2

形態素解析システム「茶筌」

松本 裕治
奈良先端科学技術大学院大学 情報科学研究科
matsu@is.aist-nara.ac.jp

形態素解析：文を形作る単語を識別する

日本語にかかわる処理を計算機を使って行う際に，単語の分かち書きは最も基本的な処理である．英語などの欧米言語では単語が空白によって区切られているため，単語の認識は大きな問題にならないが，日本語や中国語のように単語の間に区切り記号を持たない言語は，単語の認識が根本的な問題になる．単語の識別（tokenization），活用語処理（lemmatization, stemming），品詞の同定（part-of-speech tagging）は，個別の処理とみなされることもあるが，日本語では，これらの処理が密接に関連するため，同時に行うのが普通で，全体的な処理を形態素解析と呼んでいる．図-1に，現在の茶筌の出力例を示す．1つの単語が1行に出力され，その読み，品詞，活用の型や活用形が表示されている．

日本語の形態素解析では，活用語の語尾変化の規則性は書き下すことがさほど困難ではないが，それ以外の処理のための規則をすべて書き下すことは容易ではなく，分かち書きについては，従来から，最長一致法（辞書内の単語となるべく長く一致する語を優先する），文節数最小法（文全体の文節数が最小になるような単語の列を優先する）などの発見的（heuristic）規則が用いられることが多かった．

ただし，文として考えると，自然な品詞のつながりがある一方，普通の文ではとても現れないような品詞のつながりもある．そこで，品詞（活用形等を含む）の間の連接可能性を記述した表を用意したり，連接のしやすさを数値化し，その大小によって単語間のつながりの自然さを順序づけたコスト最小化という考え方を取り入れるなどの規則の詳細化が行われた．

我々のシステムでも当初からコスト最小化という考え方を取り込み，単語のコストや品詞間の連接コストを人手で記述し，連接表を作る方法をとっていた．規則を詳細化していけば確かに解析精度が上がるが，一般に多数の規則を持つシステムが規則の増加とともに規則間の整合性などのメンテナンスが困難になるのと同じ問題を経験することになった．

本稿では，形態素解析システム「茶筌」について，それが何であるかを中心に書くのではなく，我々が現在のシステムを作るに至った経緯を説明し，それによって，なぜ「茶筌」が今のようなシステムになったかを理解していただこうと思う．

茶筌が生まれるまで

著者が日本語の形態素解析システムを本格的に作ろうと思ったのは，1989年の秋に京都大学の長尾真先生の研究室に助教授として赴任したころである．このころ，研究対象の言語として日本語をメインにしようと思い始めていたが，当時日本で最も自然言語処理研究が進んでいた長尾研究室ですら，十分な日本語処理環境があるわけではなかった．すでに機械翻訳などの研究のためにシステムを開発していたいくつかのメーカから形態素解析システムの使用許諾を得ようと試みたが，さすがに辞書の内部まで公開してくれるところはなかった．

研究利用するためには少なくとも辞書や解析のための規則を自由にいじることができる必要があり，長尾先生の勧めもあって，少なくとも自分たちのグループ内で共有できる日本語形態素解析システムを作ってみようと考えるようになった．1990年の春である．

```
茶筌は、日本語の形態素解析を行う。
茶筌	チャセン	茶筌	名詞-一般
は	ハ	は	助詞-係助詞
、	、	、	記号-読点
日本語	ニホンゴ	日本語	名詞-一般
の	ノ	の	助詞-連体化
形態素	ケイタイソ	形態素	名詞-一般
解析	カイセキ	解析	名詞-サ変接続
を	ヲ	を	助詞-格助詞-一般
行う	オコナウ	行う	動詞-自立	五段・ワ行促音便	基本形
。	。	。	記号-句点
EOS
```

図-1　茶筌の実行例

著者は，それに先んじること5年，通産省主導の国家プロジェクトICOT（新世代コンピュータ技術開発機構）に所属していた折に，論理プログラミング言語を用いて動的計画法をバックトラックなしに並行探索する手法を考えており[松本 91]，これを使って構文解析システムSAXというものを作っていた．当時，同じ研究所にいた杉村領一氏（現 松下電器）と一緒に，形態素解析も同様の手法でできるということで，日本語の形態素解析の全解探索プログラムをインプリメントし，これをLAXと名づけていた．これは今から考えると，動的計画法の実現法であるViterbiアルゴリズムを論理型言語にコンパイルする手法を実現していたことになる．

