自然言語処理 117－9
（1997. 1. 20）

保守性を考慮した日本語形態素解析システム

渕武志　松岡浩司　高木伸一郎
ＮＴＴ情報通信研究所
{fuchi,matsuoka,takagi}@isl.ntt.co.jp

抄録

語の共起を用いて同形語の読み分けを行う日本語形態素解析システムを開発した。必要な語の共起は、システムが解析を誤る文から取得することができる。語の共起は人にとって識別しやすいため、間違えにくく、確実にシステムの精度をあげることができる。平均31文字で構成される漢字仮名混じり文の91.9%が本システムにより正しい仮名文に変換された。これは、他の広く使われているシステムによるものよりも18%以上良い結果である。

Easily Adjustable Japanese Morphological Analyzer

Takeshi Fuchi, Kouji Matsuoka, Shinichiro Takagi
NTT Information and Communication Systems Laboratories
{fuchi,matsuoka,tagagi}@isl.ntt.co.jp

Abstract

We developed a Japanese morphological analyzer which uses the co-occurrence of words to select a correct sequence of words in an unsegmented Japanese sentence. The co-occurrence information can be obtained from cases where the system analyzes sentences wrong. As the information increases, the accuracy of the system increases with a small risk of being degraded. Our system assigns the correct phonological representations to 91.9% of Japanese sentences which consist of an average of 31 characters. This accuracy is over 18% better than that of other popular systems.

1. はじめに

テキストを対象とする自然言語処理において、形態素解析処理は大きな役割を担っており、これまでも多くのシステムが作られてきた。日本語の形態素解析システムの詳細が公表されることは少ないが、その精度は語単位で概ね97 ～ 99 %程度であると言われている[1]。このレベルの精度は80年代の前半にはすでに可能となっていたが、その後、今日まで大幅な精度向上は報告されていない。これは、システムが用いる文法や辞書のデータを、精度向上のために調整することが困難な点に原因があると我々は考えている。従って、本システムではこの点を解消することに主眼を置き、精度向上への道を拓くことを目標とした。

我々の方針は、従来の方法による精度を生かしつつ、個々の解析失敗事例に照らして、より具体的な調整ができるようなシステムを作ることである。"より具体的な調整"とは、"より細かい調整"とは異なるものであり、調整する者にとって直観的に理解しやすく、個々の語を対象とするような調整である。このような調整は、一つ一つの調整の効果が確実であるが小さいため、手数とし

ては大きなものになる。しかし、作業自体は単純で容易であるから、人海戦術を用いることができる。

次節では、従来の日本語形態素解析システムの精度の飽和の原因について、筆者の経験に基づいて考察する。3節で、本システムの概要を述べる。4節では、本システムの評価として、人手による調整に伴う精度の推移と、他のシステムとの比較について述べる。

2. 精度の飽和に関する考察

従来の日本語形態素解析システムの大半は概ね次のような構成になっている。

(1) 入力された文字列から部分文字列を取り出し、これをキーとして辞書を検索し、各語についての情報を引き出す。この情報の中には品詞や読みの情報を含み、活用のある品詞の場合には活用形の情報なども含まれる。表記が同じ複数の語が検索された場合には、それぞれを個別に記録する。これを可能な全ての部分文字列について行う。辞書に登録されていない部分文字列は未登録語として記録する。

(2) 入力文字列中で隣り合っている部分文字列に関する情報を参照し、それらが文法に照らして隣接可能であるかを調べる。これによって文頭から文末まで隣接可能な語の連鎖を求める。この時、一般には可能な連鎖は複数ある。

(3) 可能な連鎖の間に、何らかの方法によって優先順位を付け、もっとも優先順位が高いものを解とする。

以上の処理の内、(3)の優先順位付けの方法には以下のようなものが考えられる[2]。

- 右側の語の語長が長い連鎖を優先する（右最長一致法）
- 左側の語の語長が長い連鎖を優先する（左最長一致法）
- 文節の数が少ない連鎖を優先する（文節数最小法[3]）
- 語の隣接規則にコストを設け、その合計が最小な連鎖を優先する（コスト最小法[4]）
- 語や品詞の並びのパターンに対して、連鎖の優劣を明示的に指定する
- 文節の間の係り受けが成立する連鎖を優先する[5]
- 優先度の高い語や品詞を含む連鎖を優先する

