植物に関する自由形式説明文からのJSON形式テキストの自動生成

著者	山本 富士男
雑誌名	言語資源活用ワークショップ発表論文集
巻	5
ページ	124-135
発行年	2020
URL	http://doi.org/10.15084/00003151

植物に関する自由形式説明文からの JSON 形式テキストの自動生成

山本　富士男（神奈川工科大学）†

Automatic generation of JSON-formatted text from a free-form descriptive text about plants

Fujio Yamamoto (Kanagawa Institute of Technology)

要旨

電子図鑑等において柔軟な植物検索を可能とするため、日本語の自由形式による植物の特性記述から、コンピュータで処理しやすい JSON 形式テキストを自動生成することを試みた。まず、それらの自由形式テキストに係り受け解析を施した。次に、葉と花と果実についてそれぞれ事前に設定した数種類のタグに対して、そのバリュー（値）を係り受け解析結果の木構造をたどりながら設定できる方式を考案した。さらに、自動生成された JSON を、植物を表す（JAVA の）オブジェクトのストリームに変換して、マップやフィルタ処理を施すことにより、多様な検索を効率的に実行できることを示した。

１．はじめに

近年、AI（人工知能）技術の発展によって、特に画像の認識は身近なものとなった。例えば、風景や動植物の写真を与えると、高い精度でその具体的な場所や名前を教えてくれる。植物に関して言えば、画像認識サービス Google Lens に、身近な草花を撮影して入力すると、「サクラソウ」とか「ヤマブキソウ」などと的確に表示される場合が多い。しかし、植物の名前や特性の見つけ方はこれに限らない。一例としては、葉や花や果実の特性を指定して該当する植物を検索したい場合は、このような画像認識による方法ではなく、電子植物図鑑などの中の記述文から探すことになる。

電子化されている植物図鑑や写真集には、植物の特性が自由形式で記述されている。ここで、例えば、「披針形の葉をつけ果実が楕円形で花が白い」植物を検索したい場合、単に「披針」や「楕円」や「白」というキーワードを与えるだけでは、それらが葉、花、果実のいずれについての特性かを明示できず、妥当な結果が得られない。そこで、これらの自由形式記述を係り受け解析器（『CaboCha/南瓜』『KNP』『GiNZA』『Natural Language API』）で解析して構造木を得て、そこから、コンピュータで検索しやすくするために JSON 形式を自動生成することを試みた。JSON のタグとしては、葉と花と果実のそれぞれについて、数個を設定し、対応するバリュー（値）は、構造木を一定の方式で辿りながらノード上のテキストを連結することで得ることができる。JSON テキストが自動生成されれば、JAVA プログラミングにおける（植物を表す）オブジェクトのストリームに変換して、マップやフィルタ処理を施すことにより、検索を効率的に実行することが可能になる。

以下において、主にデジタル植物写真集（渡辺坦 2020）を対象として検討した。この写真集には、多様な植物についての豊富な写真と詳細な説明が植物学の系統にそって整理され、多面的な検索が段階的にできるようになっている。なかでも、ここで着目したのは、掲載されている全植物について、それぞれ一行程度で簡潔に特性が記述された文である。これを JSON の自動生成の主な対象とした。JSON 自動生成結果とそれに基づく検索法を評価し、課題を明らかにする。

† yamamotof@ieee.org

２．自由形式から JSON 形式を自動生成する方式の概要

２.１　植物の特性の自由形式記述例

どのような自由形式の記述から JSON テキストを生成するのかを説明する。一例として、植物「ムニンノボタン」『ムニンノボタン』を取り上げる。これは、小笠原諸島父島にだけ生育する日本固有種で、平成 16 年国内希少野生動植物種に指定されている。その葉と花と果実に関する特徴を自由形式で記述したものを表１に示す。

表１　植物「ムニンノボタン」の特徴記述例（自由形式）

明瞭な３本の脈がある葉は両面にまばらな剛毛をつける。４枚又は５枚の花弁を持つ白い花が 7 月から 8 月頃に次々と咲く。小笠原が涼しくなる頃に直径１センチほどの球形の蒴果がつく。

２.２　自由形式記述に対する係り受け解析

次に、この記述を係り受け解析器 CaboCha で解析した結果を図１に示す。文節は番号 0～26 までの 27 個あり、最後の述語「26 つく。」を根（root）とする木構造になっている。また、３つの文の順序を反映して、述語「8 つける。」は述語「18 咲く。」に、さらに述語「18 咲く。」は述語「26 つく。」に係る。

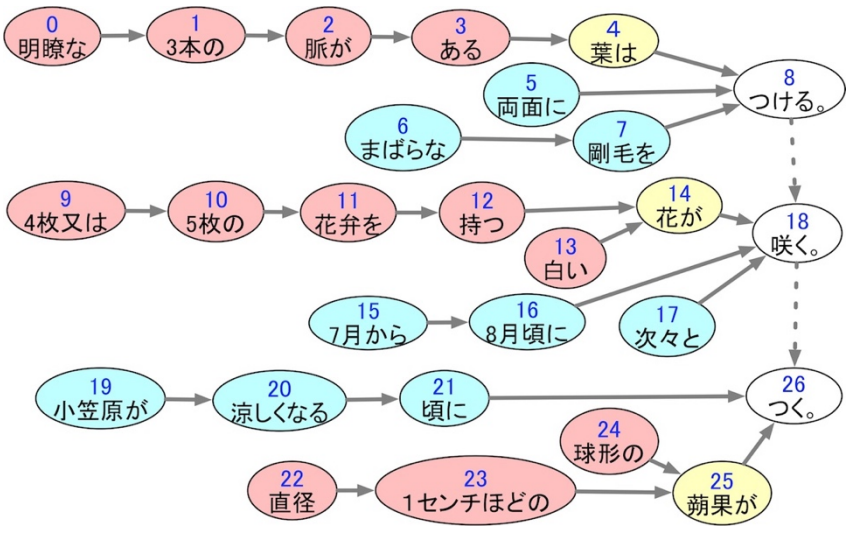


図１　植物「ムニンノボタン」の説明文の係り受け解析結果

２.３　JSON 形式テキストの自動生成方式

係り受け解析結果の木構造から、植物の特徴に関するコンピュータ処理をしやすくするため、JSON 形式テキストを生成する。その自動生成方式の概要は以下の通りである。（例外的な付加規則については省略している。）