長尾研究室で日本語形態素解析システムを開発しようと決心した際，本体のプログラムは上記の手法を使えば百数十行程度のPrologプログラムで記述できることが分かっていたので，辞書の構成を中心に考えることにした．協力を申し出てくれたのがちょうど卒業研究を始めた妙木裕君（現 キヤノン）である．まず，辞書の項目はもちろんのこと，品詞のセットの定義や活用のタイプや具体的な活用形など，辞書に関するすべての項目を利用者が自由に定義し記述できることを基本方針とした．当時そして今でも，品詞の種類や名称，活用型の類別など，すべての研究者で共有できるものはなかったからであり，特定の品詞分類に限定するのは得策ではないと考えた．

しかし，具体的には何かの品詞分類を採用する必要があり，益岡氏と田窪氏による「基礎日本語文法」（くろしお出版，1989）を叩き台として採用することにし，品詞や活用形や分類で不完全と思える部分を埋めていく作業から始めた．妙木君を中心とし，当時長尾研究室に在籍していた多くの学生諸君が週に1，2度著者の部屋に集まっては熱心に議論してくれた．文法に関する詳細が決定されてきてから最も頭を悩ませられたのは，辞書の中身をどこから集めるかということであった．幸いかな漢字変換システムWnnの辞書が完全なフリーソフトであることを知り，その辞書を元に3万数千語程度の辞書を構築することができた．辞書については，その後，ICOTから岩波国語辞典の全見出しがフリーデータとして公開され，これを取り込んで，12万語程度の辞書に拡張した．辞書の記述については，研究室にCで書かれたリスト処理のルーチンがあったため，将来の拡張を考えてLISPのS式の形式で記述することにした．最初の辞書のサイズは2Mバイト程度であったと思うが，当時，主記憶が4Mとか8Mが普通の時代には2Mバイトは巨大なファイルであり，ハッシュ表では無謀だから辞書はディスクにおくことを前提にしてB木で実現することにした．妙木君の卒業論文は利用者がカスタマイズ可能な辞書システムが中心で，本システムの稼働は残った者が引き継ぐこととなった．本体のPrologプログラムは数日で書き上げたが，連接表を探そうにも，どこにも実用的なものがなく，これも自分たちで作るしかなかった．

本体がPrologだけでは使いにくいということで，本体をCでも組むことにした．当時博士課程の黒橋禎夫君（現 京大講師）が本体の移植を担当し，利用者ごとに設定ファイルを記述できるなどの拡張が行われた．学生諸君とシステムの名称について話し合い，日本語（Japanese）の，利用者によるカスタマイズが可能な（User-customizable）形態素解析（Morphological Analyzer）ということで，JUMANと名づけることになった．Nは長尾研の頭文字ということにした．

JUMANを高精度に動かすためには，品詞の連接コスト表が必要であり，これを自分たちで作るしかなかったので，研究室で持っていたいくつかの電子化テキストをテストデータとして解析し，解析誤りの原因を考えては連接コスト表や辞書内の単語のコストを修正するという作業を黒橋氏と毎日繰り返し，夏休みの半分くらいをつぶしてしまった．1991年の夏であったと思う．解析結果をプリントアウトして家へ持ち帰り，エラーの原因とそれへの対応策を考える

ことを日課とした．毎日，2人で相談しつつ連接コスト表に反映させながらシステムを動かしてその影響を確認し，その日の結果をまた持ち帰るということを繰り返した．かなりしんどい作業だったが，日に日に精度が上がるシステムを見るのはなかなか楽しかった．

当初は研究室内での利用を考えていたが，日本語の形態素解析システムを探す声が外部からもしきりに聞こえ，これをフリーソフトとして外部公開することを決心した．1992年の冬にJUMAN0.6として公開した際の最初の利用者は米国のMCCのグループだった．その後，MCCのグループは辞書のハッシュ化を行った上でその結果を返してもらえた．フリーソフトにすることの恩恵を身をもって体験した．