実際にはこれらの方法が組み合わされて使用される。

以上の様な方法で、正解率が語単位で99%程度のシステムが構築されてきた。しかし、筆者の知る限りのシステムは、99%前後で精度が頭打ちになる。筆者は人手によるシステムの調整の困難さがこの原因であると考えている。

解析失敗の原因は未登録語によるものだけではない。実際に、解析に必要な語が全て辞書に登録されているにもかかわらず、正しい解析結果を出力できない文が多数存在する。これは上記の(2)の段階では正解が解の候補の中に含まれているにもかかわらず、その解が(3)の段階で最優先の解にならないためである。そこで、正解が最優先の解になるように、隣接コストや語の優先度などのパラメータを調整する。すると今度は今までは正解が得られていた別の文について正解が得られなくなる現象、いわゆるデグレードが起こる。

精度を上げるためには、パラメータの調整だけでなく、品詞の拡張や、語の分類体系の変更が必要な場合ある。しかし、システムがそうした変更を予定した形で作られていないと、これは非常に困難な作業になる。例え変更が容易だとしても、パラメータの調整や文法の拡張などの作業はシステムを熟知した者でないと不可能な作業である。その上、解の優先度の算出のアルゴリズムが複雑になると、システムを構築した者でさえ、その挙動を把握することは難しくなる。

そもそも、自然言語が対象のシステムでは、辞書の語彙が膨大になるため、こうした作業は、いずれ少数の人間の手に負えなくなる。未登録語の登録作業にしても、係り受けに関する情報や、語の分類に関する情報、優先度に関する情報などを

投入しようとすると、それらに関する詳しい知識を持っていないと、正確な情報を投入することはできない。こうして、精度向上のための調整ができる人材を育成することが困難な上に、例えそうした人材を育成できても、精度が向上しない事態になる。この主たる原因は、不適切な抽象化とシステムの硬直さにあると筆者は考える。

ここで言う抽象化とは、例えば、語に対して品詞や意味分類を与えることを言う。語の隣接規則や解の優劣を決める規則などが品詞を用いて記述されていたとすると、各語は品詞に抽象化されてから、それらの規則の適用を受けることになる。もちろん、個々の語の全ての組み合わせについての規則を記述することは事実上不可能であるから、適切な抽象化は必要である。しかし、用途に適した抽象化を行わないと、かえってシステムに悪い影響を与える。このような悪影響を避けて、抽象化を適切なレベルに保つためには、実例にあたるべきである。つまり、ある失敗事例を矯正するのに必要とされる場合に限って、抽象化を導入するのである。そのためには、システムの設計段階ではなく、運用段階で簡単に抽象化を導入できる仕組みが必要となる。

こうした人手による調整の困難を避けるために、事例を利用して解析を行う方法が提案された。これには、典型的な事例を用いる方法と、大量の事例から情報を抽出して用いる方法の二つが考えられる。典型的な事例を用いる方法では、典型的な例の選択が困難であるため、事例の選別を必要としない後者の方法を採用するシステム[6]が多い。この方法は、大量の解析事例に基づいて自動的に適切な抽象化とパラメータの調整を行おうとするものである。解析事例の収集自体は、例えば、文節の切れ目を入れ、読み仮名を振るだけといった程度であれば、特に訓練されていない人でも作業できるため、労力を惜しまなければ可能である。しかし、こうしたシステムは一般に事例そのものを用いて解析するのではなく、事例から語の隣接する確率等を計算し、これを用いて解析する。このように、一種の抽象化されたデータを用いて解析する場合には、ある特定の失敗例を矯正することが困難になる。これは、抽象化の段階が入るシステムでは、失敗例に対する正解を事例として投入しても、必ずしも解析に反映されるとは限らないからである。また、事例の投入がデグレードを引き起こす場合もある。さらに、この方法では、精度の要求が厳しくなるにつれて莫大な量の事例が必要になってくる。そして、事例の数が大きくなるにつれ、一つの事例の効果は相対的に小さくなるため、一定の効果をあげるために必要な事例収集のコストは容認できないほどに大きくなる。従って、筆者は、形態素解析システムの精度の向上のためには、大量の事例による裏付けと人手の効果的な介入とのバランスが重要であると考える。

