4

英語構文解析システム「Apple Pie Parser」

関根　聡
New York University
sekine@cs.nyu.edu
http://cs.nyu.edu/cs/projects/proteus/sekine/

英語の構文解析

我々が日常使っている日本語や英語などの言語は，単語や句，文節といった単位からなっており，それらの間には文法的な関係がある．これらは一般的に構文と呼ばれており，与えられた文の構文的な情報を解析する段階が，本稿で紹介する構文解析である（多くの読者は，高校の英語の時間に，例文を与えられて，「この文の主動詞と直接目的語，間接目的語は何かを答えなさい」といった質問に頭を悩ませた経験があるであろう．簡単に言うと，それを計算機でやらせようというのが課題である）．計算機が言葉を人間同様に扱えるようになるには，このような構文情報をきちんと解析することが必要であり，構文解析は，現在，自然言語処理の中心的な研究テーマの1つになっている．

日本語では文節というものが広く認知され，その文節間の係り受けという関係で構文を認定することが多いが，英語では主に句構造という表現形式で表現されることが多い．句は，名詞句，動詞句，前置詞句といった文法的な役割による単位で，英語タグ付きコーパスとして有名なPennTreeBankでは約20種類の句が設定されている．句構造表現では文は名詞句と動詞句からなり，前置詞句は，前置詞と名詞句からなることなどを表現する木の構造をしており，句の範囲を括弧で囲った形で表現されることが多い（図-1参照）．

入力文に対して，このような構文木を出力する解析段階を構文解析と呼ぶ（品詞タグ付けや単純な名詞句の抽出などを別段階として解析を行うシステムもあり，その際にはそこから構文木を出力するまでの段階を構文解析と呼ぶこともある）．

1990年代前半までは，このような構文木を出力するシステムは人間が文法規則を "S->NP VP" といった句構造文法の形式で表現し，その規則に基づいて文を解析することが多かった．しかし，ペンシルバニア大学でPennTreeBankが作成されてから状況が変わった．PennTreeBankとはウォールストリートジャーナル紙の5万文の文に対し，図-1のような構文木を人手で付与したものである．1990年代中旬からは，PennTreeBankなどの大規模な言語資源から文法規則などを自動的に抽出し，それを使って構文解析をしようという試みが始まり，成功を収めている．その1つが，本稿で紹介するApple Pie Parserである．

本稿では，まず，インストールから実行までの説明をし，それから，生い立ちと技術的な内容，高度な使用方法の紹介をする．また，最後にApple Pie Parser以外の英語のフリーソフトのリストも紹介する．

Apple Pie Parserのインストールと実行

ここでは，Apple Pie Parserのインストールと実行方法について紹介する．OSは基本的にUNIXを想定するが，MS-DOS（Windows）についても説明する．ダウンロードには，ftpを使う方法とWebのホームページからダウンロードする方法の2種類が可能である（MS-DOSユーザはWebからのみ可能である）．ftpを利用する場合には図-2にあるように入手できる．Webを利用する場合には，http://cs.nyu.edu/cs/projects/proteus/app/を開き，中ほどにあるAPP5.9.tar.gzをダウンロードする．また，MS-DOSユーザは，その下にあるthe executableも同時にダウンロードする．ダウンロードしてからは図-2の「ソースコードを取り出す」以降に従うこと．

文章：The federal government suspended sales of U.S. savings bonds.

括弧による構文表現：(S (NP (DT The) (JJ federal) (NN government)) (VP (VBD suspended) (NP ((NP NNS sales)) (PP (IN of) (NP (NNP U.S.) (NNS savings) (NNS bonds))))) (. .))

木による表現　：

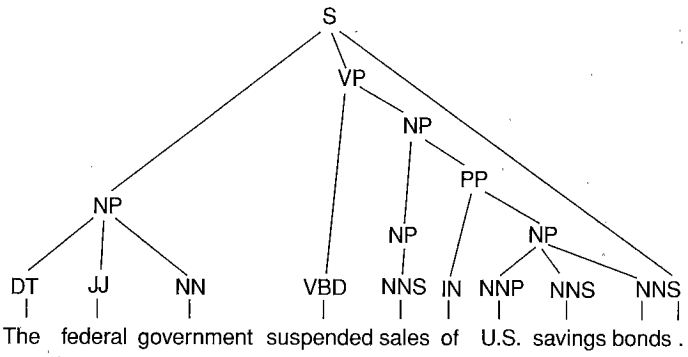

図-1　英語の構文表現

引数，パラメータの設定

Apple Pie Parserでは，ユーザが指定する引数とパラメータファイルが存在する．標準的に使用するにあたってはそれらを設定し，変更する必要はない．引数では，パラメータファイル，辞書，文法ファイル，ログファイル名の設定，最大文長，出力形式，バッチモードの設定などができる．パラメータファイルでは，辞書，文法ファイル名，出力形式の設定や，未知語候補，品詞変換規則，単語に対する品詞の強制的な設定などができるようになっている．これらについては，マニュアルに詳しいが，代表的なものを図-3に紹介する．