1993年の春に著者が，新設の奈良先端大に異動し，JUMANの開発は京大と2カ所で分かれて継続することになった．黒橋君たちがUNIXのデータベースシステムNDBMを利用して擬似的なTRIE構造辞書を構築し，辞書検索の高速化を行ってくれたが，これはその後，辞書が大規模になるにつれて速度効率の効果が薄れることが分かった．また，辞書のコンパイルに時間がかかること，および，他プラットフォームへの移植が困難であるとの理由から，より身軽な辞書の構築が必要と考えるようになった．著者は，奈良先端大に移ってから，辞書と本体の高速化と連接規則の自動学習を行いたいと常々考えていたが，1996年の春に文字列検索のためにパトリシア木の構築を米沢恵司君（現 野村総研）と山下達雄君（現 富士通研究所）が中心になって進めてくれ，これを辞書システムとして使うことにした．また，北内啓君（現 NTTデータ）が本体の改造を申し出てくれ，山下君，今一修君（現 日立製作所），今村友明君（現 富士通）たちと一緒に夏休みのプロジェクトとして，JUMANの本体と辞書システムを徹底的に作り直すことになった．その年の夏休み明けまでに改良が進み，辞書のコンパイル時間が約1/5になり，解析速度が10倍以上も高速化された．この時点で京大のJUMANとは内容がかなりずれてきたため，思いきって新しい名前を付けて別のシステムとして公開することにした．大学のある奈良県高山町が全国有数の茶筌の産地であるという単純な理由で「茶筌」という開発コード名が付けられ，これが正式名称として定着した．

現在の茶筌へ

茶筌を開発する前に，JUMANへの機能追加としてもう1つ考えていたのは，連接表を自動学習することであった．コスト最小化の考え方は，確率最大化という考えとほとんど等価と考えられるので，音声認識で定着していた隠れマルコフモデル（HMM）の考え方を取り入れるのは自然なことで，英語の品詞付与問題では標準的な手法として取り扱われていた．1994年ごろから竹内孔一君（現 国立情報学研究所）がHMMの適用について一連の研究を行ってくれ，小規模な品詞タグ付きデータから初期確率をとれば，大量の未解析のデータについてHMMを適用することにより，ある程度の性能を達成できることが分かった．しかし，扱っていた品詞のセットは巨大であり，未知の現象についてはやはり正しい事例を蓄積する必要性を認識した．また，連接コスト表は，基本的に連続する2つの品詞（または単語）の関係であり，HMMでいえば，bi-gramのモデルである．すべての現象について，2つを超える品詞列を見るにはきわめて大量の学習データを要するし，また精度向上のために必須でもない．しかし，一部の現象については，2つの品詞連続を見ていたのだけでは正しい解析が行えない．1995年ごろに可変長のマルコフモデルという考え方があるのを知ったが，これはbi-gram, tri-gramさらにそれ以上の長さの文脈を併用したモデルであり，結局は有限状態機械として実現できるので，連接表を状態遷移と考えることで無理なく実現できることが分かった．上記の夏休みの茶筌プロジェクトの後で，北内君が可変長の連接規則を状態遷移表にコンパイルするプログラムを完成してくれて[北内 97]，茶筌は実行効率を落とすことなく可変長の連接規則を扱うことができるようになった．同じころ，春野雅彦君（現 ATR）が文脈木という考え方を利用してモデルの可変長部と使用する品詞の詳細さを同時に自動学習する方法を考案してくれ，boostingという手法を使ってさらに精度向上が実現可能であることを示してくれた[Haruno 98]．boostingは，訓練データに対する学習モデルの誤りに注目して，それまでに学習したモデルの弱みを補うモデルの学習を繰り返して，最後にすべての学習モデルを混合するという方法であるが，この際に，可変長モデルのように柔軟性の高い学習モデルを用いることが必須であることが分かった．boostingにより精度向上が期待できることが分かったものの，複数のシステムを同時に動かしてその組合せ

によって解を決定するのは，速度の上から問題があり，結局茶筌では採用しなかった．

一方，精度の向上のためには大規模な解析済みコーパスが必要であるが，当時はそのようなものが存在しなかった．幸い，1996年の暮れにRWCPタグ付きコーパス（URL:http://www.rwcp.or.jp/wswg/rwcdb/text/）が公開され，その中の人手修正が施された3,000記事を学習データとして使うことにした．しかし，人手による品詞付与の揺れは深刻であり，満足のいく精度の規則はとても学習できないことが分かった．それから1年以上をかけてコーパスのタグ付けエラーの発見の手法と修正ツールを開発してコーパスの誤りを修正し，ある程度満足のいく精度の辞書（IPADIC 0.8）を公開したのが1998年の春だった．このときに，従来から用いていた益岡・田窪の品詞体系から決別し，RWCPコーパスで用いられていたIPA品詞体系を標準とすることになった．

