は 50 % 前後で一定であることがわかる．

4.5 誤り既知語

学習データ中に存在する複数の形態素が同じ文字列であるにも関わらず，それぞれ別の品詞をとる場合がある．これを，品詞付与の正解候補が複数存在すると言うことにする．NN（一般名詞）と NR（固有名詞）のように明確な区別をつけられない組み合わせが正解候補中に存在する場合は，学習データに表記の矛盾があると考えられる．

テストデータと学習データの両方に存在する語を既知語と定義する．品詞の認定を誤った既知語を正解候補に関して分類した．その内訳を表7に示す．

表 7: 誤り既知語の内訳

	A	B	C	合計
open test	355	11	40	406
closed test	32	2	15	49

A：正解候補が複数で，候補内の品詞が認定された語

B：正解候補が複数で，候補外の品詞が認定された語

C：正解候補が単一の語

open test では，Aが大部分を占めている．closed test では，Cの占める割合がより大きくなっている．正解候補が複数あるために起こる誤りは学習量を増やすにつれて減少すると考えられる．しかし，closed test においても，Cが存在しているので，ここに SVM の解析精度の限界があると考える．

さらに，closed test における全誤り語の内訳を表8に示す．

表 8: closed test における誤り語の内訳

A	B	C
NN -⟩ NR / 16	NT -⟩ DT / 1	JJ -⟩ NN / 5
NR -⟩ NN / 2	DT -⟩ NT / 1	PU -⟩ NN / 4
AD -⟩ JJ / 2		NN -⟩ JJ / 1
DEC -⟩ DEG / 2		CD -⟩ VV / 1
DEG -⟩ DEC / 2		PU -⟩ NN / 1
VV -⟩ NN / 2		VV -⟩ NN / 1
NN -⟩ VV / 2		NR -⟩ NN / 1
JJ -⟩ VA / 1		PU -⟩ AD / 1
CC -⟩ AD / 1		
AD -⟩ VV / 1		
NN -⟩ VA / 1		

Aに属する 32 語のうち，半数以上の 18 語が NN と NR に関する誤りである．これは，PCT における NN と NR のタグ付けの一部に矛盾があるためだと考える．その影響で，BやCの一部の語が誤った可能性もある．もし，学習データが矛盾を含んでいなかったとすると，closed test の誤りは少なくとも 1/3 以上減少すると予想する．

4.6 学習データ量による影響

学習データを1記事から30記事まで順に変化させ，学習データ量と学習／解析時間の変化を調査した．図3は，学習データの文字数と学習時間との関係を示している．ほぼ線形に増加していることがわかる．図4は，学習データの文字数と，無作為に選択した10 記事を解析したときの解析時間との関係を示している．線形に近い緩やかなカーブを描いている．

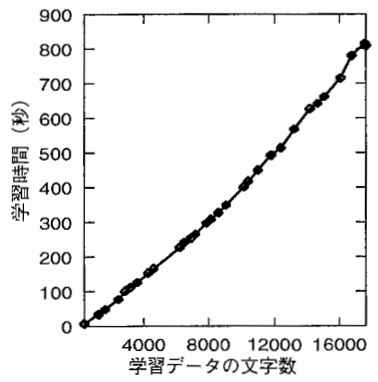


図 3: 学習データ量と学習時間

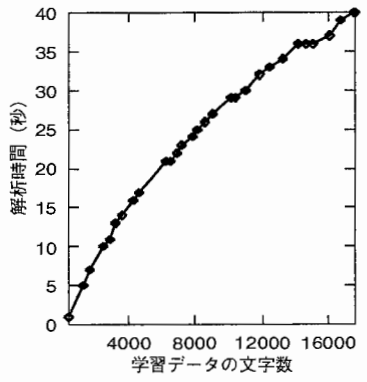


図 4: 学習データ量と解析時間

図 5は，図 4の実験における学習データ量と解析精度との関係を示している．学習量が少なくても比較的高い精度を示し，学習量を増やすにつれ精度が単調増加するが，やがてほぼ一定になる．

