形態素解析結果の視覚化システム ViJUMAN とその学習機能

山下 達雄，松本 裕治
奈良先端科学技術大学院大学 情報科学研究科
{tatuo-y,matsu}@is.aist-nara.ac.jp

形態素解析結果の曖昧性をわかり易くグラフ状に表示するシステム ViJUMAN を開発した．マウス操作のみで適切な単語の選択・品詞の修正ができ，タグ付きコーパス作成に有用なシステムである．この ViJUMAN を用いたタグ付きコーパス作成作業時の問題点として一度修正したパス間違いが別の文で繰り返し起こるというものがある．これを防ぐために，一度パスを修正したらその部分を事例データとして保存しておき，以降の解析で同じパス間違いが起こったときに自動的にパスを修正する処理をシステムに付加した．

[キーワード] 形態素解析，品詞タグ付きコーパス，支援システム，学習，GUI

Visual Interface for Morphological Analysis System ViJUMAN and its Learning Facility

Yamasita Tatuo, Matsumoto Yuji
Graduate School of Information Science, Nara Institute of Science and Technology

We developed the ViJUMAN system which is the visual interface for the Japanese morphological analyzer, JUMAN. This system visualizes the output of JUMAN system with a graphical viewer, and allows users to easily correct morphological analysis errors with quick mouse operations. It enables users to efficiently construct a part-of-speech tagged corpus. Sometimes, some morphological analysis errors tend to recur even after users correct them. To solve this problem, we added the facility that memorizes each correction of morphological analysis errors and, when the same errors occur again, corrects them automatically referring to the memorized correction.

[keyword] morphological analysis, tagger, part-of-speech tagged corpus, GUI

1 はじめに

近年，日本語テキストの電子化が急激な勢いで進んでいるが，大部分が加工されていない生のテキストデータである．自然言語処理に限らず様々な分野で基礎データとして品詞タグ付きコーパスの需要は高く，迅速に整備していく必要がある．

しかし現在の形態素解析技術は完全ではないので，精度の高い品詞タグ付きコーパスを作るには，最終的には人手による修正が不可欠である．そのためにも作業効率の良い支援システムが必要となる．

今までに，品詞タグ付けコーパスの作成支援に用いられてきたシステムは，テキストエディタベースの簡易的なものが多い．例えば，Penn Treebank[7] 作成時に用いられたものは GNU Emacs ベースのマウス操作によるユーザインターフェースである．これは一通りの形態素解析結果を表示し，品詞の誤りがあれば作業者がその単語をマウスで選び正しい品詞を打ち込み修正するいう単純なものである．

しかし，このようなシステムは，単語の品詞の修正のみが目的であり，日本語のようにわかち書きのされていない言語には不可欠の，単語境界の修正が容易ではない．また品詞情報の修正という点から見ても到底使いやすいとは言えない．

そこで我々は，使いやすさに重点を置き，品詞タグ付けコーパスの作成支援システム ViJUMAN[3] を開発した．

次節では，その ViJUMAN について解説する．

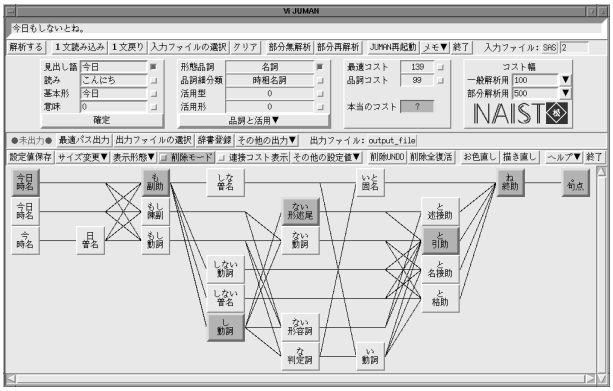

図 1: ViJUMAN

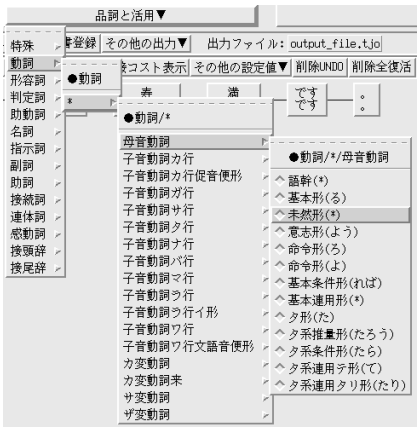

図 2: JUMAN の文法体系に沿った品詞情報の選択

そして第 3 節では,「同じ修正を何回もしなくてはならない」という ViJUMAN によるコーパス作成作業時の不満を解消するために新たに付け加えた学習機能について説明する.