(1) 葉と花と果について、それぞれ主語と述語になる文節を求める。図１において、例えば、葉については、「文節番号 4 と 8」である。ただし、例えば「葉は」や「葉が」等が出現せず、「葉の縁が波打つ」等の場合は、「葉」を含む文節を主語のように扱う。
(2) JSON のタグとして、葉と花と果のそれぞれについて、type（種類）、what（色、形、模様、大きさ、質感等）、how（配置、結合、出現状況、時期等）を設定する。
(3) タグ type の値は、主語の文節の自立語とする。

(4) タグ what の値は、主語の先祖（predecessors）の文節のテキストを文節番号順に連結したものとする。先祖のパスは複数あり得るので、原則としてその全てを連結する。
(5) タグ how の値は、述語の先祖（predecessors）の文節のテキストを文節番号順に連結したものとする。先祖のパスは複数あり得るので、原則としてその全てを連結し、最後に述語を付加する。ただし、原則として、主語や動詞の文節等は連結しない。

このようにして、タグとバリュー（値）を対応させた結果を表２に示す。カッコ内の数字は文節番号である。表のセルの色は、図１の文節のノードに付した色と対応している。このような表が得られれば、JSON テキストを生成することができる。

表２　JSON のタグとバリューの対応

	type	what	how
葉	(4)	(0)(1)(2)(3)	(5)(6)(7)(8)
花	(14)	(9)(10)(11)(12)(13)	(15)(16)(17)(18)
果	(25)	(22)(23)(24)	(19)(20)(21)(26)

表２から生成された JSON テキストを図２に示す。このようにして得られた JSON テキストを利用した多様な検索方法については、4.で検討する。


```
{
 "name": "ムニンノボタン",
 "desc": "<原文と同一なので省略>",
 "alias": "N",
 "url": "<表示省略>",
 "group": "null",
 "leaf": {
  "type": "葉",
  "what": "明瞭な3本の脈がある",
  "how": "両面にまばらな剛毛をつける"
 },
 "flower": {
  "type": "花",
  "what": "4枚又は5枚の花弁を持つ白い",
  "how": "7月から8月頃に次々と咲く"
 },
 "fruit": {
  "type": "蒴果",
  "what": "直径1センチほどの球形の",
  "how": "小笠原が涼しくなる頃につく"
 }
}
```

図２　「ムニンノボタン」に対して自動生成された JSON テキスト

３．デジタル植物写真集からの JSON テキストの自動生成

以上の方式を実用的なデジタル植物図鑑の説明文に適用する。対象としたのは、「1. はじめに」で述べたデジタル写真集（渡辺坦 2020）である。特に、その中に掲載されている、各植物の特性を簡潔に一行で記述した説明文に着目した。それを基に JSON テキストを生成すれば、植物の検索が容易になると考えたからである。詳細を説明する前に、まず、この全体イメージを表現した図３を以下に示す。