学習による精度の向上のためには，品詞だけを対象にしていたのでは限界があり，ある種の単語については，品詞ではなく，その単語そのものを1つの品詞として統計をとる必要がある．また，可変長の文脈についても，自動的に推定したものをすべて採用するのではなく，真に必要なものだけを使う方が，記憶量，速度，学習データの有効利用の点から考えて，より好ましい．北内君や春野君はこれらの自動推定に関する研究を行ってくれていたが，最終的には人間の言語的直感を記述できる機能や，真に意味のある規則を人間が選択することが必要だと考えていた．また，上記の詳細化以外に，異なる品詞や単語を，文脈に応じて同一視（グループ化）する機能も必要であると考えた．たとえば，「〜しちゃう」という表現の「ちゃう」は，ほかのどの単語とも異なる振る舞いをするので，従来の考え方では，これを1単語1品詞と考え，さまざまな使用例を学習データとして蓄積する必要があった．これでは，単語ごとに規則を記述することと大差なく，学習の恩恵を十分得ることができない．よく考えると，「ちゃう」という単語は，本来「て／しまう」という2つの単語が1つに縮退したものである．同じような単語として，「とく（て／おく）」「とる（て／おる）」「てる（て／いる）」などがある．「ちゃう」が上のような単語だと分かると，この単語は左から見れば「て」と同じ（グループの語）であり，右から見ると「しまう」と同じ（グループの語）と機械に教えてやればよい．このように連接の役割ごとのグループ化（position-wise grouping），および，上記のような単語化や可変長文脈の指定を人間が教えてやり，それに基づいて統計学習を行うことにより連接規則表を生成する方法を浅原正幸君（現 博士前期課程）が考案し，実現してくれた[Asahara 00]．1999年の暮れに公開した茶筌2.0に添付の辞書（IPADIC 2.0）はこの学習プログラムによるものである．

茶筌の本体については，高速化や可変長規則の取り扱いなどここで述べた以外に，さまざまな改良や機能追加が北内君を中心に進められた．詳しくは茶筌のマニュアルを参照してほしいが，主なものを列挙すると次のとおりである．

- 解析結果の表示を「出力フォーマット」の記述により自由に指定できる．
- SGMLやXMLなどのタグのついた文書の本文部分のみを解析できるように，文書中のタグ（注釈）部分の開始記号や終了記号を指定することができる．
- サーバ化が行われ，クライアントを用いて他のマシンから解析を行うことができる．

現在の茶筌（2.0）の仕様は，辞書項目は約23万語，解析速度はPentiumIII（450MHz）上のLinuxで約40,000文字／秒（100Mバイトの新聞記事1年分の全文を約20分で解析可能）である．

茶筌の辞書の構成について，簡単にまとめておく（図-2）．辞書項目が格納された「学習前辞書」と「品詞タグ付きコーパス」に基づいて学習プログラムが種々のコストの学習を行う．この際，文法エラー等の確認や活用形から原型に戻す操作を行うために「文法定義ファイル」が参照される．学習の結果得られるのは「連接規則辞書」すなわち状態遷移に伴うコストの一覧，および，「学習後辞書」である．「学習後辞書」の項目は学習前のものと基本的に同じであるが，単語の持つコストと「読み」が複数ある場合の優先順序がコーパスにより学習される．タグ付きコーパスには現れるが辞書にはない語が存在する場合があり得るが，そのような語は（図には示されていないが）例外辞書として別のファイルに出力され，それを辞書に含めるべきかどうかの判断は利用者に委ねられる．学習によって得られた「連接規則辞書」と「学習後辞書」は，それぞれ状態遷移表とパトリシア木に変換され，これらが茶筌本体に参照されて利用される．