2 ViJUMAN

ViJUMAN とは JUMAN[1] による形態素解析結果を視覚化し品詞タグ付きコーパスの作成支援を行うシステムである. 図 1に実行画面を示す. システムの特徴を以下に挙げる.

- JUMAN の解析結果をグラフ状に図示することにより曖昧性が視覚的に理解できる.
- グラフ状に図示された解析結果からマウス操作のみで, 任意にパスを選ぶことができる. さらに文の一部だけ制約を緩めて再び解析する機能も実装されており, 最初の解析結果で得られなかった形態素を出現させることもできる.
- メニュー選択方式により, JUMAN の文法体系に基づき矛盾なく品詞情報を選択することができる (図 2). これにより容易に新単語の登録・品詞や活用の修正ができる.
- 前記の方法により品詞情報を修正した単語をボタン一つで JUMAN のユーザ辞書へ登録でき, 次の解析から反映させることができる.

機能の詳細については文献 [3] を参照されたい.

3 学習機能

3.1 目的

ViJUMAN を品詞タグ付けコーパス作成支援システムとして利用する際の不満として, 一度行ったパス修正を別の文についても繰り返し行わなくてはならないという点が挙げられる. 例えば,「…によると,」の解析結果「…-に (格助詞) -よる (時相名詞) -と (格助詞) -,」を「…-に (格助詞) -よる (動詞) -と (格助詞) -,」に修正しても, 次にまた「…によると,」を含む文を解析すると「…-に (格助詞) -よる (時相名詞) -と (格助詞) -,」となってしまい再度の修正が必要となる. これは作業者にかなりの心理的負担を与える.

新聞を解析していくと, 十数文に一回という頻度でこのような事態が生じる. これに対処するために, 修正したパスを事例データとして蓄積しておき, それらを用い JUMAN の解析結果を自動的に修正にするという処理を行う.

3.2 処理の流れ

ViJUMAN での品詞タグ付けコーパス作成作業の流れは大体以下のようなものである.

1. 一文を JUMAN で解析する.
2. JUMAN の解析結果が束図で表示され, コスト計算による最適パスが強調される.
3. 人間が束図をマウスでクリックし, パスを修正する.
4. 修正したパスを品詞タグ付きコーパスのデータとして出力する.

本研究では, まず (4) の段階で, JUMAN が選択した最適パスと作業者が修正したパスの差分をとることによって, 修正した部分 (形態素列) を取り出し事例データ (正例) として蓄積していく. そして (1) の段階の直後に, 事例データをもとに解析結果を修正する. これにより一度人手で修正すれば, 以降, 同じ事例が現れたときに最適パスが自動的に修