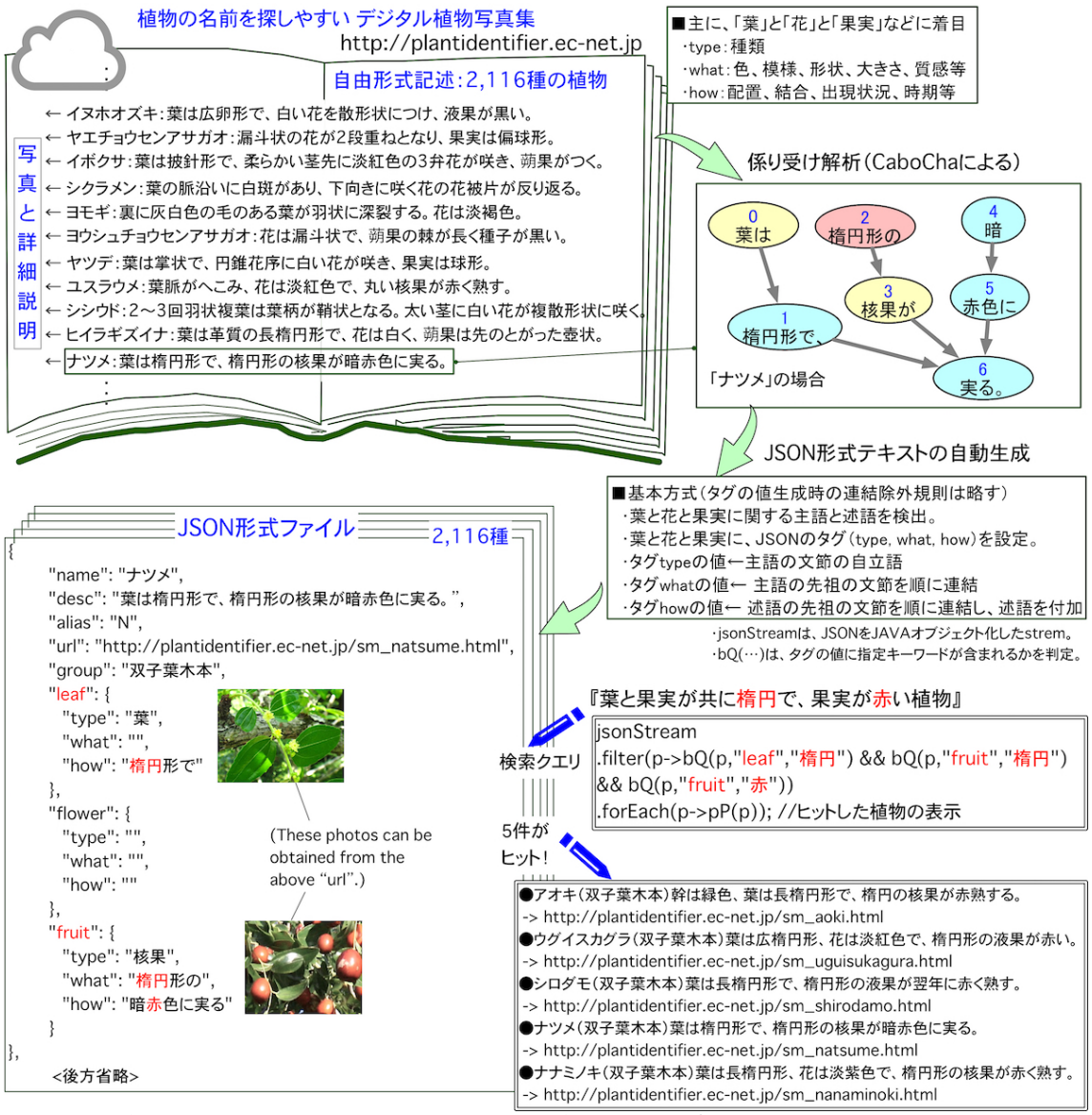


図３　自由形式記述からの JSON テキスト自動生成と多様な検索（全体イメージ）

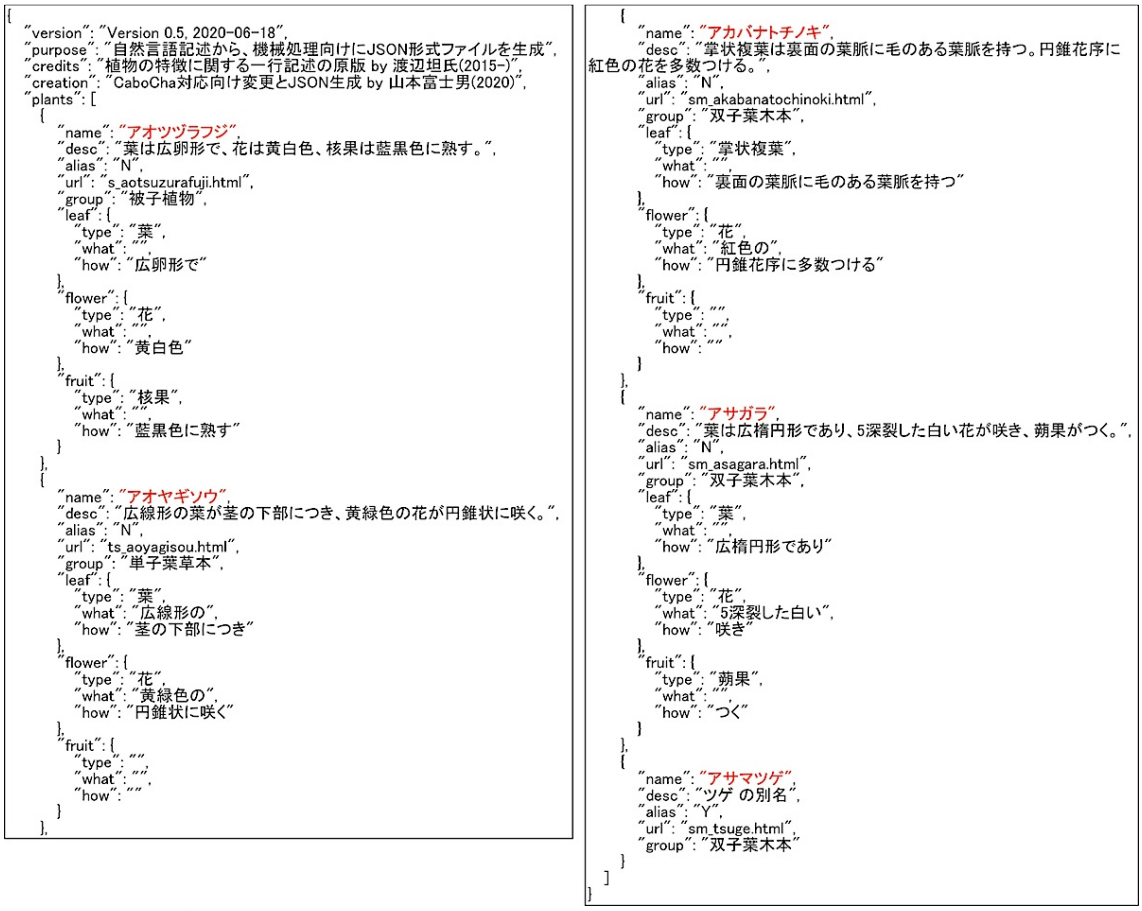
３.１　植物特性の簡潔な説明文

対象とした説明文は（2020 年 6 月現在）2,965 行ある。このうち、849 行は、ある植物の別名等であるので、実質 2,116 種の植物が記載されている。その中から５例を示す。簡単化のため、原文に含まれる一部の html タグを取り除く等の加工を施している。各植物について、表３に示すように、植物名、詳細説明用 html ファイル名、および、簡潔な説明文の３項目で構成される。最後の植物【5】は、「ツゲ」という植物の別名であることが示されている。Html ファイル名の先頭（アンダーバーの左側）の文字列は、植物の区分（双子葉草本、単子葉草本、双子葉木本、裸子植物、被子植物、シダ植物等）を表すようになっている。

表３　デジタル植物写真集の記述文の例

【1】アオツヅラフジ : s_aotsuzurafuji.html : 葉は広卵形で、花は黄白色、核果は藍黒色に熟す。
【2】アオヤギソウ : ts_aoyagisou.html : 広線形の葉が茎の下部につき、黄緑色の花が円錐状に咲く。

【3】アカバナトチノキ : sm_akabanatochinoki.html : 掌状複葉は裏面の葉脈に毛のある葉脈を持つ。円錐花序に紅色の花を多数つける。
【4】アサガラ : sm_asagara.html : 葉は広楕円形であり、5 深裂した白い花が咲き、蒴果がつく。
【5】アサマツゲ : sm_tsuge.html : ツゲ$$ の別名




```
{
  "version": "Version 0.5, 2020-06-18",
  "purpose": "自然言語記述から、機械処理向けにJSON形式ファイルを生成",
  "credits": "植物の特徴に関する一行記述の原版 by 渡辺垣氏(2015-)",
  "creation": "CaboCha対応向け変更とJSON生成 by 山本富士男(2020)",
  "plants": [
    {
      "name": "アオツヅラフジ",
      "desc": "葉は広卵形で、花は黄白色、核果は藍黒色に熟す。",
      "alias": "N",
      "url": "s_aotsuzurafuji.html",
      "group": "被子植物",
      "leaf": {
        "type": "葉",
        "what": "",
        "how": "広卵形で"
      },
      "flower": {
        "type": "花",
        "what": "",
        "how": "黄白色"
      },
      "fruit": {
        "type": "核果",
        "what": "",
        "how": "藍黒色に熟す"
      }
    },
    {
      "name": "アオヤギソウ",
      "desc": "広線形の葉が茎の下部につき、黄緑色の花が円錐状に咲く。",
      "alias": "N",
      "url": "ts_aoyagisou.html",
      "group": "単子葉草本",
      "leaf": {
        "type": "葉",
        "what": "広線形の",
        "how": "茎の下部につき"
      },
      "flower": {
        "type": "花",
        "what": "黄緑色の",
        "how": "円錐状に咲く"
      },
      "fruit": {
        "type": "",
        "what": "",
        "how": ""
      }
    },
    {
      "name": "アカバナトチノキ",
      "desc": "掌状複葉は裏面の葉脈に毛のある葉脈を持つ。円錐花序に紅色の花を多数つける。",
      "alias": "N",
      "url": "sm_akabanatochinoki.html",
      "group": "双子葉木本",
      "leaf": {
        "type": "掌状複葉",
        "what": "",
        "how": "裏面の葉脈に毛のある葉脈を持つ"
      },
      "flower": {
        "type": "花",
        "what": "紅色の",
        "how": "円錐花序に多数つける"
      },
      "fruit": {
        "type": "",
        "what": "",
        "how": ""
      }
    },
    {
      "name": "アサガラ",
      "desc": "葉は広楕円形であり、5深裂した白い花が咲き、蒴果がつく。",
      "alias": "N",
      "url": "sm_asagara.html",
      "group": "双子葉木本",
      "leaf": {
        "type": "葉",
        "what": "",
        "how": "広楕円形であり"
      },
      "flower": {
        "type": "花",
        "what": "5深裂した白い",
        "how": "咲き"
      },
      "fruit": {
        "type": "蒴果",
        "what": "",
        "how": "つく"
      }
    },
    {
      "name": "アサマツゲ",
      "desc": "ツゲ の別名",
      "alias": "Y",
      "url": "sm_tsuge.html",
      "group": "双子葉木本"
    }
  ]
}
```