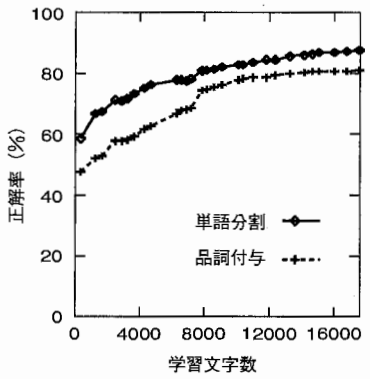


図 5: 学習データ量と解析精度

4.7 他の手法との比較

中国語を対象とした形態素解析器として，ATR の解析器が存在する．中国語の形態素辞書を持ち，PCT を学習した bi-gram を用いて解析を行う．

このシステムで PCT の 325 記事を解析した平均と，YamCha による解析器との比較を行う．結果を表 9に示す．括弧内の数値は，NN と NR 間の誤りを正解として扱ったときの正解率である．

なお，CPU : Pentium III 600 MHz，メモリ : 254 MB の計算機での解析時間の合計は，ATR 製解析器が約 9 時間，SVM の closed test で約 6 時間であった．

表 9: ATR 解析器との比較

	単語分割	品詞付与(NR 無視)
ATR 解析器	96.30%	92.57% (92.84%)
SVM open	95.43%	88.54% (90.51%)
SVM closed	99.87%	99.39% (99.70%)

4.8 考察

最も目立った誤りは NN と NR に関するものであった．これは人手でも明確な区別を付けることはできず，PCT 内でも揺れがある．また，形態素解析結果の応用次第では，区別を付ける必要のない場合も考えられる．

NN と NR 間の誤りを無視した場合には，SVM による open test の品詞付与正解率が 2% ほど上昇し，ATR 解析器に匹敵する精度を得ることができる．正解データさえ用意されていれば SVM による形態素解析器の開発は容易なので，本手法の有効性が示されたと言える．

closed test では 99% を越える高い精度を得た．誤りの多くも PCT 内の矛盾した記述に基づいている．しかし，品詞が唯一である語や，NN と VV のように文法上の役割から品詞の特定が可能なものも誤りに含まれていた．本実験での SVM による形態素解析は，ここに限界があることがわかる．

解析の正解率は未知語の数と密接に関係しているので，可能な限り多くの学習データを収集することが望ましい．しかし，学習量の増加に応じて解析時間も線形に近い形で増加してしまうことが欠点である．

解析時間は ATR 製解析器の 2/3 ほどであった．しかし，SVM による解析時間はさほど改善の余地がないのに対し，ATR 製の解析器は処理時間を大幅に短縮する余地がある．また，SVM の学習モデルに対して，後から特定の語を追加することはできない．情報を加える場合は，それを含めたすべてのデータを学習し直す必要がある．

以上より，SVM を用いた実用的な形態素解析器を構築するには，形態素辞書など他の手法を併用することが望ましい．SVM では，未知語の約半数を正しく認定しているので，辞書に登録されていない語を補佐的に解析したり，辞書によって分割された形態素に，SVM が品詞付与を行うなどの方法が考えられる．

5 YamCha による Phrase Chunking

5.1 学習データ

SVM による Base Phrase の Chunking 実験を行った．形態素解析と同様に，Chunker として YamCha を，正解データとして PCT を用いている．

PCT が表現する構文木では，葉の部分が形態素に相当する．ここでは，葉に最も近い位置に付与されている Base Phrase (基本句) を 1 つの Chunk とし，形態素がそれを構成する要素に該当する．そして，Base Phrase の区切り位置と Phrase タグを推定する．

PCT から YamCha の学習データへ変換するために用いた Perl のフィルタスクリプトを，巻末の付録Cに示す．

5.2 実験方法と結果

形態素解析と同様に，10-fold cross validation による open test を行った．学習の素性は，形態素の文字列情報，品詞情報，Chunk の区切りを示す IOB タグと Chunk の属性を示す Phrase タグである．すべての学習条件は YamCha の標準設定に従っている．

open test の結果を表 10に示す．また，タグ付与における誤りのうち出現率の大きい上位 5 種を表 11に示す．分割，タグ付与とも比較的高い精度を得られることがわかった．テストセット 1 つあたりの学習／解析時間は，表 12の通りである．計算機は形態素解析の実験と同じものを使用した．

表 10: Base Phrase Chunking の解析結果

	正解数 / 推定数 = 正解率
分割	92431 / 95798 = 96.49 %
タグ付与	94200 / 95798 = 98.33 %

表 11: 誤りパターンの上位 5 種

正解 -⟩ 認定	出現率
IP -⟩ VP	34.7 %
VP -⟩ IP	29.8 %
NP -⟩ IP	6.4 %
IP -⟩ NP	4.8 %
VP -⟩ NP	3.6 %