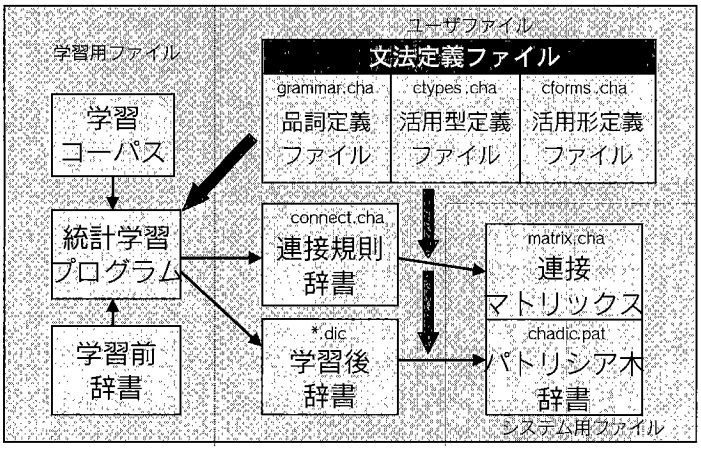

図-2　茶筌の辞書の構成

茶筌の周辺

茶筌を開発する上で忘れてはならないいくつかの事項をここで記しておきたい．1つは，利用者からの声である．茶筌だけではなく自然言語処理ツール全般のためのメーリングリスト（nlt@cl.aist-nara.ac.jp）というものを立ち上げており，現在400名弱の登録者がいるが，このメーリングリスト，あるいは，直接著者や茶筌のためのメーリングリスト（chasen@cl.aist-nara.ac.jp）に対して多くの要望や質問が寄せられた．これらの要望あるいは不満の声は，茶筌と辞書を開発していく上で大変に役に立つものであった．茶筌は最初から完全なフリーソフトを目指していたので，研究利用はもとより商用利用にも制限はなく，現在は我々も把握しきれないほどの利用をいただいているようである．

利用者からの貢献のうち特筆しておきたい事項として，IPA音声ディクテーショングループ（代表：鹿野清宏（奈良先端大））による辞書の改良を挙げておきたい．これは，日本語ディクテーションシステムを開発しフリーソフトとして公開するプロジェクトで，IPAからの支援を得たものだが，その言語処理部として茶筌を採用していただき，固有名詞（特に人名，地名）などの大量の辞書項目の追加，および，いわゆるふりがなとしての「読み」以外に「発音」の項目を追加していただいた．同グループの山田篤氏（ASTEM）と伊藤克亘氏（電総研）には大変な骨折りをいただいた（1999年春のIPADIC 1.0の公開時には「読み」の項目が削除され，発音に相当するもので置き換えられたが，1999年暮れのIPADIC 2.0では，「読み」と「発音」の両方が記述されるようになった）．

開発の過程において，解析誤りの原因の発見や学習のための正しいデータを蓄積することがきわめて重要なことだった．山下君が開発してくれた「美茶」というシステムは，このような要望に応えてくれるもので，解析結果を曖昧性を含めてグラフの形で表示してくれ，連接や単語のコストを表示してくれる．これにより解析誤りがある場合に，何が原因であるかを簡単に特定することができる．また，マウスによって正しい解析結果を選択し，それを格納することにより，正しい解析結果の蓄積を行うことができた．このツールは辞書の改良に大いに貢献してくれた．なお，このシステムはTcl/Tkで書かれており，UNIX以外の環境への移植が難しかったことなどがあり，プラットフォームやその他の拡張性を重視して同様の機能をJavaで実装した新たなシステムとして，松田寛君（現 ジャストシステム）がVisualMorphsを開発してくれた（図-3）．これらのシステムはいずれも茶筌のホームページでフリーソフトとして公開している．

開発の当初からWindowsへの実装を望む声が多く寄せられていた．UNIX環境にどっぷり浸かっていた我々には容易なことではなかったが，平野善隆君（現 ギフトケン・ドット・コム株式会社）が茶筌1.0をWindowsに移植してくれた（WinCha 1.0）．また，1999年暮れの茶筌2.0の公開時には，松田君がWindowsへの移植を行ってくれた（WinCha 2.0）．その後，Windows版とUNIX版のソースの共通化が進められ，茶筌2.0の改良版として現在公開されようとしているシステムでは本体部分の区別はなくなった．