図４　自動生成された JSON テキストファイル（表３の５つの植物に対して）

３.２　デジタル写真集の全植物に対する JSON の自動生成

このような、写真集の全ての植物の記述文（全 2,965 行）に対して、JSON テキストの自動生成を行った。その結果、約 21,000 行からなる JSON テキスト（pretty formatted では約 52,000 行）が得られた。全ての植物に対して、必ずしも葉と花と果のすべてに言及があるわけではないので、タグに対するバリュー（値）が空白になっている場合も多い。図４は、表３に掲載した５つの植物に対して生成された完全な JSON ファイルである。

４．JSON テキストを利用した多様な検索

本論文の主な目的は、冒頭で述べたように、自由記述文のままでは難しいと考えられる多様な検索を容易にすることにある。ここでは、上記のデジタル植物写真集に対して自動生成した JSON ファイルを使った検索を検討する。検索プログラムは JAVA で書く。JSON テキストを JAVA のオブジェクトとして扱うためのライブラリが利用できる。単一の JSON 形式、およびそれらの配列も扱える。その上で、JAVA のラムダ式とストリームを利用することで、効率の良い検索プログラムを簡潔に書くことができる。

一般に、JSON を入力とした検索アプリケーションを作成する場合、多くのプログラミン

グ言語を利用することができる。JAVA 以外に、例えば JavaScript や Python でも同様のことは（ラムダ式やストリームも利用できるので）可能と思われるが、本論では、形態素解析器 MeCab と係り受け解析器 CaboCha を共に JAVA アプリケーションの中で利用しているので、一貫性を持たせるため、ここでも JAVA を利用する。

以下において、jsonStream は、自動生成した JSON を JAVA オブジェクトに変換し、それをストリーム化したものである。また、bQ(...)、sQ(...)、pP(...)などは、JSON のタグに対するバリューの検索や出力表示の簡易化のためのユーザ設定メソッドである。検索で与えるキーワードの候補を下記【検索例３】の如く、事前に収集することも考えられる。

【検索例１】「ムニンノボタン（2. で述べたもの)」の特徴を与えた場合の検索

■検索クエリ

```
// 葉に 3 脈があり、花が白いことに着目した検索
jsonStream　//植物オブジェクトのストリーム
.filter(p -> bQ(p,"leaf","3") && bQ(p,"leaf","脈"))
.filter(p -> bQ(p,"flower","白"))
.forEach(p -> pP(p));　//ヒットした植物の表示
```

■検索結果　−>以下のように、正解が１件ヒットした

●ムニンノボタン（双子葉木本）葉は長楕円形で 3 脈明瞭で、白い花が咲き、丸い蒴果がつく。
-> http://plantidentifier.ec-net.jp/sm_muninnobotan.html

【検索例 2】葉の形状と花の色を与えた場合の検索

■検索クエリ

```
// 葉が"倒披針"を含み、花が"白"または"紫"の植物
jsonStream
.filter(p -> bQ(p, "leaf", "倒披針"))
.filter(p -> bQ(p, "flower", "白") || bQ(p, "flower", "紫"))
.forEach(p -> pP(p));
```

■検索結果　−>以下のように、２件ヒットした

●シライトソウ（単子葉草本）葉は倒披針形で、花序は白いブラシ状。
-> http://plantidentifier.ec-net.jp/ts_shiraitosou.html
●トウテイラン（双子葉草本）全体に綿毛があり、葉は倒披針形で、青紫色の花が総状に咲く。
-> http://plantidentifier.ec-net.jp/ss_touteiran.html

【検索例 3】果実の種類を重複無く列挙する（ただし、type タグのみを対象とする）

■検索クエリ

```
//このデジタル植物写真集の果実の種類（type)を重複無く列挙する
System.out.println(jsonStream
.map(p ->　sQ(p, "fruit","type"))
.filter(p ->　!p.equals("")).distinct()
.collect(Collectors.joining(", ", "[", "]")));
```

■検索結果　−>一部にバグが残っているが、以下のようにリスト形式で表示

[核果, 果実, 翼果, 梨状果, 液果, 胞果, 堅果, 蒴果, 朔果, 4 分果, 果実に, 豆果, 果穂, 卵形蒴果, 痩果, 分果, 小節果, 果胞, 長角果, 3 分果, イチジク状果, 球果, 袋果, ミカン果, 2 分果, 分離果, イチゴ果, 苺果, 集合果, 蒴果か, 梨果, バナナ状果, 果, 果皮, 総状液果, 瓜果, 4 分果, 球形液果, 果苞, 偽果, 蓋果, バナナ果, 穎果, 核果中]

【検索例 4】植物の種類（区分）毎に葉の特徴を纏める

このような多重のグループ化は下記のように一見複雑に見えるが、大半は出力表示のためのものである。JAVA の Collectors::groupingBy メソッドが威力を発揮している。以下において、Plants は、JSON から植物オブジェクトを再構成するためのユーザ定義クラスである。全体として、JSON を生成したことの効果が発揮される検索例となっている。