以上考察したように、従来の形態素解析システムに精度向上の飽和現象が見られた原因は、人手の介入の難しさであると筆者は考える。複雑な抽象化、複雑な処理、システムの柔軟性の欠如は、システムの挙動に対する人間の理解を阻み、解析失敗例への対処を困難にする。システムの精度を向上させるためには、誰でも簡単にシステムの挙動を理解でき、システムの運用の段階で、的確にシステムを制御することを可能とする仕組みが必要である。

3. システムの概要

上記のような考察の下、筆者らは日本語形態素システムJTAGを開発した。システムの設計にあたって、次の点に留意した。

・単純であること
・具体的で理解しやすいこと
・柔軟であること

これらを実現するためにシステムに導入した特徴を以下に説明する。

3.1 システムの特徴

●属性は構造を持たないアトムに限定する

本システムにおいて、各語に属性が与えられるが、この属性は構造を持たないアトムのみとした。

語に属性を与えることは、抽象化の手段であり、語の集合に名前を付けることと同じである。本システムでは、システムが複雑化しないように、これ以上の抽象化の手段は与えない。つまり、属性の集合は取り扱わず、属性には、語の集合に付けられた名前以外の意味は持たせない。

●新しい属性を簡単に導入できる

属性は、実際には1以上の長さを持った単なる文字列である。従って、属性を記述する欄に新しい文字列を書き込めば、それで新しい属性が導入されたことになる。個々の属性の役割は、どういう文字列が使われたかには関係がない。しかし、システムを理解しやすくするために、属性を表わす文字列はその属性の役割を適切に表現したものであることが望ましい。

●各語に任意の数の属性を与えられる

本システムでは、個々の語に付加できる属性の数を制限しない。これは、どのように抽象化すべきかについて前もって分からない以上、語に与える属性の数を制限することは望ましくないからである。

●品詞属性とグループ属性の2種類の属性を用いる

品詞属性は隣接規則に用い、グループ属性は各語との共起の指定に用いる。この二つは、本質的には分ける必要はないが、処理の効率化のために分割した。

●品詞属性を用いて隣接規則を記述する

本システムでは品詞属性の隣接規則によって文法が表現される。本システムで用いられる日本語の文法は、派生文法[7]に基づき、筆者が文献[8]で提案した文法である。

本システムでは、隣接可能な品詞属性のパターンが、あるファイルに記述されている。システムはこのファイルを参照して、各語の隣接可能性を判断する。この隣接規則中には、個々の語を記述できるない。しかし、必要なら、特定の語のみが持つ品詞属性を作り、同様の効果を得ることができる。リスト1に隣接規則の例を示す。

リスト1　隣接規則の記述例

名詞,	格接尾辞:連用,	50	..(1)
名詞:固有,	名詞接尾辞:名詞,	100	..(2)
名詞:-固有,	名詞接尾辞:名詞,	90	..(3)
判定詞:で,	動詞語幹:Lで,	50	..(4)

