自然言語処理入門

岸山 健 (31-187002)

Dec. 3, 2018

課題

「ヒロシが病院でもらった薬を飲んだ」を CYK 法で解析せよ.

1 下準備

1.1 文法

CYK 法のような “上昇型文法解析” を実行するためには 1. 左辺と右辺から成る文法 と， 2. 右辺から左辺を取得する関数の 2 つがまず必要となる． この節でははじめに，与えられた文法を 1. 句構造規則 `rule.p` と 2. 辞書規則 `rule.d` に分け，Char 型で定義する.

```
# 句構造 (phrase) 規則
rule.p <- c("S -> PP VP",  "PP -> NP P",
            "NP -> VP NP", "VP -> PP VP")
# 辞書 (dictionary) 規則
rule.d <- c("NP -> 薬", "NP -> 病院", "NP -> ヒロシ",
            "VP -> 飲んだ", "VP -> もらった",
            "P -> が", "P -> で", "P -> を")
```

次に，上の各規則に含まれる `->`で左辺と右辺を分割し Char 型の vector を生成する関数 `char2cfg` も定義する． 仮に “S -> PP VP” という文字列を与えた場合，左辺として “S”，右辺として “PP VP” が返される.

```
char2cfg <- function(s) unlist(strsplit(s, " -> "))
char2cfg("S -> PP VP")
# [1] "S"     "PP VP"
```

上で行なった規則の定義には `c` 関数を使っているが，`c` 関数の “c” は “concatenate” を意味し，ベクトルの各要素が同型であることを保証できる．先ほど定義した `char2cfg` は Char 型をベクトルに変換する関数だった．`lapply` 関数を使えば Char 型のベクトルである `rule.p` や `rule.d` の各要素に対し，`char2cfg` を apply できる．したがって，`rule.p` の各要素に `char2cfg` を `lappply` すると以下のような結果となる.

```
lapply(rule.p, char2cfg)
# [[1]]
```

```
# [1] "S"     "PP VP"
#
# [[2]]
# [1] "PP"   "NP P"
#
# [[3]]
# [1] "NP"    "VP NP"
#
# [[4]]
# [1] "VP"    "PP VP"
```

CYK 解析は上昇型の解析であるため，‘PP’ と’VP’ を見たら’S’ を返し，’ 太郎’ を見たら ‘NP’ を返してくれる機能が必要となる．先ほどの char2cfg を規則に lapply すると，各リストの要素には LHS と RHS が格納された．このリストをまず data.frame 型に変えて次に t 関数で転置し，(転置すると型が matrix になってしまうので)as.data.frame で型を戻し，最後に列に LHS と RHS の名前をつける．以上の作業を句構造規則 rule.p 対して行なうと table.p という data.frame が作れる．

```
vector.p <- lapply(rule.p, char2cfg)
as.data.frame(vector.p, stringsAsFactors=FALSE)
#   c..S....PP.VP.. c..PP....NP.P.. c..NP....VP.NP.. c..VP....PP.VP..
# 1               S              PP               NP               VP
# 2           PP VP            NP P            VP NP            PP VP
t(as.data.frame(vector.p, stringsAsFactors=FALSE))
#                   [,1] [,2]
# c..S....PP.VP..  "S"  "PP VP"
# c..PP....NP.P..  "PP" "NP P"
# c..NP....VP.NP.. "NP" "VP NP"
# c..VP....PP.VP.. "VP" "PP VP"
table <- as.data.frame(t(as.data.frame(l, stringsAsFactors=FALSE)),
                  stringsAsFactors=FALSE)
# 上の作業 + 列名変更すると目的達成
vector2table <- function(l) {
    table <- as.data.frame(t(as.data.frame(l, stringsAsFactors=FALSE)),
                      stringsAsFactors=FALSE)
    colnames(table) <-  c("LHS","RHS")
    return(table)}
table.p <- vector2table(vector.p)
#                   LHS   RHS
# c..S....PP.VP..    S PP VP
# c..PP....NP.P..   PP  NP P
# c..NP....VP.NP..  NP VP NP
# c..VP....PP.VP..  VP PP VP
```

上の様なテーブルがあれば，RHS から LHS の取得は容易となる．例えば，仮に “PP VP” を RHS にもつ規則が欲しければ，RHS 列が “PP VP” のデータフレームを subset として取り，そのデータフレームの LHS 行

をベクトルとして取得すれば良い．そのような関数を以下に rhs2lhs と定義する．まず参照するテーブルを table として引数にとり以下の関数を返す．RHS を Char 型として引数に取り，先に部分適用した table の LHS を取得できる．仮に table.p を与えてから “PP VP” を与えた場合，“S” と “VP” の 2 つがベクトルとして返される．

```
rhs2lhs <- function(table) function(rhs){
    unlist(subset(table, table$RHS==rhs)$LHS)}
rhs2lhs(table.p)("PP VP")
# [1] "S"  "VP"
```