■検索クエリ

```
// 植物の種類（区分）毎に葉の特徴を纏める
Map<String, Map<String,List<Plants>>> myg =
jsonStream
.map(p -> new Plants(sQ(p,"group"),
"葉",sQ(p,"leaf","type"),sQ(p,"leaf","what"),sQ(p,"leaf","how"),
"花",sQ(p,"flower","type"),sQ(p,"flower","what"),sQ(p,"flower","how"),
"果",sQ(p,"fruit","type"),sQ(p,"fruit","what"),sQ(p,"fruit","how")))
.collect(Collectors.groupingBy(
Plants::getGroup,Collectors.groupingBy(Plants::getElem1)));
Set<String> groupKey = myg.keySet();
for (String g: groupKey){
System.out.print("¥n●" + g + ":" );
Set<String> elemKey = myg.get(g).keySet();
for (String e: elemKey){
System.out.print("¥n - " + e + ":" + "type ");
System.out.print(myg.get(g).get(e).stream().map(p -> p.getTypeLe())
.filter(p -> !p.equals("")).distinct().collect(Collectors.joining(",","【","】")));
System.out.print("¥n - " + e + ":" + "what ");
System.out.print(myg.get(g).get(e).stream().map(p -> p.getWhatLe())
.filter(p -> !p.equals("")).distinct().collect(Collectors.joining(",","【","】")));
}
}
```

■検索結果　−＞項目多数のため、一部のみ表示

```
●双子葉木本:
 - 葉:type 【葉,奇数羽状複葉,掌状複葉,小葉,葉腋,旧葉,若葉,3 出複葉,葉柄,羽状複葉,偶数羽状複葉,2 回偶数羽状複葉,葉裏,苞葉,2 回羽状複葉,葉脈,3 回羽状複葉】
 - 葉:what 【楕円状の,3 出複葉の,長楕円形の,掌状に裂けた,棘が大きく,被針形の,細長い,鱗片状の,卵形の,緑白色の丸い,奇数羽状複葉の,倒卵形の,奇数羽状複葉で棘のある,常緑樹の,線形の,披針形の細鋸歯の,掌状複葉の,革質の,卵形で束生する,大きな,丸い,白い 2 枚の,羽状複葉の,広卵形の,3 枚の,先が 3 浅裂した,有毛の】
●単子葉木本:
 - 葉:type 【葉,羽状複葉,葉状枝,葉鞘】
 - 葉:what 【多肉の剣形の,被針形の,掌状に深裂した,長い,剣状の,尖った板のような多肉の,線形の長い,葉のような,大きな】
●裸子植物:＜表示省略＞
●双子葉草本:＜表示省略＞
●被子植物:＜表示省略＞
●単子葉草本:＜表示省略＞
```

５．JSON テキスト自動生成上の問題点

５.１　係り受け解析器 CaboCha での問題点

今回の JSON 自動生成は、原文の係り受け解析結果を基にしている。係り受け解析器として、広く普及している CaboCha を利用した。しかし、上記に述べたデジタル植物写真集にある簡潔な一行記述には、多様な表現が含まれているため、期待される（人間が読んで自然と感じる）解析結果が得られない場合がいくつか見られた。その中には、句読点を補うことで解決できる軽微なものも多いが、原文を一部変更をせざるを得ない場合もあった。

それらの要点を以下に纏める。対象とした 2,116 件の説明文を係り受け解析し、JSON を自動生成した場合の状況は以下の通りであった。

(a) 原文のままで、正常に JSON 生成 -> 約 55%
(b) 原文に句読点を追加することで、正常に JSON 生成 -> 約 30%
(c) 原文の助詞等を置き換えることで、正常に JSON 生成 -> 約 9%
(d) 妥当な係り受け解析結果を得るために原文の一部を書き換え -> 約 6%

以上のうち、(b)は、原文に含まれている複文（主語と述語の組が複数）の区切りを、係り受け解析がうまく判断できなかったためと思われる。複文を単文に切り分けることは、(b)のように単純に句読点を追加すれば済む場合もあるが、一般にはそれでは済まない場合も多く知られている。また、(c)のケースは、人間が意図した意味が伝わらないという問題であるが、係り受け解析の前段で行われる、形態素解析で使用される辞書の問題も含まれるかも知れない。より踏み込んだ意味解析として、格フレーム辞書を用いた解析（杉本徹･岩下志乃 2018、山本和英 2018）が必要かも知れないが、本論は、その領域には言及していない。

今回、期待通りの係り受け解析結果が得られなかった例のうち 9 例を**付録**に示した。何が問題であるかを具体的に示し、それを回避するために変更した文例も示してある。また、CaboCha に加えて、KNP と GiNZA による解析結果も簡単に比較している。

５.２　今後の課題

(1) 係り受け解析結果の木構造をたどって、植物の what（色や形や大きさ）、how（配置や結合や出現時期等）を合成する際に、句読点を含む文節には例外的処置を施している。しかし、場合によっては、不都合な結果が生ずるので、その条件の精密化を要する。
(2) 例えば、「葉」について、明確に「葉は」や「葉が」という主語が無く、いきなり「鋸葉を持ち…」などがある場合、「鋸葉」を含む文節を主語のように扱っているが、必ずしも妥当な結果が得られないことがある。ここでも、条件の精密化を要する。
(3) 今回は、葉と花と果についてだけ着目したが、「根」や「茎」や「穂」に関する記述もあるので、今後それらに関する JSON タグも用意して JSON テキストを生成する。
(4) 検索プログラムでは、部分一致（contains）のみを利用している。表層形だけでなく基本形を含む自動検索の方式も検討したい。
(5) 検索プログラムにおいては、より使いやすいユーザインタフェース（GUI）が必要である。また、検索の利便性を高める、スマートフォンアプリケーションの開発も望ましい。

６．おわりに

植物の特徴が自由形式で記述されている文書から、自動的に JSON 形式テキストを生成して、それをもとに多様な検索を行うためのプロトタイプを作成した。今後、いくつかの同類の電子図鑑や山岳一覧案内の文書などにも適用して、課題を探りたい。また、基本となる係り受け解析が失敗している場合、その状況を素早く把握できる仕組みも得たい。さらに、迅速に別の係り受け解析器に切り替える方策も（解析器ごとに出力形式が異なるので）必要かも知れない。

謝　辞

渡邊坦電気通信大学名誉教授には、氏のデジタル植物写真集の本研究での利用をご快諾いただき、その設計理念と web ページの階層や構成についても詳細にご教示戴いた。また、安岡孝一京都大学教授と田中博神奈川工科大学教授からは、本研究の基になった筆者のブログの連載記事に対して有益なコメントおよび激励を戴いた。これらの方々に感謝申し上げる。