Apple Pie Parserの生い立ち

Apple Pie Parserは著者が，1993年ごろから研究開発を始め，1995年のInternational Workshop on Parsing Technologiesで発表，1996年に一般公開を始めた．中心となっているコーパスベース学習のアイディアは著者が1989年にEDRで英語の文法を自分で作成していたころにさかのぼる．同様なアイディアは，著者が論文を発表するよりも前の1992年にも発表されているが，Apple Pie Parserは実装性という点で優位性がある．

公開して4年以上経ち，Apple Pie Parserよりも精度のいい構文解析の手法が多数発表されているが，一般公開されている構文解析としてはいまだ，性能が高い方であると思われる．ダウンロードするために登録などをする必要が一切ないため正確なダウンロード数やユーザ数は把握できないが，READMEには「その旨著者にメールをお送り下さい」という内容のことが書かれており，そのようなメールは現在のところ，10カ国以上から100通程度きている．実際にはメールを送ってこない人も多いと思われるので，ダウンロード数はこの数倍程度あるのではないかと思われる．また，バグレポートや他のプラットフォームへの移植などをボランティアで行ってくれた方も多くあり，そのたびにバージョンアップなどを繰り返してきた．それらの方々に感謝したい．特にMS-DOS（Windows）への移植では，フランスの方が最初に行ってくれたが，現在は慶應大学の鳥原信一さんが移植してくれたものをバイナリー形式で公開している．

具体的な利用事例としては，MITの英韓機械翻訳システムや，その他，英語教育，英語文法の分析，情報検索システムの一部，情報抽出の一部などに使われているという報告を受けている．

Apple Pie Parserの技術的内容や種々の条件における実験結果などは著者の博士論文に詳しい．著者のホームページからダウンロードできるが，冊子になっているものを希望する方があれば，著者までメールをいただきたい．

よく，Apple Pie Parserという名前について質問を受けるが，この名前には深い（？）理由がある．英語で書くと，Apple Pieもこの構文解析システムも "A product of fruit of tree" と表現できる．英語に詳しい方ならお気づきかもしれないが，"fruit"には，果物という意味と，成果という意味があり，"tree" は実際の木とPennTreeBankの木という2つの意味を掛けている．つまり，食べる方のApple Pieは，「木になる果物からできたもの」であり，この構文解析システムは，「PennTreeBankという木（構文木）の成果としての製品」という意味がある．果物に「りんご」を選んだのは，ニューヨークの別名が "Big Apple" であることに由来する．厳密に英語的に正しいかどうか知らないが，著者の洒落を理解していただけたら光栄である（少なくとも英語ネイ

＊ダウンロード
Apple Pie Parser用に適切なディレクトリを用意し（以下ではこのディレクトリをAPP_DIRとする）そこに移動し，ダウンロードを行う．以下の実行例ではユーザの入力する部分を太字で示している（UNIX/MS-DOS）．

```
> cd APP_DIR
> ftp cs.nyu.edu
Connected to cs.nyu.edu.
220 cims.nyu.edu NcFTPd Server (free educational license) ready.
Name (cs.nyu.edu:sekine) : anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: your-email-address
230-You are user #12 of 50 simultaneous users allowed.
230-230 Logged in anonymously.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub/local/sekine
250 "/pub/local/sekine" is new cwd.
ftp> bin
200 Type okay.
ftp> get APP5.9.tar.gz
local: APP5.9.tar.gz remote: APP5.9.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for APP5.9.tar.gz (1064399 bytes).
226 Transfer completed.
1064399 bytes received in 1.26 secs (8.3e+02 Kbytes/sec)
ftp> quit
221 Goodbye.
```

＊ソースコードを取り出す．（UNIX/MS-DOS）

```
> gzip -dc APP5.9.tar.gz | tar xvf -
APP5.9/
APP5.9/README
APP5.9/bin/
APP5.9/bin/app.prm
...（省略）...
APP5.9/src/extern.h
APP5.9/src/version.h~
```