隣接規則は一行一規則で、カンマで区切られ、最初の項が左側の語の品詞属性群、2番目の項が右側の語の品詞属性群、3番目の項がその規則の持つコストの値である。品詞属性群はコロンで区切られた品詞属性で構成される。品詞属性の前に付いたマイナス記号は否定を表す。例えば、(1)は、「名詞」の品詞属性を持つ語の後に、「格接尾辞」と「連用」の品詞属性[1]を持つ語が続くことができ、そのコストが50であることを表している。また、(2)の規則は、「名詞」と「固有」の品詞属性を持つ語の後に、「名詞接尾辞」と「名詞」の品詞属性を持つ語が続くことが可能で、そのコストが100であることを示し、(3)の規則は、「名詞」の品詞属性を持ち、「固有」の品詞属性を持たない語の後に、「名詞接尾辞」と「名詞」の品詞属性を持つ語が続くことが可能で、そのコストが90であることを示している。(2)と(3)によって、「名詞接尾辞:名詞」の品詞属性を持つ語は固有名詞よりも固有名詞以外の名詞の後に来やすいことが表現されている。(4)の規則では、判定詞の「で」のみが「判定詞:で」という品詞属性の組み合わせを持つため、特定の語を指定する規則となっている。

このように、本システムでも隣接規則で抽象化を用いている。事実、この隣接規則を変更するとデグレードが発生する危険性が高い。そこで、隣接規則は主に可能な語の連鎖を求めることに用い、その中から正解を選ぶのは他の手段に負っている。

●語の共起を用いて正解を選ぶ

本システムでは語の共起の有無に基づいて解

[1] 左端の属性は主属性として特別扱いである。その他の属性の順番は意味を持たない。この場合は「格接尾辞」が主属性である。

を選択する。共起の指定の仕方を説明するため、本システムで用いる辞書のレコードの記述例をリスト2に示す。

リスト2　辞書レコード記述例

額,名詞,	G量, 90,G金銭,	税金:報酬,	ガ'ク
額,名詞,	G枠, 96,外す:G絵,	絵,	ガク'
額,名詞,	G体, 95,広い:狭い,	汗,	ヒタイ
外,動詞語幹:S,	外す, 90, G枠,,	ハメ,	ハズ

辞書のレコードは一行一規則[2]で、各項はコンマで区切られている。最初の項が語の表記、2番目が品詞属性、3番目がグループ属性[3]、4番目が語のコスト、5番目がグループ属性との共起、6番目が他の語との共起[4]、7番目が読みのデータである。それぞれの属性および共起はコロンで区切ることで複数指定が可能である。「額」は上記に示したコストを持つため、共起が認識されなければ、「ガ'ク」が選択される。しかし、共起関係が同一文中で認識されると、それぞれの語が選択される。これによって、「額を外した」「額に汗して働く」等の読み分けが可能になる。

このように、本システムでは、共起に関して非常に単純な扱いをしている。当然、共起に関してさらに細かい適用条件を付けることは考えられるが、そうした複雑さを導入するとシステムの挙動が把握し難くなる上に、あまり効果が期待できない。従って、本システムでは単純なものに留めた[5]。もちろん、この単純な扱いを欺く文を考えることは容易であるから、これだけで精度が100%になることはありえない。しかし、実用上十分な精度に達することは可能であると考えている。

語の共起をこのように用いることは本システムの中で最も重要な部分である。なぜなら、これによってシステムの調整の容易さが生み出されるからである。ある失敗事例を矯正するために隣接コストを変更するとデグレードを引き起こす可能性が高い。また、意味分類など広範囲に抽象化されたものに対して共起を設定するのも同様な危険がある。しかし、個々の具体的な語の共起に関してなら、人が十分に把握することが可能である。影響が広範囲に及ばないという点でも、デグレードの可能性は低い。

3.2 解の選択方法

本システムでも隣接規則を用いて可能な語の連鎖を求めることは、上記した従来の方法と同様である。以下では、本システムにおいて、複数の解の候補の中から解を選択する方法について述べる。

●隣接コストの利用

コスト最小法に見られるように、隣接コストを用いることは明らかに効果がある。本システムでもデフォルトの方法としてコスト最小法を用いる。また、処理の効率を上げるために最初に隣接コストよって枝刈りを行う。ただし、この時点で正解が刈られないように十分に幅を持たせる。