5.3 学習条件による変化

学習条件を変化させた場合の精度を調査した．ここでは，PCT の 33 記事を学習し，別の 33 記事を解析した．結果を表 13に示す．

語彙を素性とせず，品詞だけで学習した場合でも，正解率の低下は少ない．Base Phrase の推定は，大部分が品詞情報に依存していることがわかる．また，学習方向を逆転した場合や，素性の窓を変化させた場合は，いずれも正解率が低下している．これらの条件の範囲では，標準の設定による学習が最良であると言える．

表 12: Base Phrase Chunking の処理時間

タグ	: 32 種類
トークン数	: 約 9 万
学習	: 約 2 時間
コンパイル	: 約 1 分
解析	: 約 3 分

表 13: 学習条件による解析結果の相違

学習条件	分割正解率	タグ正解率
指定なし（窓 2）	94.60%	97.65%
品詞のみ学習	-3.12%	-1.42%
学習方向を逆転	-0.24%	-0.13%
素性の窓を前後 4	-1.10%	-0.48%
素性の窓を前後 3	-0.48%	-0.22%
素性の窓を前後 1	-0.14%	-0.04%

5.4 考察

Base Phrase Chunking は，形態素解析と比べて高い解析精度を得た．また，学習／解析に要する時間も短いことがわかった．これは，入力トークン数や，推定するタグの種類が少ないためである．

学習条件は，YamCha の標準設定が最も良好な結果を得た．ただし，Base Phrase の種類によって Chunk の平均長が異なると考えられる．特定の Phrase に限定した解析では，素性の範囲を変化させた方が良い結果を得られる可能性もある．

工藤らは，SVM に基づく Chunking の段階適用が，日本語の係り受け解析に有効であることを示している [2]．本実験の手法を段階的に適用することで，比較的精度の高い中国語の構文解析が可能であると予想できる．しかし，これらは形態素解析の結果に基づく処理なので，より高精度の形態素解析器を構築することが優先課題である．

6 むすび

本報告では，SVM を用いた Chunking による中国語解析を検討した．解析精度は比較的高く，形態素解析において，未知語の半数を正確に認定するなどの長所もある．欠点としては，処理時間や学習モデルの扱いにくさが挙げられる．他の手法と組み合わせることで，実用的な解析器の構築が可能だと考える．

参考文献

[1] 工藤拓, 松本裕治. Support Vector Machine を用いた Chunk 同定. 情報処理学会研究報告書 2000-NL-140, pp.9-16 (2000).

[2] 工藤拓，松本裕治．チャンキングの段階適用による係り受け解析. 情報処理学会研究報告書 2001-NL-142, pp.97-104 (2001).

付録A：PCT のタグセット

品詞：

AD	adverbs
AS	aspect marker
BA	把 in ba-const
CC	coordinating conj
CD	cardinal numbers
CS	subordinating conj
DEC	的 for relative-clause etc.
DEG	associative 的
DER	得 in V-de const. and V-de-R
DEV	地 as the head of DVP
DT	determiner
ETC	tags for 等 and 等等 in coordination phrases
FW	foreign words
IJ	interjection
JJ	noun-modifier other than nouns
LB	被 in long bei-construction
LC	localizer
M	measure word (including classifiers)
MSP	some particles
NN	common nouns
NR	proper nouns
NT	temporal nouns
OD	ordinal numbers
ON	onomatopoeia
P	prepositions (excluding 把 and 被)
PN	pronouns
PU	punctuation
SB	被 in chort bei-construction
SP	sentence-final particle
VA	predicative adjective
VC	copula 是
VE	有 as the main verb
VV	other verbs

Phrase :

ADJP	adjective phrase
ADVP	adverbial phrase headed by AD (adverb)
CLP	classifier phrase
CP	clause headed by C (complementizer)
DNP	phrase formed by ‘‘XP + DEG’’
DP	determiner phrase
DVP	phrase formed by ‘‘XP + DEV’’
FRAG	fragment
IP	simple clause headed by I (INFL)
LCP	phrase formed by ‘‘XP + LP’’

LST	list marker
NP	noun phrase
PP	preposition phrase
PRN	parenthetical
QP	quantifier phrase
UCP	unidentical coordination phrase
VP	verb phrase