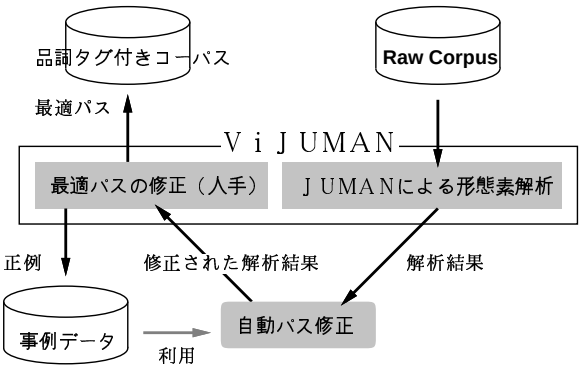

図 3: 処理の流れ

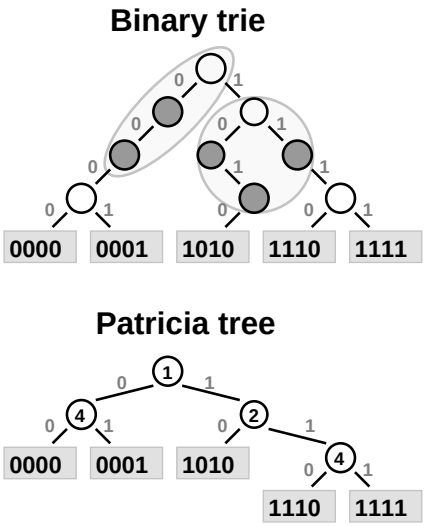

図 4: パトリシア木

ノード上の数字は何ビット目をチェックするかを表す.

正されるので，人手による作業が軽減される．図 3 に全体の処理の流れを示す.

本研究では，事例データの蓄積・検索にパトリシア木を用いている．パトリシア木とは，ビット単位のトライ中で枝が 1 本しかない無駄なノードをまとめて圧縮した効率の良いデータ構造である (図 4). 詳しくは，文献 [4][5] を参照されたい.

3.3 正例の蓄積

最適パスと修正したパスの差分をとることにより得られる正例を事例データとして蓄積する.

例えば,「先日、彼と食べた。」という文の場合，図 5のような手順で正例を得る．そして，正例部分

元の最適パス

先日	せんじつ	先日	名詞	時相名詞	*	*
、	、	、	特殊	読点	*	*
彼	かれ	彼	名詞	普通名詞	*	*
と	と	と	助詞	引用助詞	*	*
食べた	たべた	食べる	動詞	*	母音動詞	タ形
。	。	。	特殊	句点	*	*

修正したパス

先日	せんじつ	先日	名詞	時相名詞	*	*
、	、	、	特殊	読点	*	*
彼	かれ	彼	名詞	普通名詞	*	*
と	と	と	助詞	格助詞	*	*
食べた	たべた	食べる	動詞	*	母音動詞	タ形
。	。	。	特殊	句点	*	*

⇓

差分

と	と	と	助詞	引用助詞	*	*
と	と	と	助詞	格助詞	*	*

正例

修正したパス側の差分に前後 1 つずつの形態素を含めたもの.

彼	かれ	彼	名詞	普通名詞	*	*
と	と	と	助詞	格助詞	*	*
食べた	たべた	食べる	動詞	*	母音動詞	タ形

⇓

蓄積するデータ

キー	彼と食べた
内容	A1B034 (コード化された正例)

図 5: 正例の蓄積

の表層文字列をキー，正例を固定長ビットでコード化したものを検索される内容としてパトリシア木に入れていく.

3.4 最適パスの自動修正

JUMAN での形態素解析後に，最適パスの自動修正を行う.

解析した文に正例と同じ文字列が存在し，JUMAN の解析結果のグラフ中に正例と同じ形態素列がある場合，その形態素列を最適パスに含めるよう形態素解析結果を修正する．解析した文に正例と同じ文字列が存在するが正例と完全にマッチする形態素列がない場合，後述 (3.4.2) の条件にあえば，その正例の形態素を作り，最適パスに含めるよう形態素解析結果を修正する.

3.4.1 パトリシア木からの正例の探索

形態素解析結果のグラフから正例と同じ形態素列を探す第一段階である．全ての正例に対して単純にマッチングを行うと計算量がかかりすぎるため，まず，ありそうな正例を絞り込むという処理を行う.