＊システム対応の設定
大部分の場合，これを省いても動作はするはずであるが，メモリが小さい，特殊なOSを使用している場合には行う必要がある．また，MS-DOS実行する場合には，このシステム対応設定と次のmakeは行わず，最後の「MS-DOSで実行」まで飛ばす．
APP5.9/src/config.hを適切に編集する．
もし，使用するマシンのメモリサイズが大きい場合にはALLOCATE_ANODEの数字を適当に大きくすると解析精度が向上する（ただし，長い文に対する実行時間は長くなる）．

＊make (UNIX)
システム内にgccがなく，ccしかない場合などには，Makefileの該当部分を書き換える必要がある．

```
> cd APP_DIR/APP5.9/src
> make
gcc -g -c debug.c -o debug.o
gcc -g -c dict.c -o dict.o
gcc -g -c func.c -o func.o
...（省略）...
gcc -g -o ../bin/apps debug.o dict.o func.o grammar.o heap.o param.o parse.o print.o tag.o simple.o -lm -lc
```

＊UNIXでの実行

```
> cd APP_DIR/APP5.9/bin
> app
Apple Pie Parser (5.9) April.4.1997 S.Sekine (NYU)
------
Type '*help' for help, or type your sentence after prompt
Now loading dictionary and grammar...
------
>>（自由に英語の文章を入れてみる）
>> I downloaded the apple pie and it worked.
(S (SS (NPL I) (VP downloaded (NPL the apple pie))) and (SS (NPL it) (VP worked)) -PERIOD-)
>> *param
> PRINT_STYLE 2
>> I love you.
(S (NPL (PRP I)) (VP (VBP love) (NPL (PRP you))) (. -PERIOD-))
>> *quit
```

＊MS-DOSでの実行
上記ソースコードをダウンロードし，展開した上で，（makeは行わないでよい）MS-DOS用にダウンロードした実行形式(app.exe)をAPP_DIR\APP5.9\binに置き，実行してみる．

```
> cd APP_DIR\APP5.9\bin
> app.exe
```

図-2　Apple Pie Parserのインストールと実行

```
<引数>
-p filename ：パラメータファイル名の設定
-t n ：プリント形式の設定
  n=1: PARSED style
       (S (NP I) (VP love (NP you)))
  n=2 COMBINE style
       (S (NP (PRP I)) (VP (VBP love) (NP (PRP you))))
  n=3 TAGGED style
       [ I/PRP ] love/VB [ you/PRP ]
  n=4 POS tagged style
       I/PRP love/VBP you/PRP
-b: バッチモード
  ファイルを入力とする場合このオプションを利用し以下のように実行する.
  app -b < inputfile > outputfile

<パラメータファイル>
PRINT_STYLE:     引数で説明したものと同じ
NO_WARNING:      ワーニングを出力しない
TOO_LONG:        入力文の最大を設定する（単語数）
START_SYMBOL:    トップノードの非終端記号を設定する．たとえば，NPを認識することで文の解析を終えることもできる
SPECIAL_HEAD:    連続文字列のprefixにある単語
FLAG_OOV:        辞書にない単語の推定を行う
TRANS_POS:       強制的に解析前に品詞を変換する
INV_TRANS_POS:   変換した品詞を最終出力では元に戻す
SUP_WORD:        辞書の定義に優先して単語の定義を行う
```

図-3　代表的な引数とパラメータ

ティブにも笑ってもらえたので，意味としては通るのであろう）．また，著者はほかの研究者と共同で，日本語の解析システムとして "Cherry Pie Parser", "Peach Pie Parser" も構築した．一般公開はしていないが，これらについて興味ある方は著者まで連絡をいただきたい．

構文規則の学習

Apple Pie Parserの特徴は，教師つき学習によるデータからの学習にある．

1990年代の前半にPennTreeBankが公開されて，自然言語処理の研究は大きな変換点に立ったといっても過言ではないだろう（このあたりについては，本会誌Vol.41 No.7の特集「ここまできた自然言語処理」にも詳しい）．大量の正解データが作成されたことで，言語現象の知識を作成するにあたって人間の直観だけではなく，実際の大規模データが利用できるようになった．この点に注目して，面白いことをやってみようという動きが1990年代の前半から始まり，著者もその1人であった．