●共起の利用

同一文中に共起する語が含まれる連鎖の優先度を上げる。複数の共起が認定された場合、その数に比例して優先度を上げる。

●語の優先度の利用

語の共起と隣接コストを用いても、複数の解が同一の優先度を持つ場合は、語に付された優先度を用いる。これは、典型的には、同形異音語のデフォルトの読みを指定するのに用いられる。

---

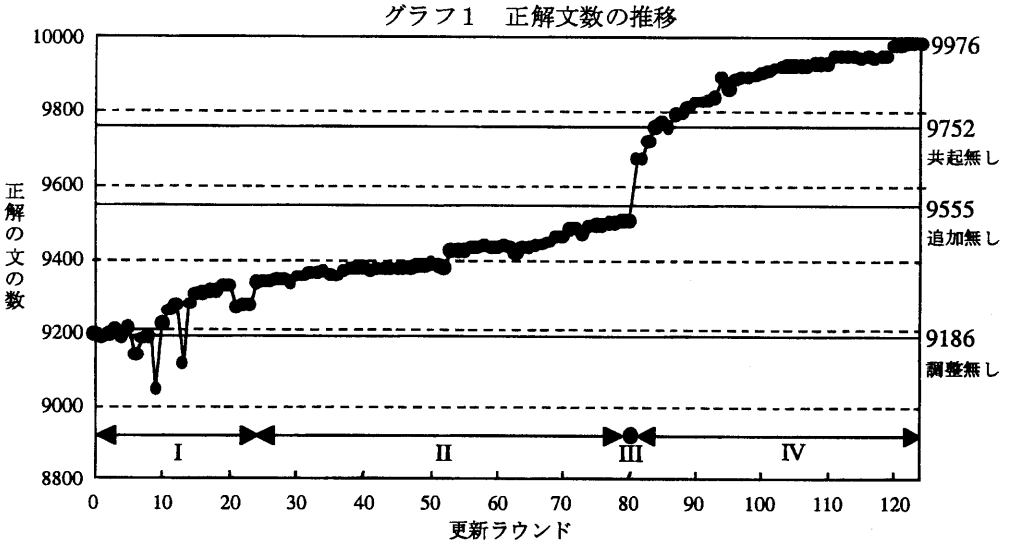
[2] UNIX等の処理系で処理しやすくするため。

[3] 動詞語幹と形容詞語幹の場合、グループ属性の欄に基本型を記述している。これはシステムの仕様ではなく、運用上の便法である。グループ属性の頭にGを付けているのも同様で、付ける必要性はないが、理解しやすさのための便法である。

[4] 語と語の共起は字面だけで指定する。

[5] 実際には、左右方向の指定、共起する語間の文節数の制限を記述することはできる。

グラフ１　正解文数の推移



●語長の利用

辞書の中から同形異音語を抽出するのはたやすいので、それらの間に優先度を付けるのは容易である。しかし、長い複合語を解析する場合に、思わぬ分け方が同一の優先度を持つことがある。こうした場合を前もって把握することは難しい。そこでそうした場合には、語長が２文字の語を優先させる。

●解の固定

最後まで優先順位が決まらない場合には、語の表記の文字コードが小さいものを優先させると決めておく。いずれ失敗事例の中に現れて、適切な共起が設定されることが期待される[6]。

4. 評価

本稿でのシステムの評価は以下の基準で行った。

解析によって一文を仮名表記に変換して一つの文字列にまとめ、それが正解と等しい文を数える。

これは従来の評価方法とは異なるが、評価が容易で安定していることから採用した。従来の様に個々の形態素毎に正解を数える場合、形態素の切り分け方に揺れが生じる。実際、本システムとＥＤＲコーパスでは使用する品詞体系が異なり、接頭語、接尾語、用言の語尾および複合名詞の切り分けなどのポリシーが異なるため、単純に比較することができなかった。上記の評価法では切り分けのポリシーの違いによる影響を受けない。ただし、切り分けの正誤が評価されない。概ね従来の形態素単位の正解率を、１文に含まれる形態素の数だけ冪乗した値が文単位の正解率になるが、切り分けの誤りが評価されないため、全く同じではない。