付録B： PCT から形態素を抽出する perl スクリプト

```
while( ⟨⟩ ){

    chomp;
    next unless(/\)\s+$/);
    $flag=0;

    if( /\) \)/ ) {  # おわり？
        $flag=1;
    }

    /([^\)]+)\(([^\(\)]+)\)(\)*)/;

    $the_word=$2;
    next unless ( $the_word =~ /^([^\-]+)\-?[^\-]* (.+)/ );
    $pos = $1;
    $mor = $2;

    @chars = $mor=~ /[\200-\377].|\S|\s/g;
    print "$chars[0] B-$pos\n";
    shift @chars;
    foreach $line (@chars) {
        print "$line I-$pos\n";
    }
    print "\n" if ($flag);
}
```

付録C： PCT から Base Phrase を抽出する perl スクリプト

```
@dam = ( 'ADJP', 'ADVP', 'CLP', 'CP', 'DNP', 'DP', 'DVP', 'FRAG',
  'IP', 'LCP', 'LST', 'NP', 'PP', 'PRN', 'QP', 'UCP', 'VP' );
$tags{$_}=1 foreach( @dam );

while( <> ){

    chomp;
    undef @stk if( /^\(/ );
    next if( /^</ || /^$/ );

    # 語彙や品詞を切り出す
    if ( /^(.*)\(([^\(\s]+)\s+([^\)\s]+)([\)\s]+)$/ ){
        ( $front,$word,$paren ) = ( $1,$3,$4 );
        $none = ( $2 eq "-NONE-" ? 1 : 0 );
        $2 =~ /^([^\-]+)/;
$pos = $1;
$none = ( $word =~ /^\*/ ? 1 : 0 );
$badline=1 if( $paren !~ /\)/ );
    } else {
        $badline=1;
    }

    if( $badline ){
$badline=0;
$line = $_;
$nline = <IN>;
chomp $nline;
$nline =~ s/^[\s\t]+//;
$line .= $nline;

$line =~ /^(.*)\(([^\(\s]+)\s+([^\)\s]+)([\)\s]+)$/;
( $front,$word,$paren ) = ( $1,$3,$4 );
$none = ( $2 eq "-NONE-" ? 1 : 0 );
$2 =~ /^([^\-]+)/;
$pos = $1;
$none = ( $word =~ /^\*/ ? 1 : 0 );
    }
    $front =~ s/^\($/\( /;
    $paren =~ s/\s//g;

    # 非終端記号をスタックに追加
    $size = $#stk;
    @sped = split( /\(/ ,$front );
    if ( $#sped>0 ){
foreach $part ( @sped ){
    $part =~ s/\s+//g;
```

```
    push( @stk, $part ) if( $part );
}
    }

    # 採用する非終端記号を決める
    $iob='O-';
    for ( $i=$#stk; $i>=0; $i-- ){
        $stk[$i] =~ /^(\w+)/;
if ( $tags{$1} ){
    $iob = 'I-';
    $iob = 'B-' if ( $i>$size || $pr ne $1 );
    $pr = $1;
    last;
}
    }

    $frag=1;
    foreach ( @stk ){
$frag=0 if( $_ =~ /^FRAG/ );
    }
    if ( $frag && $#stk>=0 ){
if ( $flag ){
    print "\n";
    $flag=0;
}
if ( !$none ){
    print "$word\t$pos\t$iob$pr\n";
}
    }

    # 閉じ各個の数だけ非終端記号を消去
    $del_begin = $#stk-length($paren)+2;
    $del_begin = 0 if( $del_begin<0 );
    splice( @stk, $del_begin );
    $flag=1 if( $#stk<0 );
}
```

付録D： 学習データサンプル

形態素解析：

中 B-NR
国 I-NR
最 B-JJ
大 I-JJ
氨 B-NN
纶 I-NN
丝 I-NN
生 B-NN
产 I-NN
基 B-NN

Base Phrase Chunking：

旅日	JJ	B-ADJP	
华侨	NN	B-NP	
和	CC	I-NP	
台湾	NR	B-NP	
同胞	NN	B-NP	
２０００多	CD	B-QP	
人	NN	B-NP	
今晚	NT	B-NP	
在	P	B-PP	
中国	NR	B-NP	
驻日	JJ	B-ADJP	
使馆	NN	B-NP	
欢聚一堂	VV	B-VP	
，	PU	I-VP	
喜迎	VV	B-VP	
羊年	NN	B-NP	
新春	NN	I-NP	
佳节	NN	I-NP	
。	PU	B-IP	