形態素解析システムの機能分割と再利用を目指して

山下達雄 (tatuo-y@is.aist-nara.ac.jp)

奈良先端科学技術大学院大学 自然言語処理学講座

「言語資源の共有と再利用」シンポジウム 1999.2.1-3

last update 99/01/28 21:32

---

1. はじめに

他の自然言語処理処理技術と比べ形態素解析技術は実用に近い場所を占めてお り、それゆえ、形態素解析システムに対する現場からの 使い勝手の向上のための様々な要求が多い。 また、すでに成熟している技術とは言え、 解析精度や速度の向上のために様々な手法を試みる余地はあり、 そのための技術的な拡張要求もある。 本研究では、 使い勝手の向上や技術的な拡張を容易に行うことのできる形態素解析処理システム、 つまり「いじりやすい形態素解析システム」 の枠組の提案とそれに基づいた 形態素解析ツールキット LimaTK の実装を行った。

当研究室では、形態素解析システム「茶筌」[茶筌 97]の開発を行ってきたが、 開発時には様々な拡張要求への対応についてほとんど考慮していなかったため、 アドホックな拡張の影響により拡張作業が行いにくい場面もある。 また、システムに関するドキュメント不足もあり、関係者以外、容易に手をつけられない状態にある。 この経験から、「いじりやすい形態素解析システム」の開発には、 拡張が容易な構造と開発関係者でなくともソースの理解できるドキュメントが必要になることが分かる。

本稿で提案する形態素解析ツールキットの開発方針を以下に示す。

単純化

形態素解析システムのモジュール分割

自然言語処理システムは保守の容易な小さな部品の組合せで実現すべきであると考える。 形態素解析処理を、トークン認識、辞書検索、接続処理、最適解探索 等の モジュールに分割し、独立性を保つように実装しているため、部品単位での拡張が容易になる。 例えば、高速検索アルゴリズムを試してみたい場合は辞書検索モジュールの中身だけを 置き換えれば良いし、シソーラスを用いた接続処理をする場合は接続処理モジュー ルの中身だけを置き換えれば良い。
また、 形態素解析と他の処理とのプログラムレベルでの融合も容易である。 例えば、形態素解析の結果を使って、構文解析をするのでは なく、これらを同時に行うシステムを構築する場合など。

扱うデータの単純化

システムで扱うデータを極度に単純化することで以下の利点が得られる。

- 解析処理の高速化
- プログラムの簡素化

欠点として、単純化により、人が見て分かりにくいフォーマットになるということが挙げられるが、これは変換フィルターを用意すれば解決する問題である。

柔軟性
: ここで言う柔軟性とは、 ある特定の言語や用途に特化せずに様々な使い方ができるという意味で、 これは、前述の単純化により実現される特徴とも言えよう。 例えば、 日本語や英語などに限らず様々な言語の形態素解析を行う統一的な枠組(文献 [山下,松本 98])や、 文節区切りなど形態素解析以外の処理(当研究室にて現在研究中)への利用などである。 また、部分的にタグが振られているテキストに対してそのタグ情報を利用して残りの部分の形態素解析を行うというタグ付き文解析機能により、 品詞タグ付きコーパスの品詞体系の変換や、 パターンマッチングによる品詞推定と形態素解析の融合など、 単なる形態素解析以上の処理を行うことができる。

文書化
: 知識の共有をスムーズにするためには不可欠なものである。 各モジュールのインターフェースの解説だけでなく、 アルゴリズム、ソースなどの技術的な解説、 辞書や接続表のデータフォーマットの解説も重要視する。

システムの「単純化」により技術的な拡張が行いやすくなる。 つまり、 プログラムを改良して様々な用途に用いるユーザにとっての 「いじりやすい形態素解析システム」と言える。 また、データの「単純化」「柔軟性」により使い勝手の向上が容易になる。 システムの中身(プログラム)は気にせず道具として様々な用途に用いるユーザー にとって「いじりやすい形態素システム」と言える。 「文書化」はこのどちらのユーザにとっても重要な要素であることは言うまでもない。