参考文献

渡辺坦(2020). 『植物の名前を探しやすいデジタル植物写真集』
http://plantidentifier.ec-net.jp
杉本徹･岩下志乃(2018). 『Java で学ぶ自然言語処理と機械学習』オーム社, pp.179-183.
山本和英(2018). 『箱庭言語処理のための格フレーム辞書構築の意義』
https://www.japio.or.jp/00yearbook/files/2018book/18_5_03.pdf

関連 URL

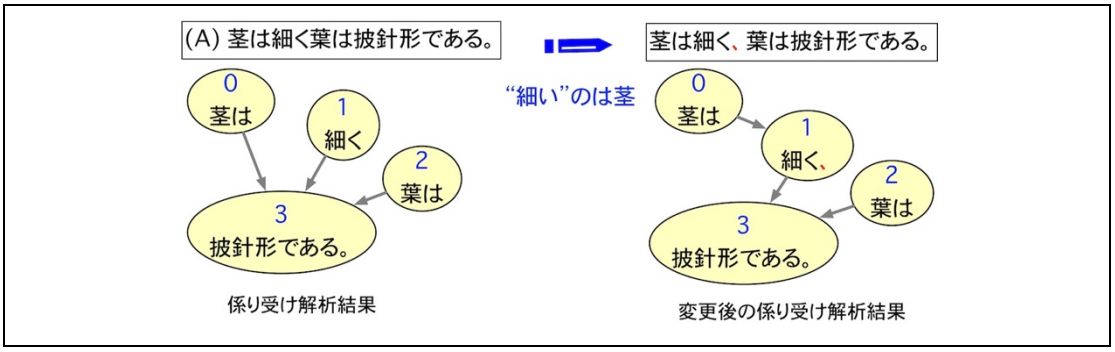
Yet Another Japanese Dependency Structure Analyzer『CaboCha/南瓜』
https://taku910.github.io/cabocha/
黒橋･村脇研究室　日本語構文･格･照応解析システム『KNP』
http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP
Japanese NLP Library『GiNZA』https://megagonlabs.github.io/ginza/
Google『Natural Language API』https://cloud.google.com/natural-language/
環境省　自然環境･生物多様性『ムニンノボタン』
https://www.env.go.jp/nature/kisho/hogozoushoku/muninnobotan.html

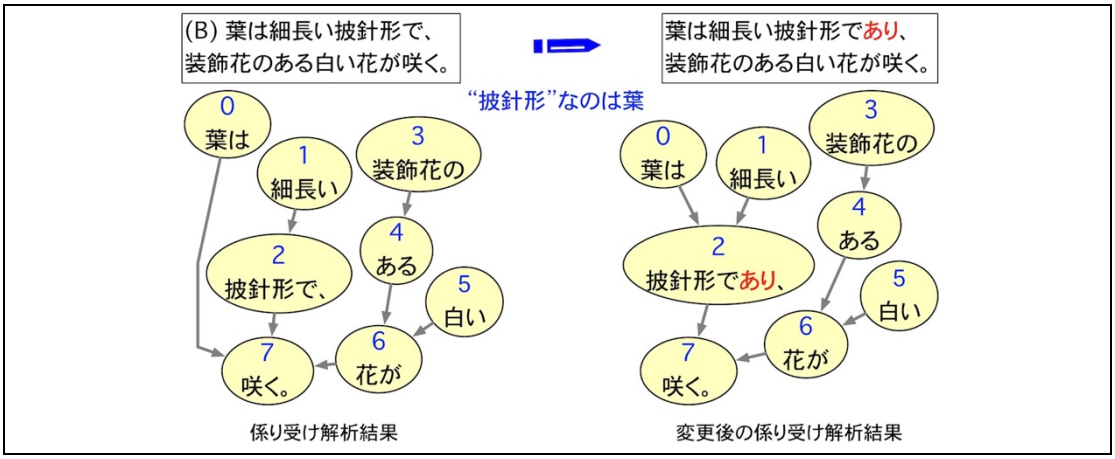
付　録

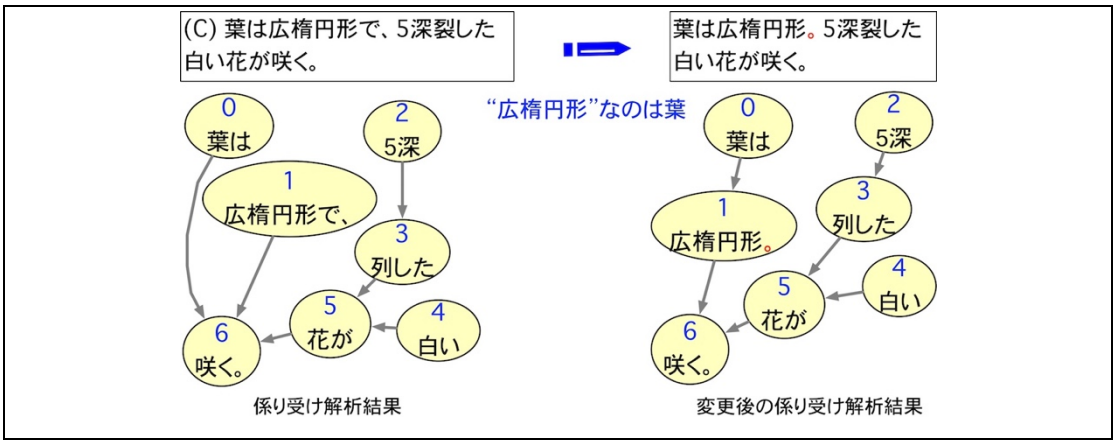
【期待通りの係り受け結果が得られない場合の検討】

以下の９例は、（渡辺坦 2020）から引用したが、問題点を明確にするため、原文の一部省略している。いずれも、人間にとっては自然に理解できる表現だが、何らかの理由で期待通りの係り受け解析結果とならなかったケースである。どのように変更して、係り受け解析をやり直したかも示した。

