ĐẠI HỌC CÔNG NGHỆ
ĐẠI HỌC QUỐC GIA HÀ NỘI

BÁO CÁO BÀI TẬP LẬP TRÌNH

BỘ PHÂN TÍCH CÚ PHÁP $H_U syntactic$

Nguyễn Minh Hoàng

K53–CLC

Hà Nội, Ngày 21 tháng 12 năm 2011

Mục lục

Tóm tắt nội dung

Bộ phân tích cú pháp *HUsyntactic* được xây dựng sử dụng hai thuật toán phân tích cây cú pháp phổ biến: CKY và Earley.

Báo cáo này sẽ trình bày một số vấn đề về mã nguồn: công cụ sử dụng, tổ chức dữ liệu, cấu trúc dữ liệu cần thiết và hướng dẫn sử dụng; và một vài đánh giá trong quá trình xây dựng bộ phân tích cú pháp.

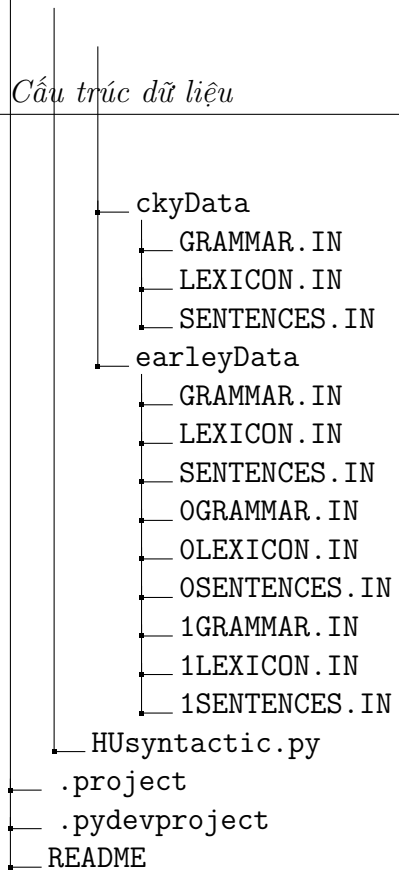
Phần I
Mã nguồn

1 Công cụ

- Language: Python 2.7 (64 bits)
- IDE: Eclipse + PyDev
- Platform: Window 7 Professional 64 bits/Ubuntu 11.04 64bits

2 Tổ chức dữ liệu

```
HUsyntactic
├── .git
├── .settings
├── docs
│   └── <các tệp tài liệu mã nguồn dưới dạng html>
└── src
    ├── cky
    │   ├── algorithm
    │   │   └── ckyAlgorithm.py
    │   ├── datastructure
    │   │   ├── dsCell.py
    │   │   ├── dsGrammar.py
    │   │   ├── dsLexicon.py
    │   │   └── dsSentence.py
    │   ├── unittest
    │   │   ├── testParseLexicon.py
    │   │   └── testReadFiles.py
    │   └── utils
    │       └── ckyIO.py
    ├── earley
    │   ├── algorithm
    │   │   └── earleyAlgorithm.py
    │   ├── datastructure
    │   │   ├── dsRule.py
    │   │   └── dsState.py
    │   └── utils
    │       └── earleyIO.py
    └── data
        └── ambiguousData
            ├── GRAMMAR.IN
            ├── LEXICON.IN
            └── SENTENCES.IN
```

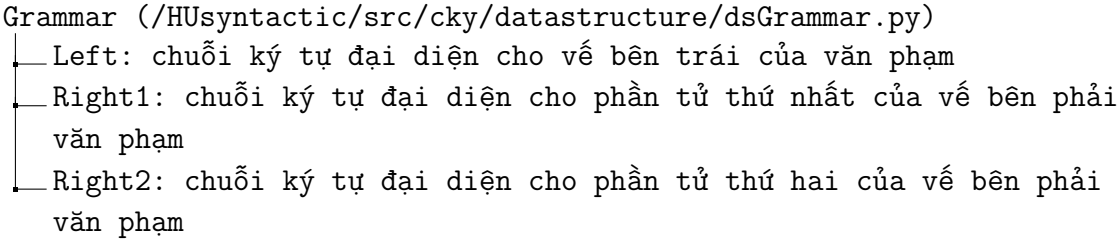


3 Cấu trúc dữ liệu

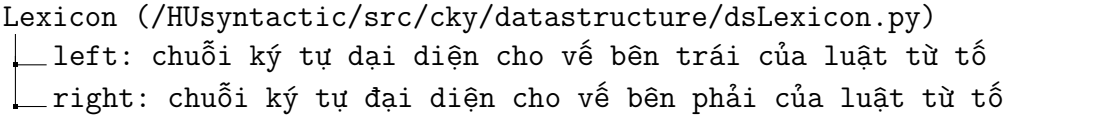
Mỗi thuật toán có những đặc thù riêng nên cấu trúc dữ liệu có sự khác nhau. Trong bộ phân tích này, các thuật toán sẽ sử dụng cấu trúc dữ liệu riêng.