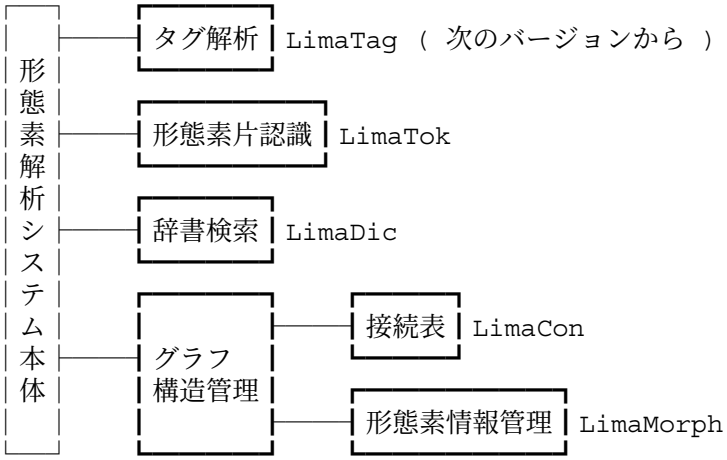
2. モジュール

LimaTK は Language Independent Morphological Analysis Tool Kit の略で、 コスト最小法に基づく、多言語対応の形態素解析ツールキットである。

コスト最小法とは、 検索結果である各形態素間の接続(枝)と形態素そのもの(ノード)にコストを与えて、 文頭から文末までのこれらの合計値が最小の経路(パス)上の形態素を 最適解とする方法である。 コスト最小法は確率モデルによる最適解選択法と同等の性能を持つ。 つまり、品詞タグ付きコーパスから学習した確率値をコスト値に変換して用いることにより、確率モデルによる形態素解析も可能になる。

ツールキットの各モジュールは C++ のクラスとして実装した。下図に構成を示す。 入力文整形や出力フォーマット処理などの解析には直接関係ない処理は、 ツールキットには含めていない。 そのような処理は、ツールキットを使う側 (形態素解析システム本体) や、 形態素解析システム外部のフィルターなどで行うべきであると考えている。 未定義語処理は、言語や処理方式の多様性により一般化するのが困難で、 現在はツールキット側ではモジュールを用意しておらず、 形態素解析システム本体で行うようにしている。 未定義語処理モジュールの作成は今後の課題である。

本章では、各モジュールの働きについて解説する。 実際のメソッド等の解説は、文献 [山下 99] を参照のこと。



LimaTK の構成

2.1. タグ付き文の解析

部分的にタグが振られているテキストに対してそのタグ情報を利用して、 効率的な形態素解析を行うという機能である。

例えば、以下のような「固有名詞タグだけが付いたコーパス」が蓄積されているとすると、 その固有名詞の部分には品詞と区切りの曖昧性がないので、その分解析精度は向上することが予想できる。

> <W POS="固有名詞">DPマッチング</W>アルゴリズムは検索対象データの大きさに比例した計算時間がかかるのに対し、 <W POS="固有名詞">トライ</W>を用いた <W POS="固有名詞">Error-tolerant Recognition</W> アルゴリズムは検索対象データの大きさに依存せず高速に類似検索が行える。

以下のような用途が考えられる。

- 品詞タグ付きコーパスの変換： ある品詞体系の品詞タグ付きデータを他の品詞体系に変換するとき、１ 対１で品詞の対応がつかない場合がある。 このような曖昧性のある形態素には、複数の品詞タグを振っておき、 そのタグ情報を利用して品詞の曖昧性を解消する。
- 品詞タグ付きコーパスの構築： 品詞タグ付きコーパスを蓄積するときに、 確実な品詞タグ、曖昧な複数の品詞タグ、品詞指定しない、というように 自由にタグが付与できる。 これにより、困難な部分の解析をあと回しにすることができる。
- 外部のシステムによる解析の補助： 未定義語処理が外部でも可能になる。
- 外部のシステムによる解析の補助： パターンマッチングで、専門用語だけタグを付けてから解析すれば、 辞書を変更せず済む。 また、専門用語にだけなにか特殊なタグ(用語集へのリンクなど)が 付与されている文書では、 そのタグ領域を名詞などにみなすことによって効率的に解析が行える。
- 外部のシステムによる解析の補助： パターンマッチングなどの手法で、 テキスト全てに曖昧性を持たした品詞タグを付与しておけば、 LimaTK に最適パス選択作業をさせれば良いので、状況によっては有用。