4.1 正解率の推移

本システムの解析結果の正解率を調整作業毎にプロットしたグラフをグラフ１に示す。解析の対象文はＥＤＲコーパス[9]から無作為に選んだ数字を含まない１万文[7]で、原文と読み仮名を校正し、読み仮名の揺れを統一した文[8]である。対

[6] ここで解が固定された場合の記録を取れば、より積極的な対応が取れる。

[7] 数字を含む文を除いたのは、ＥＤＲコーパスで数字を含む文の仮名表記の揺れが非常に多かったためである。

[8] 原文または読み仮名の誤りが2.3%、読みの揺れが1.5%の文に含まれていた。

象文の文字数の1文当たりの平均値は約31文字で、形態素数の平均値は約21個であった。システムの調整は、未登録語の登録、登録語の削除、登録語のコストの変更、隣接規則の追加／削除／コスト変更、共起データの追加等によって行った。このように様々な調整が混在したため、更新ラウンド毎の条件を揃えることができなかった。

以下に、グラフ中にIからIVで示した範囲における調整の特徴を述べる。グラフ中のIの範囲では、隣接規則の変更がしばしば行われたため、正解数が大きく減少する場面が見られる。その後、隣接規則の変更に迫られることが少なくなったため、大きく減少する場面はなくなったが、小さい減少は残っている。これは、登録語のコストの変更等が悪影響を及ぼすためで、デグレードが完全になくなっていないことを示している。IIの範囲では、概ね50箇所程度の調整を行う毎に記録を取っていたため、正解数が単調に増加している。その後、システムが自動抽出した未登録語を人手で手直しし、IIIの時点で一気に登録した。従って、IVの範囲では共起データの追加が主となった。本システムの仕様では正解数を向上させられなくなった時点で作業は終了した[9]。

グラフからオープンテストでは91.9%[10]の正解率[11]であることが分る。また、クローズドテストの結果は99.8%[12]であり、本システムでの精度の限界値を示している。調整が終了した後、共起を用いる機能を無効にして測定した結果をグラフ中に「共起無し」として示した。これが97.5%であることから、共起による処理が有効な文が2.3%であることが分る。また、調整の後、調整の過程で登録した未登録語を全て外して測定した結果をグラフ中に「追加無し」として示した。これが95.6%であることから、未登録語による解析誤りが4.2%であったことが分る。

表1　原因別の誤りの割合

	全文に対する割合	誤りに対する割合
未登録語	4.2%	52%
共起	2.3%	28%
その他	1.6%	20%
合計	8.1%	100%

表1に原因別の解析誤りの割合を示す。表中の「その他」には、登録語のコストの誤り、登録語の読みの誤り、登録語の属性の誤り、隣接規則の誤り、誤った語の登録による誤りが含まれる[13]。表によると、仮名振りの誤りの半分強が未登録語によるものであるが、共起データの不足によるものも3割弱を占めている。従って、正解率を上げるためには、未登録語を減らすと共に、共起データを充実させることも必要である。

4.2 システムの比較

一般に公開されている日本語形態素解析システムJUMAN[10]と本システムJTAGとを本稿の評価法で比較した。対象文は未校正のEDRコーパスで数字を含まない8万文(平均38文字)である。これには、上記の調整に用いた1万文を含まない。システムの性能は用いる辞書の質や規模によって左右されるため、それぞれのシステムに付属する辞書に含まれる自立語を相互に入れ換えるなど条件を変えた場合の正解率も算出した[14]。付属語については各々の使用している文法と密接に結びついているため、各々のシステムのものを使用した。また、公平を期すため、JTAGと同じ当て読み[15]の機能をJUMANに追加して測定した。