3.1 Thuật toán CKY

Thuật toán CKY sử dụng 4 cấu trúc dữ liệu cơ bản: Grammar, Lexicon, Sentence, Cell.



Grammar là cấu trúc dữ liệu lưu trữ các văn phạm. Ví dụ với văn phạm S -> NP VP, *Left = S*, *Right1=NP*, *Right2=VP*.



Tương tự như *Grammar*, *Lexicon* lưu trữ các luật từ tố theo hai phần trái và phải.

```
Sentence (/HUsyntactic/src/cky/datastructure/dsLexicon.py)
└─ list of words: danh sách chứa các chuỗi ký tự đại diện cho từng từ
   trong câu
```

Sentence lưu trữ một câu trong tệp chứa câu đầu vào.

```
Cell (/HUsyntactic/src/cky/datastructure/dsCell.py)
└─ list in cell: danh sách chứa các phần tử theo cấu trúc từ điển key:value
   ├─ key: một kiểu 4-tuple, mục đích lần vết để xây dựng cây cú pháp
   │  ├─ phần tử 1: một kiểu 2-tuple, chứa chỉ số (i,j) của ô ngang
   │  ├─ phần tử 2: một kiểu 2-tuple, chứa chỉ số (i,j) của ô dọc
   │  ├─ phần tử 3: một số nguyên, vị trí của văn phạm trong danh sách
   │  │  thuộc ô ngang
   │  └─ phần tử 4: một số nguyên, vị trí của văn phạm trong danh sách
   │     thuộc ô dọc
   └─ value: một văn phạm Grammar
```

Cell thể hiện một ô trong bảng thực hiện thuật toán. Mỗi ô sử dụng một danh sách để lưu trữ các văn phạm hoặc luật từ tố xuất hiện trong ô theo quá trình thực hiện thuật toán. Văn phạm trong danh sách này sẽ đi cùng với các chỉ số để tiện cho việc lần vết xây dựng cây cú pháp, được biểu diễn theo kiểu từ điển (cung cấp bởi Python) ở dạng {key:value}. *Key* là một 4-tuple chứa 4 chỉ số của hai văn phạm trước đó tạo nên văn phạm trong ô hiện tại: vị trí (i,j) của ô nằm ngang, vị trí (i,j) của ô nằm dọc, vị trí của văn phạm trong danh sách của ô nằm ngang, vị trí của văn phạm trong danh sách của ô nằm dọc. *Value* là văn phạm. Luật từ tố cũng được lưu trữ trong *Cell* như văn phạm, nhưng có sự khác biệt đôi chút ở những chỉ số. Luật từ tố nằm trên đường chéo của bảng, được lấy ra từ danh sách luật từ tố nên không được suy diễn từ bất kỳ ô nào trong bảng, vậy nên chỉ số ô ngang và chỉ số ô dọc được đặt bằng chính vị trí của ô chứa luật từ tố đó, 2 chỉ số cuối cùng được đặt bằng *None*.

3.2 Thuật toán Earley

Hai cấu trúc dữ liệu `Rule` và `State` sử dụng để cài đặt thuật toán được mô hình ở dưới đây.

```
Rule (HUsyntactic/src/earley/datastructure/dsRule.py)
├─ Left part: chuỗi ký tự đại diện cho vế bên trái của văn phạm hoặc
│  luật từ tố
├─ Right part 1: danh sách phần tử đứng trước dấu •
└─ Right part 2: danh sách phần tử đứng sau dấu •
```

Cấu trúc dữ liệu `Rule` dùng để lưu trữ văn phạm và luật từ tố. Ví dụ với văn phạm *S -> NP • VP*, *Left part*= S, *Right part 1* = [NP], *Right part 2* = [VP]. Tại thời điểm ban đầu, khi văn phạm và luật từ tố đọc từ tệp đầu vào, toàn bộ vế phải được xem như đứng sau dấu •.

```
State(HUsyntactic/src/earley/datastructure/dsState.py)
    rule : văn phạm hoặc luật từ tố tại trạng thái hiện thời
    position: danh sách 2 phần tử số nguyên, các vị trí trong câu phân
    tích
    backpointer: danh sách chứa các trạng thái trước đó sinh ra trạng
    thái hiện tại
```

Cấu trúc dữ liệu `State` lữu trữ mỗi trạng thái của văn phạm (hoặc luật từ tố) trong quá trình biến đổi để tìm ra cây cú pháp. *position* là một danh sách 2 số nguyên biểu thị quãng vị trí của các từ trong câu đã được phân tích. *backpointer* cũng là một danh sách, nhưng lưu trữ các trạng thái cần đi qua để đạt tới trạng thái hiện tại.