現在利用できるタグの種類を以下に示す。

必ず区切られる場所を指定

日本語をある程度わかち書きしたりする場合、そのわかち書き情報を活かしたい。わかち位置情報が必ず形態素の区切りになるということを意味し、 わかち書きされた領域内の文字列が必ず一つの形態素になるということではない。 「茶筌」[茶筌97]と同じように、 タグではなくブランク文字で「必ず区切られる場所」を定義する方針。

例： 今日 は 学校に 行きました。

絶対に区切られない領域を指定

開始タグと終了タグでくくる。この領域の持つ属性の種類を以下に示す。

領域のみを指定：一つの形態素になる文字列ということだけを指定

属性の指定なし。 その領域の文字列で exact match で形態素辞書を引く。領域内の文字列が必ず一つの 形態素になる。つまり、区切りの曖昧性のみを解消する。

例： 今日は<W>学校</W><W>に</W>行きました。

領域と品詞を指定：一つの形態素として確定

品詞が指定されるので、辞書を引く必要がないが、 品詞以外の付加情報を得るために exact match で形態素辞書を引く。 区切りと品詞の曖昧性が解消される。

例： 今日は<W PL=4>学校</W><W PL=12>に</W>行きました。
PL は POS Label の略。4 は名詞、12 は助詞。

領域と複数の品詞を指定

この領域の文字列は、品詞の曖昧性がある形態素だということを指定する。 品詞が指定されるので、辞書を引く必要がないが、 品詞以外の付加情報を得るために exact match で形態素辞書を引く。 区切りと曖昧性が解消される。 品詞の曖昧性は減少される。

例： 今日は<W PL=4,8>学校</W><W PL=4,12,20>に</W>行きま した。

現在考案しているものを以下に示す。

- タグ領域の属性に品詞以外のもの (「読み」など) も指定できるようにする。
- 絶対に区切られない領域だけではなく、 絶対に区切ってはいけない位置を指定できるする。

タグ解析モジュールでは、元の文から以下のデータを得る処理を行う。

- タグを抜いた文
- タグ領域の情報

システム本体で、このタグ領域の情報を用いて、 辞書検索の切替えや形態素作成等の処

理を行う。

2.2. 形態素片認識

わかち書きされる言語とそうでない言語 の形態素解析処理における差異の一つに辞書検索のタイミングが挙げられる。 これは、 解析対象文中のどの位置から辞書を引き始めたらいいのか、 どの位置で辞書を引き終えたらいいのかという問題である。 LimaTK では、「形態素片」という辞書検索単位を用い、 この問題を解決し、言語に依存しない辞書検索を実現している(文献 [山下, 松本 98])。 ここでは形態素片について簡単に説明する。 文献 [山下 99] に詳細な解説があるので参照されたい。

形態素辞書検索の実装の際には、 わかち書きされる言語では単純な exact match、 わかち書きされない言語では common prefix search が一般的に用いられる。 例えば英語のようなわかち書きのされている言語では、 ‘‘This is my pen.’’ のように単語がブランクや記号文字で明確に区切られているので、 ‘‘This’’, ‘‘my’’ といった文字列ごとに、 その文字列が辞書に存在するか否かという基準で 形態素辞書検索を行う(exact match)のが単純で効率的な実装である。 これに対し、 日本語などのわかち書きされていない言語では、 区切り認識も同時に行う必要がある。 そのため、解析対象文の先頭から一文字ずつずらしながら、 その文字位置から始まる文字列と一致する全ての単語を辞書から検索 (common prefix search) するのが現実的である。