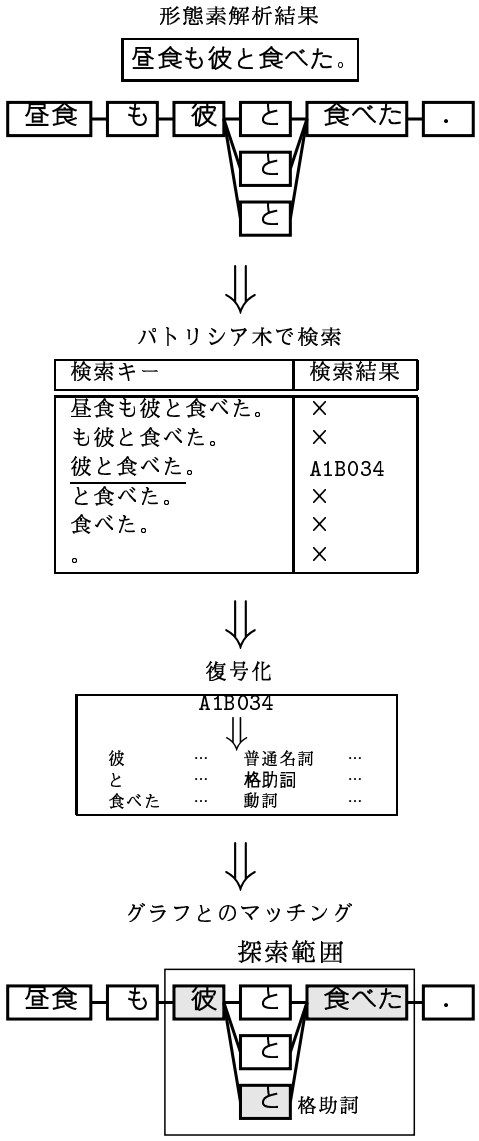

図 6: 正例のマッチング

解析した文の先頭からポイントを次の単語区切りの場所までずらしながら，そのポイントから始まる文字列でパトリシア木を探索する (図 6).

以下に述べる方法[1]により，解析した文のそのポイントから始まる全ての正例候補をパトリシア木から引き出すことができる．

1. コード列をビット列とみなし，パトリシア木を探索する．
2. 途中に敷居ビット (この場合，16 の倍数) をチェックするノードがあれば，そこの左 (0) 側のノードのデータを取り出し，検索ビット列と矛盾が無いかチェックする．矛盾があれば終了する．
3. いきどまりならば，データを取り出し，検索ビット列と矛盾が無いかチェックする．

[1]これはトライと同様に，形態素解析の辞書引きに非常に効率の良い方法であり，現在，我々が開発している JUMAN の次期バージョンにも採用されている．

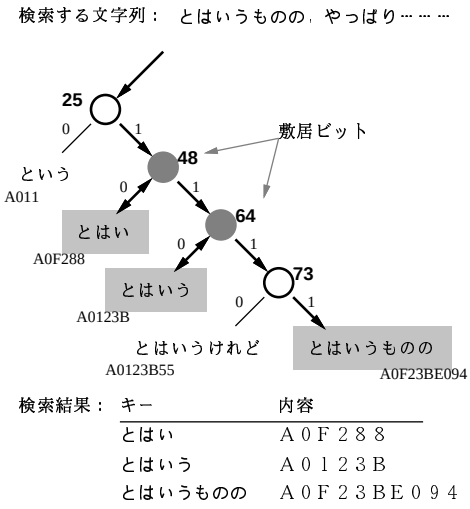

図 7: パトリシア木による正例の検索

図 7に例を示す.

3.4.2 形態素解析結果からの正例の探索とコスト修正

第二段階として，パトリシア木から引き出した正例候補とマッチする形態素を形態素解析結果のグラフから捜し出す．

さきほどの文字列マッチングでマッチした範囲 (何文字目から何文字目か) から，グラフの中での探索範囲を計算し，その範囲内で形態素のマッチングを行う (図 6)．マッチングに成功した場合，それらの形態素同士の連接コストや品詞コストを 0 にする．また，全ての正例候補とのマッチングに失敗した場合，(最長文字列の) 正例の両端の形態素が探索範囲の両端に存在すれば，新たにその正例と同じ形態素を作り，それらの連接コストや品詞コストを 0 にする．この処理により，最適パスに必ず正例の形態素が含まれることになる．