---

[9] 最後まで残ったのは「方（かた／ほう）」と「後（のち／あと）」であった。

[10] 形態素単位に換算すると99.60%。

[11] JUMANの正解率は70.6%であった。

[12] 形態素単位に換算すると99.99%。

[13] 都合上、個別の割合を記録できなかった。

[14] 品詞の分類体系が異なるため、互いに正確な変換ができない部分もあった。

[15] 「当て読み」とは未登録語の漢字の部分を音読みで置き換えることである。

表2　JTAGとJUMANの比較

自立語 / システム	JTAG	JUMAN	JTAG+JUMAN
JTAG	88.3%	70.4%	86.7%
JUMAN	62.0%	69.6%	68.0%

結果を表2に示す。本来の辞書を用いた場合、JTAGは88.3%[16]、JUMANは69.6%[17]の正解率であり、JTAGの方が18.7%優っている。JUMANでJTAGの自立語を用いると、JUMANでJUMANの自立語を用いた場合より正解率が低くなる。また、JTAGでJUMANの自立語を用いた場合は、JUMANでJUMANの自立語を用いた場合より0.8%ほど正解率が高い。これは機能的にJTAGがJUMANを包含し、かつ、共起の利用が効果を上げたためと考えられる。また、両方の自立語を共に用いた場合、どちらのシステムでも正解率が減少した。このことは、闇雲に辞書を大きくしてもデグレードが起きて精度が上がらないことを示している[18]。

5. おわりに

本稿では、精度向上のためのシステムの調整が容易であることに重点を置いた日本語形態素解析システムについて述べた。本システムで、調整の際のデグレードの危険性が小さくなった結果、投入した労力に見合った精度向上が期待できるようになった。現時点での仮名振りの精度は文単位で91.9%であり、さらに、99.8%まで向上する可能性がある。精度向上のためには、未登録語を減らし、必要な共起データを集める必要がある。しかし、デグレードを避けつつ、語彙を増やすためには、解析誤り例に立脚してデータを集めなければならない。そこで我々は形態素解析システムJTAGのユーザを増やし、ユーザからのフィードバックを集約する環境の構築を計画中である。詳しくは以下のURLを参照願いたい。

http://info.isl.ntt.co.jp/chisho/jtag

[16] この値が先の値よりも劣るのは、正解として用いたコーパスを校正していないためと、対象文の平均の文字数が異なるためである。

[17] 当て読みの機能を外した場合は67.9%。

[18] JUMANの辞書には漢字を展開した語が含まれるため、例えば「白衣(びゃくい)」を展開した「白い(びゃくい)」などの名詞があり、形容詞の「白い(しろい)」と衝突してデグレードを起こしている。しかし、展開した語彙を全て外すとJUMANの正解率は60.5%まで落ちてしまう。

参考文献

[1] 長尾真 他, 自然言語処理技術のこれからの課題, 「自然言語処理の技術動向」調査報告会, 1994.

[2] 人工知能学会編, 人工知能ハンドブック, オーム社, pp.226-227, 1990.

[3] 吉村賢治 他, 文節数最小法を用いたべた書き日本語文の形態素解析, 情報処理学会論文誌, Vol.24, No.1, pp.40-46, 1983.

[4] 久光徹 他, 接続コスト最小法による形態素解析の提案と計算量の評価について, 電子情報通信学会技術研究報告, NLC90-8, pp.17-24, 1990.

[5] 宮崎正弘 他, 日本語文音声出力のための言語処理方式, 情報処理学会論文誌, Vol.27, No.11, pp.1053-1060, 1986.

[6] 永田昌明, 前向きDP後向きA*アルゴリズムを用いた確率的日本語形態素解析システム, 情報処理学会研究報告, 94-NL-101-10, pp.73-80, 1994.

[7] 清瀬義三郎則府, 日本語文法新論-派生文法序説-, 桜楓社, 1989.

[8] 渕武志 他, 日本語形態素解析システムのための形態素文法, 自然言語処理, Vol.2, No.4, pp.37-65, 1995.

[9] 日本電子化辞書研究所, EDR電子化辞書使用説明書, 1993.

[10] 松本裕治 他, 日本語形態素解析システムJUMAN使用説明書 version 2.0, TR94025, 奈良先端科学大学院大学, 1994.