LimaTK では、わかち書きされる言語の扱いを わかち書きされない言語と同じ枠組 (common prefix seach) で扱うことにより辞書検索処理の統一を行う。 そのためには辞書検索単位を統一する必要がある。 例えば、英語の「単語」と日本語の「文字」を同じ単位として扱えば、 辞書検索処理を統一できる。 この検索単位を「形態素片」と呼ぶ。

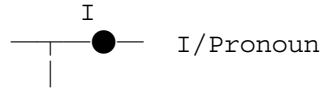
形態素認識モジュールでは、 個別の言語の形態素片とブランク文字 (形態素片の区切りを示し、辞書の見出し語の先頭と末尾には現れない文字) の定義を与えれば、 あらゆる言語の形態素片を認識できる。 以下に認識例を示す。形態素片は [ ] 、ブランクは _ で表される。

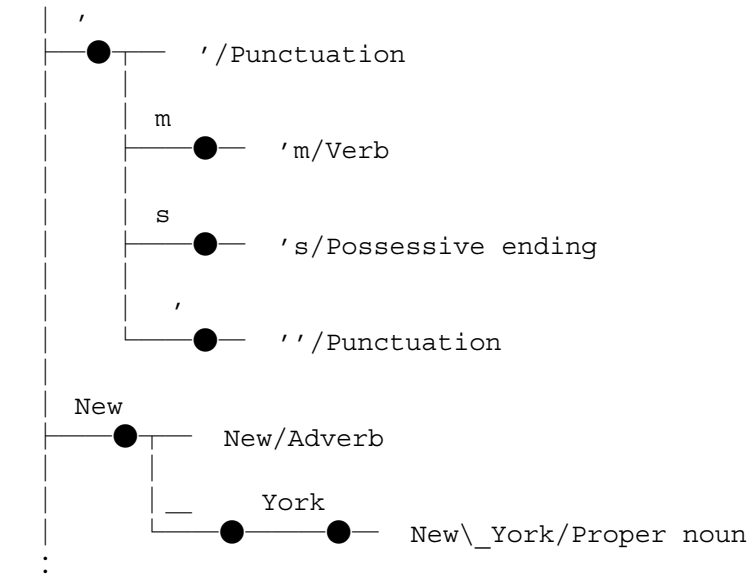
例： I’m John. → [I][’][m]_[John][.]
例： ございます。 → [ご][ざ][い][ま][す][。]