3.5 実験

3.5.1 実験方法

新聞コーパスのテキストデータと品詞付きデータを用い，品詞タグ付け作業のシミュレーションを行い，パスの修正数をカウントした．修正数とは，

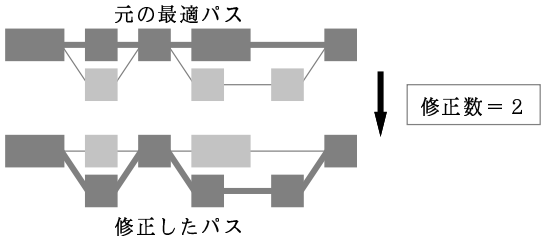

図 8: 修正数の数え方

実験 1 : 日経新聞 600 文

	学習なし	学習あり
修正数	1009	959
再修正数	51	0

自動パス修正の回数	65
修正数の減少分	50

蓄積された正例	959
利用された正例	38

実験 2 : 日経新聞 300 文

	学習なし	学習あり
修正数	535	520
再修正数	16	0

自動パス修正の回数	20
修正数の減少分	15

蓄積された正例	520
利用された正例	16

表 1: 実験結果

実験 1 と実験 2 は記事データ及びタグ付け作業者が異なる.

図 8に示す通り，元の最適パスと修正後のパスの差分の数である．シミュレーションには ViJUMAN のプログラムから GUI の処理を削ったものを用いている．

3.5.2 実験結果と考察

表 1に実験結果を示す．再修正数とは一度修正したパスをまた修正する回数である．

どちらの実験でも自動パス修正の回数に比べ学習による修正数の減少分の方が少なくなっている．これは，自動パス修正が修正数減少に寄与しない，つまり，自動パス修正の結果，別のパス修正が必要となることがあるからである．

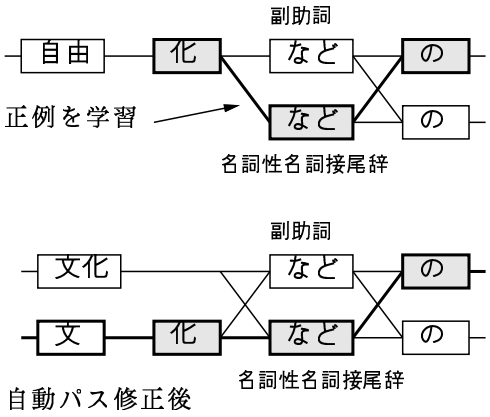

図 9: 自動パス修正が修正数減少に寄与しない例

例えば，図 9に示すように,「自由化などの」の解析結果から「化-など (名詞性名詞接尾辞) -の」を学習すると,「文化などの」の解析結果を自動パス修正したとき「文-化-など-の」が最適になってしまい，結局「文化-など-の」に修正しなければならない．

また，解析結果の同じ部分に影響する複数の正例が蓄積されると，自動パス修正が意味をなさなくなるという問題がある．これを「ゆれ」と呼ぶことにする．

例えば,「太郎とみた」の「と」は,「私の勘では真犯人は太郎とみた。」と「ＴＶを太郎とみた。」では違う．前者では「太郎-と (引用助詞) -みた」，後者では「太郎-と (格助詞) -みた」となる．このような形態素列がどちらも正例として蓄積されると，自動パス修正は機能しなくなる．

しかし，各形態素の表層レベルまで考慮して正例を蓄積していくので,「ゆれ」は非常に頻度の低い現象と思われる．実際，この実験に先立って行われた試験実験では，全ての「ゆれ」はタグ付け間違いによるものであった．本研究の実験は，そのタグ付け間違いを全て正したものを用いて行っている．

3.5.3 正例データの利用

今回の実験では蓄積された正例のほとんど (約 96%) は利用されていない．これは副作用を避けるために表層レベルまで見て蓄積しているためで，無駄が多くなってしまうのは避けられない．利用された正例のうち頻度の高いものを表 2に挙げる．

利用頻度にかかわらず正例データを整理し抽象化することによって，有用なデータとすることができる．一つの方法は，先頭と末尾以外の形態素が共通である正例のそれぞれの先頭と末尾の形態素を品詞レベルで抽象化するというものである．例えば,「みる-と (述語接続助詞) -、」「する-と (述語接続助詞) -、」

に	よる (動詞)	と (述語接続助詞)	、
(文頭)	一方 (接続詞)		、
みる	と (述語接続助詞)		、
する	と (述語接続助詞)		、
した	こと (名詞)	に (格助詞)	ついて
した	もの (名詞)	で (判定詞)	、
	⋮		