著者の最初のアイディアは，文の品詞列パターンは有限であり，大規模データによって大部分がカバーされているのではないかということであった（英語では，自動的に単語に品詞をつける品詞タガーというものがあり，その正解率は96%程度である）．つまり，ある文の品詞列が与えられたのなら，それとまったく同じ品詞列を持つ文を正解データから探し出し，それとまったく同じ構文構造を出力すれば，構文解析の問題は解けるのではないかというアイディアである．しかしながら，PennTreeBankを調査してみると，この考えは無謀であることがすぐに分かった．PennTreeBankの96%の文を対象に調査した結果，それだけのデータでは，新しい文に対して4.7%のカバー率しかないことが分かった（これは，1つの文をデータから取り出し，それと同じ品詞列の文が残りのデータに存在するかどうかで推測した）．入力の95.3%には答えを返せないようなシステムでは，いかに答えを返せた場合にその答えが正確であっても役には立たない．自然言語の単語頻度などの現象を経験的に表現したZipfの法則を用いると，80%のカバー率を得るためには10の70乗から140乗のデータが必要であると推測された．

最初のアイディアはうまくいかないことが分かったため，多少の一般化を導入してみることにした．最初のアイディアでは文全体をマッチングさせたため，あまりにカバー率が低かったが，マッチングさせる範囲を狭めれば，同じものがデータ中に存在する確率は高くなる．次のアイディアは，文を示す "S"（これは，1文全体だけではなく，一般的な節（clause）を示すのにも使われている）と名詞句である "NP" だけにマッチング範囲を限定し，マッチング対象にS, NPを導入することにより，あたかも，それらの2つだけが非終端記号のようにして構文解析を行うという方

パラメータファイル（gtest.prm）
```
DICTIONARY_FILE gtest.dic
GRAMMAR_FILE gtest.grm
NICKNAME_FILE gtest.nic
```

ニックネームファイル（gtest.nic）
```
1 S
2 NP
3 VP
11 PRP
12 VBP
13 VBN
```

辞書ファイル（gtest.dic）
```
Ich : 11#1
liebe : 12#1
dich : 11#1
habe : 12#1
das : 11#1
gegessen : 13#1
```

文法ファイル（gtest.grm）
```
1 : 2 3 ;
2 : 11 ;
3 : 12 2 ;
3 : 12 2 13 ;
:struct " (VP (VG <1> <3>) <2>) " ;
```

図-4 自分の文法の例

法である．

この手法では一般的なCFG規則に対してより広範囲の情報を利用できるようになるため，精度が向上することが考えられる．このようにした場合のカバー率は，Sが77.2%，NPが98.1%とかなり高いことが分かり，実現可能性の期待が持てた（基本的にこの線でApple Pie Parserは実現されているが，実際はSのカバー率が十分に高くないことが理由で高い精度が出ていない．Sは1文に平均2回出現するので，2文に1回程度カバーされないSが出現することになる．ただし，NPについては比較的精度が良いことが知られている）．

最終的には，2種類のS，To不定詞句，2種類のNPの合計5つの非終端記号を導入し，文法規則を抽出した．統計的な学習を行い，構文木の確率が最も高いものを出力するようにしている．解析には，構文解析の典型的な解析方法であるチャート法に確率を採り入れ，解析の各段階で最良のものだけを記憶するように探索を行うビタビサーチを組み合わせている．また，メモリを使いきり，解析が途中で終了してしまった場合には，その時点で解析できた部分のうち，最も良さそうな部分をつなげて出力とするFitted Parsingを採り入れている．この機能により，どんな入力に対しても出力が行えるようになる．

自分の文法を作ってみよう

構文解析のエンジンは基本的にはチャート法であり，Apple Pie Parserについている文法や辞書ではなく，自分で作成した文法や辞書でも使用できる．ここでは，このような使用を目的とする人にファイルフォーマットなどを理解してもらうため，簡単な辞書と文法を作成する例を挙げる（例では確率をまったく使わない普通のCFG規則を使用しているが，もちろん，確率を使ってもよい）．

以下の2つのドイツ語の文を解析するシステムを作成する．

"Ich liebe dich",

"Ich habe das gegessen"

最初の文は，英語にすると"I love you"であり，2番目の文は，"I have eaten it"である．

ドイツ語では，2番目の文のように本動詞と目的語の位置が入れ替わることがあり，本来，"habe"と"gegessen"で1つの動詞のグループとなるべきである．ここでは，それぞれの文の出力は以下のようになることを目的とする．

(S (NP (PRP Ich)) (VP (VBP liebe) (NP (PRP dich))))

(S (NP (PRP Ich)) (VP (VG (VBP habe) (VBN gegessen)) (NP (PRP das))))

注目すべき点は，Apple Pie Parserでは2つ目の文の解析結果にあるように，単語の順序を入れ替え2つの動詞を一緒にしたような出力ができるようになっていることである．なお，品詞の記号は，PRPは代名詞，VBPは現在形動詞，VBNは過去分詞の動詞，Sは文，NPは名詞句，VPは動詞句，VGは動詞グループを意味する．