以上の操作を lexicon でも行ない，vector.d と table.d を作成する．これらにより，単語から前終端器号へのマップが可能になる．半角スペースので与えられた Char 型を vector に変換する関数 char2vector を考えた時， 問題となっている “ヒロシ が 病院 でもらった 薬 を 飲んだ” は Char 型のベクトルに変換できる．そのベクトルを先ほど作った rhs2lhs に辞書 (table.d) を部分適用して作った関数に sapply すると，CYK に必要な対角が得られる．

```
vector.d <- lapply(rule.d, char2cfg)
table.d <- vector2table(vector.d)
# [1] "NP"
char2vector <- function(s) unlist(strsplit(s, "[ ]"))
input <- "ヒロシ が 病院 で もらった 薬 を 飲んだ"
vector.input <- char2vector(input)
vector.input
# ヒロシ　が　病院　で　もらった　薬　を　飲んだ
class(rhs2lhs(table.d))
# [1] "function"
sapply(vector.input, rhs2lhs(table.d))
# ヒロシ　が　病院　で　もらった　薬　を　飲んだ
#   "NP"  "P"   "NP"   "P"      "VP"  "NP"  "P"    "VP"
```

1.2　三角行列

CKY 法は三角行列の a_{ij} をベースに構文解析を進めていくため，最後の準備として三角行列 (対角線で区切られている行列) を用意する．行列の i も j も長さは入力文字列がもつ前終端器号数となる．スペース区切りで形態素解析が終わっている文字列が与えられるとする．それを char 型のベクトルとして今後扱う．

```
char2vector <- function(s) unlist(strsplit(s, "[ ]"))
input <- "ヒロシ が 病院 で もらった 薬 を 飲んだ"
v.i <- char2vector(input)
# [1] "ヒロシ" "が" "病院"   "で"  "もらった" "薬"    "を"
# [8] "飲んだ"
length(v.i)
# [1] 8
```

ただ，行列内に更にベクトルが生まれることを考えると，行列を多次元に拡張した配列の使用が好ましい．まずは rhs2lhs を終端器号 (単語) に apply して前終端器号の list，v2l とする．つまり，“ヒロシ” を rhs2lhs

に apply すると NP となる．これを行列に組み込む操作を一般化すると，文字列の i 番目の要素を行列の i 行目，i 列目に挿入，という操作になる．

```
v2l <- lapply(v.i, rhs2lhs(table.d))
# 行列の中がベクトルとなりうるので配列を使う.
triangle <- array(dim = c(len, len, 1))
mapply((function(x,i) triangle[i,i,1] <<- x),
       v2l, 1:(length(v2l)))
triangle

# , , 1
#
#      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
# [1,] "NP" NA   NA   NA   NA   NA   NA   NA
# [2,] NA   "P"  NA   NA   NA   NA   NA   NA
# [3,] NA   NA   "NP" NA   NA   NA   NA   NA
# [4,] NA   NA   NA   "P"  NA   NA   NA   NA
# [5,] NA   NA   NA   NA   "VP" NA   NA   NA
# [6,] NA   NA   NA   NA   NA   "NP" NA   NA
# [7,] NA   NA   NA   NA   NA   NA   "P"  NA
# [8,] NA   NA   NA   NA   NA   NA   NA   "VP"

triangle[1,1,]
# [1] "NP"
```

ミソは単純な i と j による 参照ができない，という点．とりあえず，土台を d と置いて話を進める．n-d では上の階層を見ている．

外部パッケージには rbind を配列にも拡張した abind のみを利用する．

```
install.packages("abind")
library(abind)

triangle <- array(dim = c(len, len, 1))
mapply((function(x,i) triangle[i,i,1] <<- x),
       v2l, 1:(length(v2l)))
triangle

c(outer(rhs1, rhs2, FUN=paste))

triangle

n <- 8
for(d in 1:(n-1)){
    for (i in 1:(n-d)){
        j <- i + d
```

```
        for (k in i:(j-1)){
            ik   <- triangle[ i , k, ]
            k1.j <- triangle[k+1, j, ]
            product <- c(outer(ik, k1.j, FUN=paste))
            m <- unlist(sapply(product, rhs2lhs(table.p)))
            if(length(m)==1){
                triangle[i,j,1] <- m
            }else if(length(m)>1){
                len.m <- length(m)
                tmp <- array(dim = c(n, n, len.m))
                tmp[i,j,1:len.m] <- m
                triangle <- abind(triangle,tmp)
            }
        }
    }
}

cat.table <- array(dim = c(len, len))
for(i in 1:8){
    for(j in 1:8){
    cat.table[i,j]<- toString(na.omit(triangle[i,j,]))
    }
}
cat.table
```

一番左上のレベルでは “S” が 3 つ作成されているため，3 つの曖昧性があると分かる． まず 「ヒロシが病院でもらった薬を (PP)」「飲んだ (VP)」 がある． 「病院で」という PP と 「もらったという」VP がマージして「病院でもらった」という VP となり，さらに「ヒロシが」という PP と 「病院でもらった」という VP がマージして「ヒロシが病院でもらった」という VP が作られる．この VP と「薬」がマージし NP，そしてさらに 「を」とマージして PP となる．これは「ヒロシが病院でもらってきた薬」を誰かは知らないけど誰かが飲んだ，という意味になる．

こちらは曖昧性が生じていないようである． しかし 「ヒロシが (PP)」「病院でもらった薬を飲んだ (VP)」のほうは VP が 2 つできてる．VP に統合できる右辺は PP と VP であることを念頭に 探すと，「病院でもらった薬を (3–7)」「飲んだ (8)」 と「病院で (3–4)」「もらった薬を飲んだ (5–8)」 のパターンがある．