表 2: 使用頻度の高い正例

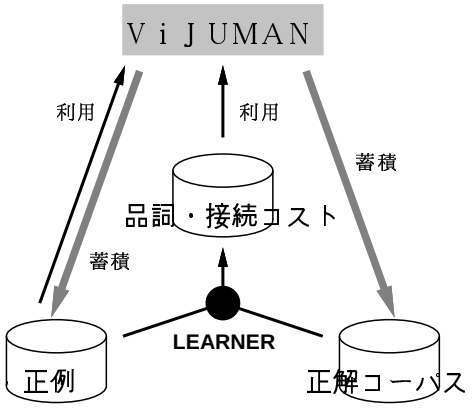

図 10: 正例と正解コーパスからの品詞・連接コストの学習

「買う-と (述語接続助詞) -、」などがあった場合，「動詞/基本形 -と (述語接続助詞) -、」と整理できる．

これらのデータは JUMAN が最適パスの選択をたまたま間違えたために得られたものであり，統計的に普遍的な価値は無いが，JUMAN での解析の弱点を表すデータとして利用できる．

例えば，今までの連接規則に加え形態素列を連語として登録・利用できる JUMAN3.0Beta[2] で，大量の整理された正例を連語として用いることによって，かなりの形態素解析精度が期待できると思われる．

また，正例の蓄積だけでなく，ある程度品詞タグ付きコーパスが蓄積されたら bigram 統計をとり品詞・連接コストを更新する処理も並行して行うつもりである (文献 [6] の 2.3.1 を参照)．その際に，整理された正例も bigram 統計のデータとして利用することで，その時点での解析の弱点を早く克服できるように品詞・連接コストが更新されていくと期待できる．図 10に概観を示す．

4 おわりに

本研究では，品詞タグ付きコーパス作成時の作業負担を事例データの学習により軽減できることを示した．また，実験過程から，今回の手法は作業負担の軽減だけでなく，品詞タグ付きコーパスの一貫性チェックにも利用できることが明かになった．今後は，事例データと品詞タグ付きコーパスの両方を用いた品詞・連接コストの学習や，作業時にタグ付けの一貫性を保つために過去に作成した品詞タグ付きコーパスを容易に参照できる機能等，コーパス作成支援のためのさらなる研究を続けていく予定である．

本研究及び ViJUMAN に関する情報・ソース・マニュアル [3] 等は以下の URL にて入手できる．

http://cactus.aist-nara.ac.jp/student/tatuo-y/vijuman.html

謝辞: 本研究では，日経新聞 CD-ROM 94 年版を利用した．新聞記事データの研究利用許諾を頂いた日経新聞社に感謝する．

参考文献

[1] 松本裕治・黒橋禎夫・宇津呂武仁・妙木裕・長尾真, 日本語形態素解析システム JUMAN 使用説明書 version 2.0, NAIST Technical Report, NAIST-IS-TR94025, July 1994.

[2] 黒橋禎夫・山地治・大石巧・坂口昌子, “付録 G: JUMAN 2.0 から JUMAN 3.0 Beta への変更点”, 日本語形態素解析システム JUMAN 使用説明書 version 3.0 Beta, July 1996.

[3] 山下達雄・松本裕治, 形態素解析視覚化システム **ViJUMAN** 使用説明書 version 1.0, NAIST Technical Report, NAIST-IS-TR96005, February 1996.

[4] 菊田昌弘, “用語解説：パトリシアツリー (Patricia Tree)”, 人工知能学会誌, Vol.11, No.2, pp.337-339, 1996.

[5] R. Sedgewick 著 野下浩平・星守・佐藤創・田口東 共訳, アルゴリズム (Algorithms) 原書第 2 版 第 2 巻 探索・文字列・計算幾何, 近代科学社, 1992.

[6] 竹内孔一, 隠れマルコフモデルによる日本語形態素解析システムのパラメータ推定, 奈良先端科学技術大学院大学修士論文, NAIST-IS-MT9451067, Aug 1995.

[7] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz: Building a Large Annotated Corpus of English:The Penn Treebank, Computational Linguistics, Vol.19, No.2, pp.313-330, Jun 1993.