これらの文を解析するために，図-4にあるようにパラメータ，ニックネーム，辞書および文法ファイルを用意する．文法ファイルにあるのは，ニックネームファイルとつき合わせれば分かるが，それぞれ，"S -> NP VP", "NP -> PRP", "VP -> VBP NP", "VP -> VBP NP VBN"というCFG規則を意味する．そして，最後の規則が適用された際には，動詞グループをVGという形にまとめて出力する．

実行結果は図-5に示すようになり，期待した通りの出力が得られた．さらに興味のある読者は，"-D 1"というオプションをつけて実行してもらいたい．チャートが生成されていく過程が段階ごとに追っていけるようになっている．

```
> app -p gtest.prm
Apple Pie Parser (5.9) April.4.1997 S.Sekine (NYU)
-----
Type '*help' for help, or type your sentence after prompt
Now loading dictionary and grammar...
-----
>> Ich liebe dich
(S (NP (PRP Ich)) (VP (VBP liebe) (NP (PRP dich))))
>> Ich habe das gegessen
(S (NP (PRP Ich)) (VP (VG (VBP habe) (VBN gegessen)) (NP (PRP das))))
```

図-5　自分の文法の実行画面

＊ 品詞タグつけシステム
+ Eric Brill's Tagger http://www.cs.jhu.edu/~brill/
+ QTAG (Oliver Mason) http://www-clg.bham.ac.uk/QTAG
+ Xerox Tagger
+ TreeTagger http://www.ims.uni-stuttgart.de/projekte/corplex/DecisionTreeTagger.html
+ TnT http://www.coli.uni-sb.de/~thorsten/tnt/
+ CLAWS http://www.comp.lancs.ac.uk/ucrel/claws/
+ TOSCA/LOB tagger ftp://lands.let.kun.nl/pub/tosca/tlbtag/
+ MXPOST ftp://ftp.cis.upenn.edu/pub/adwait/jmx/

＊ 構文解析システム
+ Apple Pie Parser http://cs.nyu.edu/cs/projects/proteus/app
+ LINK Grammar Parser http://www.link.cs.cmu.edu/
+ PC-PATR http://www.sil.org/pcpatr/

＊ 辞書など
+ WordNet http://www.cogsci.princeton.edu/~wn/
+ CHILDES http://childes.psy.cmu.edu/
+ Project Gutenberg http://www.promo.net/pg/index.html
+ GATE http://www.dcs.shef.ac.uk/research/groups/nlp/gate/
+ Alembic http://www.mitre.org/technology/alembic-workbench/

＊ 有料の物，団体など
+ LDC http://www.ldc.upenn.edu/
+ ELRA/ELDA http://www.elda.fr/
+ Alvy Natural Language Tools http://www.cl.cam.ac.uk/Research/NL/anlt.html
+ ENGCG http://www.conexor.fi/analysers.html

＊ リンク集
+ http://www.cs.columbia.edu/~radev/u/db/acl/html/RESOURCES/
+ http://www-a2k.is.tokushima-u.ac.jp/member/kita/NLP/nlp_tools.html
+ http://www.sil.org/linguistics/computing.html
+ http://registry.dfki.de/
+ http://info.ox.ac.uk/bnc/corpora.html
+ http://pioneer.chula.ac.th/~awirote/ling/corpuslst.htm

図-6　他の英語に関するツール

他の英語に関するツール

Apple Pie Parser以外の主な英語のフリーソフトを図-6に列挙する．これらの中には著者がダウンロードしていないものや動作を確認していないものなどがあり，それぞれのインストールなどは著者の責任外である．興味ある読者は自分でダウンロードしてインストールしてみることをお勧めする．

また，学会などで発表されているが公開されていないものでも，その使用目的を明確にし，作者に誠意を持ってお願いすると入手できることもある．興味があるシステムがあった場合には試みてみるとよいと思われる．

参考文献
1) Apple Pie Parser Homepage: http://cs.nyu.edu/cs/projects/proteus/app/
2) Sekine, S.: Corpus-based Parsing and Sublanguage Studies, Ph.D thesis (New York University) (1998).
3) Sekine, S. and Grishman, R.: A Corpus-based Probabilistic Grammar with Only Two Non-terminals, Proceedings of the 4th International Workshop on Parsing Technologies (1995).
4) The Penn Tree Bank Project, Homepage: http://www.cis.upenn.edu/~treebank/home.html

(平成12年9月28日受付)