一形態素ずつずらしながら common prefix search を行うことで、 言語に依存しない辞書検索が実現できる。

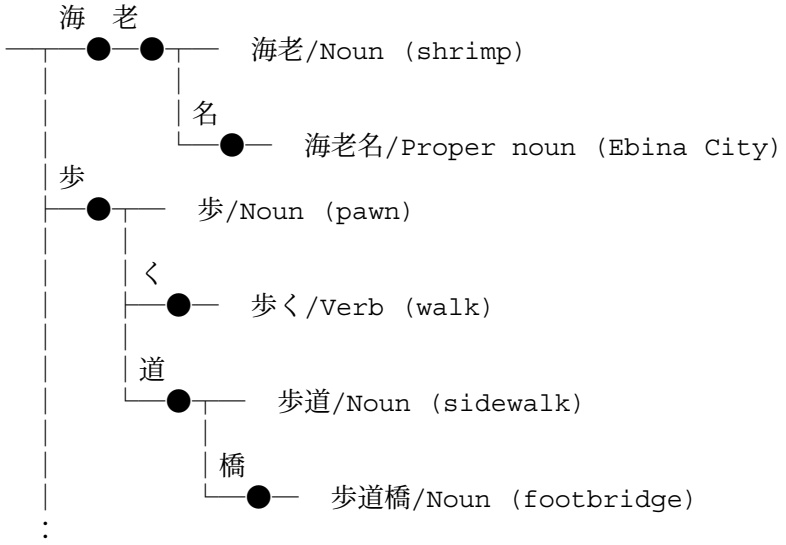
2.3. 辞書検索

辞書検索モジュールには、 前節で述べた common prefix search と、タグ付き文解析で利用される exact match を行うメソッドが用意されている。 common prefix search には一般に TRIE と呼ばれるデータ構造が用いられる。 LimaTK では、形態素片をラベルとしたノードを持つ TRIE を仮定している (下図に、英語と日本語の TRIE を示す)。 現在のバージョンでは、suffix array [Manber and Myers 90] と呼ばれるデータ構造を用いた、文字列検索ライブラリ SUFARY [SUFARY] を使って辞書検索を行っている。





English TRIE Structure Dictionary



Japanese TRIE Structure Dictionary

2.4. グラフ構造管理

辞書検索の結果である形態素情報は、グラフ構造のデータとして格納される。 LimaTK は、一般的な日本語形態素解析システムで採用されている bigram モデルだけではなく、さらに表現力のある状態遷移モデルの形態素解析も可能である。 グラフの各ノードは形態素の情報だけでなく、状態の情報も合わせ持つことになる。

2.5. 接続表

状態遷移モデルにおける遷移コストを扱うモジュールであり、「現在の状態」と「入力」を引数に「遷移コスト」と「遷移先状態」を取り出すためのメソッドがある。

3. データフォーマット

LimaTK では、解析の際に、 形態素辞書と接続表という二つのデータファイルが必要である。 どちらも、 必要最小限の要素だけから構成される、 内部フォーマットと言ってしまっても良いような、 非常にシンプルなフォーマットを規定した。 このような至極単純なフォーマットは、 人間にとって分かり難いという欠点があげられる。 しかし、Perl 等の簡易な言語を用いれば、 人間にとって分かりやすいフォーマットで辞書等を蓄積していても、 簡単にフォーマットの変換ができるので、何の問題もない。 普段は、人間にとって分かりやすいフォーマットで辞書等を扱えば良い。 LimaTK が直接扱うフォーマットがきちんとドキュメント化されているという点が重要なのである。

形態素辞書に必要最小限のデータを考えてみる。 文字列検索するためには「見出し語」必須である。 そして、解析のためには「品詞」と「形態素コスト」も必要である。 読み、活用型、活用形、意味情報などの情報は解析には必要ない。 しかし、出力時には必要かもしれないので「その他の情報」として一つの文字列にまとめておくのが良いであろう。 結局、形態素辞書に格納しておく必要のある情報は以下の4つだけである。

> 見出し語, 品詞, 形態素コスト, その他の情報

これに基づき、LimaTK で用いる単純な辞書フォーマットを定めた。

MOZDF-98-2

> 最初の行 `#MOZDF-98-2\n`
> それ以降 `見出し語\0コスト\0品詞\0その他の情報\0\n`

> '\n' は改行文字、'\0' はコード 0x00 の文字を表しま す。「見出し語」「その他の情報」は文字列です。'\0', '\n' が含まれていてはいけません。「その他の情報」はシステ ム内部でもそのまま文字列として扱われます。「コスト」「品詞」は数 字(0-9)で構成される文字列で、内部では整数(int)として扱われます。

MOZDF-98-1

> 最初の行 `#MOZDF-98-1\n`
> それ以降 `見出し語\0コスト\0左属性\0右属性\0その他の情報\0\n`

> MOZDF-98-2 とは、品詞情報が二つ格納できる点が異なります。 品詞情報「左属性」「右属性」は数字(0-9)で表され、内部では整数 (int)として扱われます。 EDR 日本語単語辞書 [EDR 95] などのデータが利用できます。

形態素辞書の例を以下に示す。

```
#MOZDF-98-2
日本\010\0147\0[Y:ニホン  POS:名詞-固有名詞]\0\n
得る\0207\0100\0[Y:ウル  H:動詞-自立/五段・ラ行/基本形]\0\n
```