山下達雄君は，形態素解析システムの改良や新たな

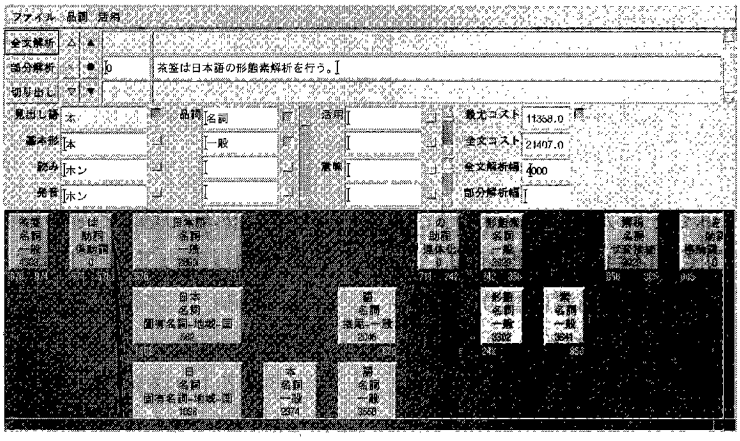

図-3　VisualMorphsの動作画面

機能の追加に際して，その容易さを図ること，および，個々の機能追加がどのような効果を持っているかを明らかにするために，モジュール化を重視したLimaTKというシステムを作っていた[Yamashita 00]．特に，辞書引きのタイミングをいくつかに分類して整理することにより，わずかな言語的特徴を記述することによって性質の異なる言語の解析を統一的に行えることを示した．茶筌に彼の考えを取り入れ，現在では，茶筌は，日本語だけでなく英語や中国語などの形態素解析に対応できるシステムになっている．

これからの茶筌

現在作成中のシステムと辞書は，本稿の出版時までには（おそらく，茶筌2.1およびIPADIC2.4の名称で）公開されているはずである．学習データ（約95万語）からバグを根絶するのが困難であるため解析精度を正確に議論するのは難しいが，学習データの一部をテスト用として分離して繰り返し評価すること（cross-validation）により，詳細な品詞レベルまで見ると，精度97.7%，再現率97.5%であり，分かち書きだけを見ると，精度99.1%，再現率98.9%である☆1．学習データの不足が原因の誤りの場合は，正しい解析結果を学習データに追加することにより，簡単に誤りに対応することができる．

今後の茶筌について，著者が考えている主な点についてまとめておきたい．前章の最後で述べたように，茶筌のシステムと辞書はすでに多言語対応している．いくつかの改善の余地はあるが，現在内部的には，英語，中国語（BIG5コード），ドイツ語のシステムが稼働している．韓国語については，以前，平野君が実装したが，辞書の規模を大きくすることができず，その後，システムの進展に追従していないため，現在は稼働していない．本体は共通なので，いずれの言語についても辞書の著作権をクリアすることが当面の問題である．英語については，Oxford Text Archive（URL: http://ota.ahds.ac.uk/）のOALD（Oxford Advanced Learner's Dictionary of Contemporary English）の見出し語が利用可能であり，またLDC（Linguistic Data Consortium, URL:http://www.ldc.upenn.edu/）で扱っているPenn TreeBankのタグ付きデータからの学習（辞書項目と連接コスト）の許可が得られたので，いつでも公開可能である．単語化，可変長文脈の選定，品詞や単語のグループ化などの設定が進み，学習が完了すれば公開したいと考えている．なお，茶筌の標準辞書で用いているパトリシア木は2バイトコード文字に特化していたので，山下君が開発したSUFARY（Suffix Arrayを用いた文字列検索システム，URL:http://cl.aist-nara.ac.jp/lab/nlt/ss/）を英語用の辞書引きシステムとして用いている．その他の言語の辞書についても，学習データや辞書の使用許諾が得られれば順次公開したいと考えている．

日本語の辞書や学習コーパスを構築するにあたって，常に頭を悩ませるのは，単語の定義である．不思議に思われるかもしれないが，日本語において何

☆1 精度とは，システムが出力した総単語のうち正解コーパスと一致しているものの割合，再現率とは，正解コーパスの単語のうちシステムの出力と一致している単語の割合である．日本語のように分かち書きされない言語では，正解コーパスとシステムの出力の分かち書き結果が異なるため，これらの数値は一致しない．

が基本的な単語かを定義することは容易なことではないのである．たとえば，「お願い」という1つの表現をとっても，これを「お（接頭辞）／願い（名詞）」と考えることもできるが，「願い」を動詞「願う」の連用形の名詞用法と考えると，「願」が動詞の語幹で「い」は活用語尾ということになる．一方，「願いする」とはいえないが，「お願いする」は自然な表現なので，「お願い」を「サ変接続名詞」という品詞に属する1つの単語として登録することも考えられる．学習データにもこの種の揺れは散見され，茶筌の辞書にも多くの不整合が残されている．