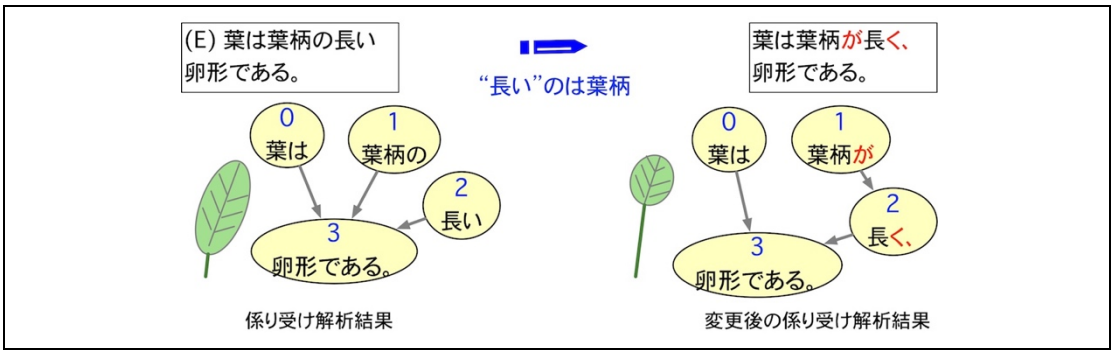
●複文に含まれる単文の区切りの認識に問題か？

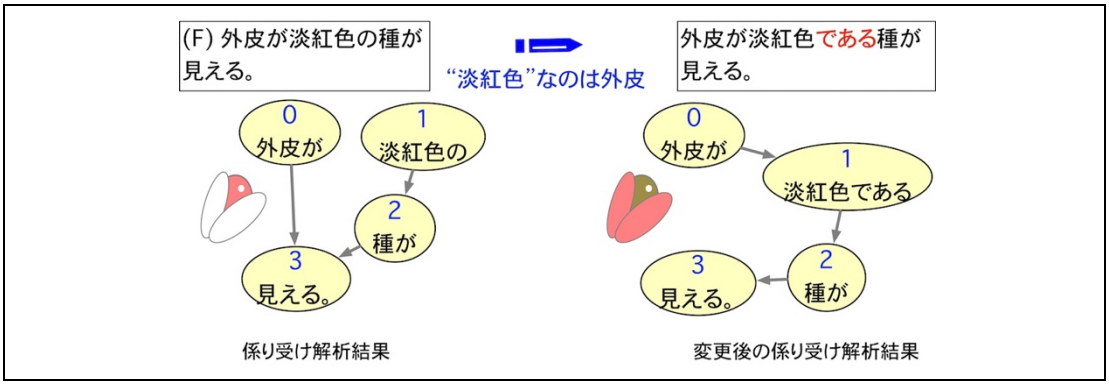


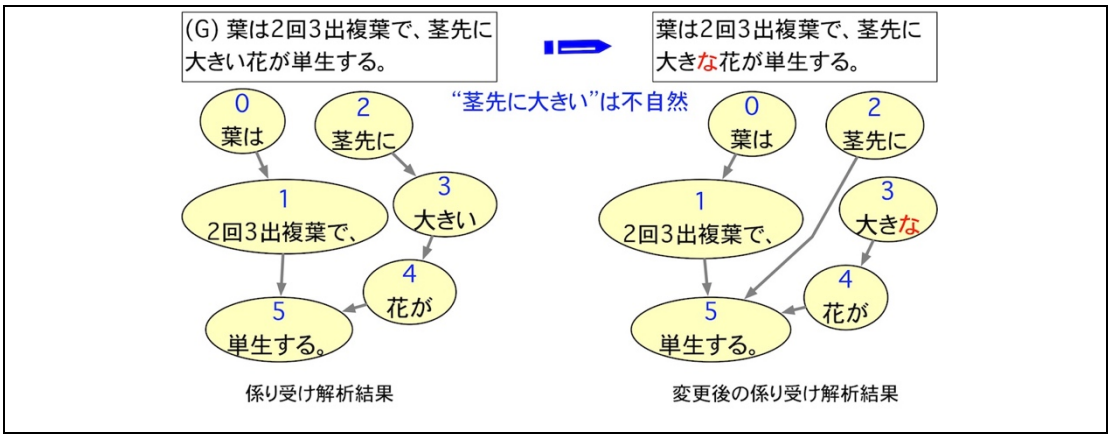


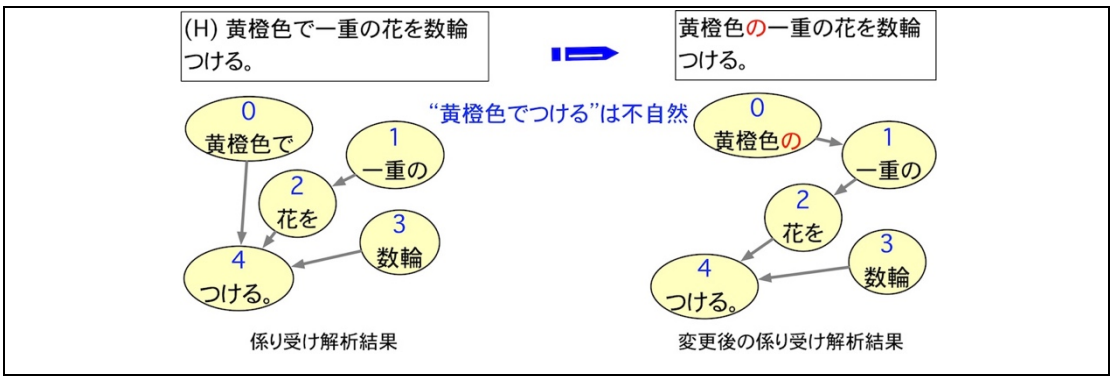


●何らかの理由により、不自然な修飾となる問題

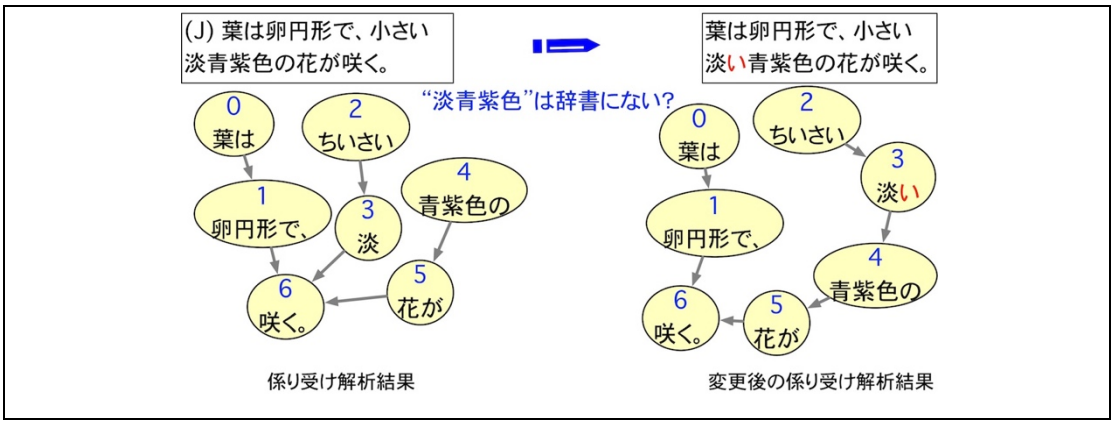


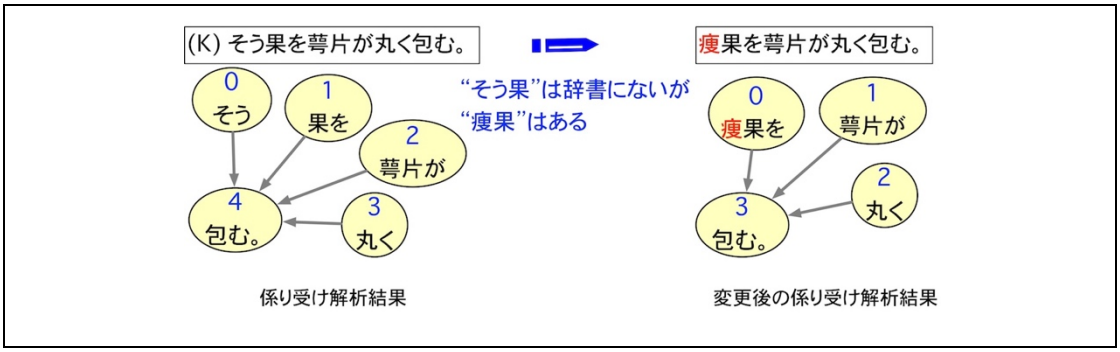






●文節を生成するための辞書に載っていない用語



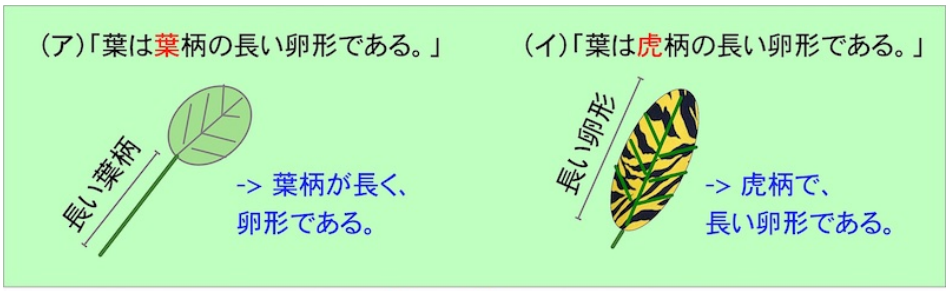


【CaboCha 以外の係り受け解析器の利用の検討】

ここでは、以下の２つの例文（ア）と（イ）について、３つの解析器（CaboCha、KNP、GiNZA）の解析結果を比較する。これだけでは十分な検証はできないので、３つの解析器の比較評価を意図したものではなく、ひとつの参考記録として残す。