活用処理などをシステム内部で行う場合は、 活用型などの情報が辞書も独立な情報として必要かも知れない。 しかし、そのような場合は、 後で「その他の情報」のフォーマットを新たに考慮すれば良いだけである。 LimaTK では活用する言語に特化したデータ構造は柔軟性に支障をきたすと考え、 必須の情報のみで辞書フォーマットを定義した。

接続表に必要な情報は、 引数である「現在の状態」「入力」と 返し値である「遷移コスト」「遷移先状態」の4つである。 様々な用途に利用できるように、これらの4つのデータは数値として扱う。 実際の接続表ファイルフォーマットを以下に示す。

MOZCF-98-1

```
最初の行 #MOZCF-98-1\n
  二行目 状態の最大値\s入力の最大値(\sコメント)\n
それ以降 現状態\s入力\sコスト(\s次状態)(\sコメント)\n
```

'\n' は改行文字、'\s' は半角スペース文字を表します。 括弧で括られているものは省略可能です。 「現状態」「入力」「次状態」は数字(0-9)とマイナス記号('-')で構成される文字列で、 内部では整数(int)として扱われます。
「次状態」を省略すると、「次状態」は「入力」と同じとみなされます。
コストはデフォルトでは -1 で、これは「非常に接続しにくい」を意味します。 コストを -2 に設定すると「絶対接続しない」を意味します。
「現状態」「入力」にマイナスの値を指定するとワイルドカードになります。 0 は文頭/文末を表します。
曖昧性がある場合、ファイルの後の方の接続規則が優先されます。
接続表の例を以下に示す。

```
#MOZCF-98-1
356 150        # table size = 356 x 150
37 5 116 120   # 助詞(37)と連体詞(5)のコストは 116 で、次状態は 120
120 92 30 52   # 状態 120 の次に名詞(92)がくると、コストは 30、次状態は 52
120 37 243     # 状態 120 の次に助詞(37)がくると、コストは 243、次状態は 37
92 -1 45       # 名詞(92)は全ての品詞とコスト 45 で接続　次状態は各々の品詞番号
92 5 -2        #         だけど、名詞(92)は連体詞(165)とは絶対接続しない
-1 0 1         # 全ての状態がコスト 1 で文末(0)につながる
```

4. おわりに

形態素解析ツールキット LimaTK の 技術的な詳細については、 「MOZ と LimaTK の説明書 (文献 [山下 99])」 を御参照下さい。 開発方針の一つである「文書化」を体現すべく、モジュールのインターフェース、アルゴリズム、ソース、データフォーマット、 品詞タグ付きコーパスからの辞書の作成法など、 徹底的に解説しています (至らない部分はありますが)。 もし、興味を持っていただけたら是非御一読下さい。

LimaTK のパッケージは以下の URL から入手先できます。 最新情報、辞書、説明書、周辺ツールも入手できます。

http://cl.aist-nara.ac.jp/~tatuo-y/ma/

参考文献

[茶筌 97] 松本裕治, 北内啓, 山下達雄, 今一修, 今村友明, "日本語形態素解析システム『茶筌』 version 1.0 使用説明書," NAIST Technical Report, NAIST-IS-TR97007, February 1997. (入手先)

[山下 99] 山下達雄, "MOZ と LimaTK の説明書," LimaTK パッケージ付属, 1999. (PS 640k)

[山下, 松本 98] 山下達雄, 松本裕治, "言語に依存しない形態素解析ツールキットの開発," 情報処理学会研究報告 98-NL-129, pp.17-22, November 1998. (PS+gzip 59k)

[Manber and Myers 90] Udi Manber and Gene Myers, "Suffix Arrays: A New Method for On-Line String Searches," in 1st ACM-SIAM Symposium on Discrete Algorithms, pp.319-327, 1990.

[SUFARY] "高速文字列検索ライブラリ SUFARY," http://cl.aist-nara.ac.jp/lab/nlt/ss/.

[EDR 95] "EDR電子化辞書仕様説明書," 日本電子化辞書研究所, 1995. (HTML)