最近，形態素解析のためだけでなく，音声処理，言語処理，国語学などさまざまな異なる背景を持つ研究者が共通に使えるような日本語辞書を作ろうとする動きがあり，伝康晴氏（現 奈良先端大，2000年10月より千葉大学）の呼びかけにより，科技庁開放的融合研究「話し言葉工学」の形態素グループ（国立国語研究所，郵政省通信総合研究所），IPA擬人化音声対話エージェントプロジェクト（代表：嵯峨山茂樹（北陸先端大））の音声合成・読みグループなどいくつかの研究グループのメンバによるメーリングリストが作られ，統一辞書へ向けての議論が開始された．

注意しておきたいのは，統一辞書の目的は単語の定義を1つの考え方に限定するものではない．重要なのは，全体の整合性であって，その中にいくつもの理論や考え方が共存できるようなことができればそれがベストである．この目的のためには不十分であると思うが，茶筌の辞書では，単語の定義の中にそれを複合語としてより細かい語構成を記述するための機能がある．たとえば，「お願い」の例では，これを「サ変接続名詞」として辞書登録し，その語構成が接尾辞と一般名詞から成り立っていることを記述することができる．利用者は，解析結果としてどちらの出力を得るかを選択できるようになっている．

また，何人かの学生諸君が，茶筌本体のモジュール化を進めることを考えてくれており，茶筌自体がより分かりやすく手を入れやすいシステムに生まれ変わるかもしれない．機能面としては，今まであまり力を入れてこなかった未知語処理のモジュールを充実させていきたいと思っている．また，学習方法として，Support Vector Machineの利用も試みている．これは，速度の面から考えて茶筌に直接取り込むのは難しいと思うが，学習コーパスの誤り検出や整合性の維持のため，また，現在の学習法の弱点を知る上で有効に利用できるかもしれないと期待している．

学術研究と実用化の狭間で（まとめにかえて）

日本語処理の基盤となる形態素解析システムの開発にかかわりあってもう10年以上の時間が経ってしまった．当初，日本語の形態素解析は技術的には終わった研究と認識されていたこともあり，学術研究ではなく，使えるものを作ると割り切って考えていた．これにかかわってくれる学生諸君が，それを通していかに研究上の成果をあげられるかということに常に腐心してきた．しかし，どのような問題も，それが100％疑問の余地なく解決されているのでなければ技術的にも終わっていない問題であり，さまざまな関連研究を行うことができた．ただ，よく考えてみると，茶筌の開発に関係して行ったさまざまな学術成果物（学術誌論文，国際会議論文等）は，我々が茶筌を開発する上で重要な知見になったのは確かであるが，いずれもそのままの形で茶筌の中には生き残っていない．また，茶筌そのものについては，我々は論文を1編も書いていない．学習モデルについての浅原君の国際会議論文[Asahara 00]が，そのままの形で茶筌に利用された（ただし，本体ではなく学習プログラムとして）唯一のものである．実用化（研究）と学術研究を両立させることは大変に難しいと思う．

最後になったが，これまで，JUMANと茶筌の開発を通じて，開発者および利用者としてさまざまな貢献や意見をくださったすべての方々に（個々の名前を記すことはできないが）心から感謝したい．なお，茶筌の本体と詳細情報は，以下のURLから手に入れることができる．

茶筌ホームページ: http://chasen.aist-nara.ac.jp/

参照文献
[Asahara 00] Asahara, M. and Matsumoto, Y.: Extended Models and Tools for High-performance Part-of-speech Tagger, Proceedings of the 18th International Conference on Computational Linguistics COLING 2000, pp.21-27 (July 2000).
[Haruno 98] Haruno, M. and Matsumoto, Y.: Mistake-Driven Mixture of Hierarchical Tag Context Trees, Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics, pp.230-237 (1997).
[北内 97] 北内　啓, 山下達雄, 松本裕治: 日本語形態素解析システムへの可変長連接規則の実装, 言語処理学会第3回年次大会論文集, pp.437-440 (1997).
[松本 91] 松本裕治, 奥村　晃: 並列論理型言語による探索問題のプログラミング－層状ストリーム法の拡張－, 情報処理学会論文誌, Vol.32, No.7, pp.897-905 (July 1991).
[Yamashita 00] Yamashita, T. and Matsumoto, Y.: Language Independent Morphological Analysis, Proceedings of 6th Applied Natural Language Processing Conference, pp.232-238 (2000).

（平成12年9月24日受付）