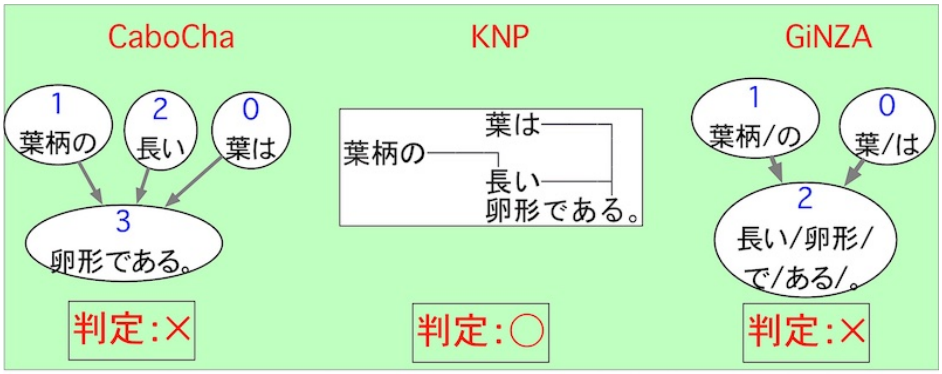
●例文（ア）：葉は葉柄の長い卵形である。

●例文（イ）：葉は虎柄の長い卵形である。



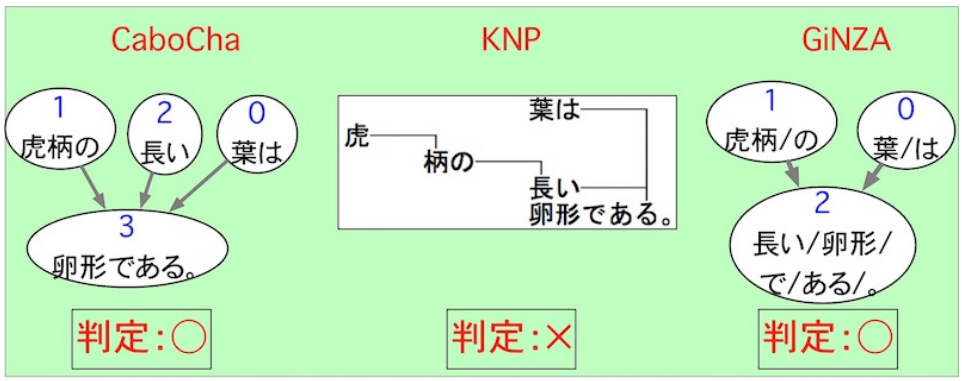
二つの例文(”葉”か”虎”かの違いがあるだけ。)

例文（ア）に関しては、KNP は、「葉柄の長い」という係り受け関係をきちんと捕らえているが、CaboCha と GiNZA はいずれも、「長い卵形」と判断しており、妥当ではない。



例文(ア)「葉は葉柄の長い卵形である。」に対する係り受け解析結果

一方、例文（イ）は例文（ア）に出現する"葉"が"虎"に変わっただけである。解析結果結果は少し想定外であった。（ア）の場合の判定結果（○×）が逆転した。すなわち、KNP は、「虎柄の長い」という不自然な係り受け関係を採用してしまった。



例文(イ)「葉は虎柄の長い卵形である。」に対する係り受け解析結果