4 Hướng dẫn sử dụng

Bộ phân tích cú pháp này sử dụng giao diện dòng lệnh để thực thi.

$ *pwd*
HUsyntactic/src
$ *python HUsyntactic.py -a [c/e] -g grammarFile -l lexiconFile -s sentenceFile -o outFile*

Tham số:

- -a: loại thuật toán phân tích
  - c: thuật toán cky
  - e: thuật toán earley
- -g: đường dẫn tới tệp chứa văn phạm
- -l: đường dẫn tới tệp chứa luật từ tố
- -s: đường dẫn tới tệp chứa câu
- -o: đường dẫn tới tệp đầu ra
- -h: xem hướng dẫn. Ví dụ, *python HUsyntactic.py -h*.

Ngoài ra, để hiểu rõ hơn về mã nguồn, chúng ta có thể xem thêm các tệp tài liệu trong thư mục *HUsyntactic/docs*.

5 Thực nghiệm

Dữ liệu thực nghiệm cho 2 thuật toán được lưu trữ trong *HUsyntactic/src/data*

5.1 Thuật toán CKY

Thuật toán CKY sử dụng 2 bộ dữ liệu dưới dạng chuẩn Chomsky:

- *HUsyntactic/src/data/ckyData*: dữ liệu mẫu đã cho.
- *HUsyntactic/src/data/ambiguousData*: dữ liệu để phân tích 3 câu nhập nhằng.

$ pwd
HUsyntactic/src
$ python HUsyntactic.py -a c -g data/ckyData/GRAMMAR.IN -l data/ckyData/
LEXICON.IN -s data/ckyData/SENTENCES.IN -o data/ckyData/OUTFILE.OUT

5.2 Thuật toán Earley

Thuật toán Earley thực nghiệm với 3 bộ dữ liệu dưới dạng văn phạm phi ngữ cảnh CFG được lưu trữ trong *HUsyntactic/src/data/earleyData*.

$ pwd
HUsyntactic/src
$ python HUsyntactic.py -a e -g data/earleyData/GRAMMAR.IN -l data/earleyData/
LEXICON.IN -s data/earleyData/SENTENCES.IN -o data/earleyData/OUTFILE.OUT

Chú ý:
Thuật toán Earley được cài đặt sử dụng hai tập ký hiệu sau:

- Tập ký hiệu không kết thúc: nonSet[1] [2] = {'NP', 'VP', 'PP', 'ADV', 'ADJ','N', 'Nominal', 'V', 'DET', 'Det', 'P', 'ROOT'}
- Tập ký hiệu nhãn từ loại: POS [3] [4]= {'N', 'V', 'P', 'DET', 'Det', 'ADV', 'ADJ'}

Quy ước rằng vế bên trái của luật từ tố chỉ chứa kí hiệu thuộc tập *POS*. Để thuật toán có thể hoạt động tốt, văn phạm chỉ nên chứa các từ trong tập *nonSet*.

[1]Dòng 25, HUsyntactic/src/earley/datastructure/dsRule.py
[2]Dòng 24, HUsyntactic/src/earley/algorithm/earleyAlgorithm.py
[3]Dòng 28, HUsyntactic/src/earley/datastructure/dsRule.py
[4]Dòng 29, HUsyntactic/src/earley/algorithm/earleyAlgorithm.py

Phần II
Đánh giá

6 Độ phức tạp

6.1 Thời gian

Thuật toán CKY sử dụng 3 vòng lặp chính. Trong khi đó, thuật toán Earley có 2 vòng lặp chính, và một vòng lặp ở quá trình *predict* hoặc *complete*. Như vậy, cả hai thuật toán CKY và Earley đều có độ phức tạp về mặt thời gian là $\mathcal{O}(n^3)$.

6.2 Không gian

Môi trường thực hiện của thuật toán CKY ở dạng bảng, với độ dài của câu đầu vào là n thì chỉ cần sử dụng tới nửa trên của bảng cỡ $n \times n$. Nghĩa là số ô cần thiết bằng $n^2/2$. Suy ra, độ phức tạp không gian của thuật toán CKY bằng $\mathcal{O}(n^2)$.

Thuật toán Earley sử dụng $n+1$ cho câu đầu vào có độ dài n. Một cột là một danh sách lưu trữ trạng thái của các văn phạm hay luật từ tố trong quá trình biến đổi để đưa dấu $\bullet$ về cuối. Gọi r là độ dài tối đa vế bên phải của một văn phạm (hoặc luật từ tố), g là tổng số văn phạm, l là tổng số luật từ tố. Mỗi cột chứa không quá $m = (r+1) \times (g+l+1)$. Bởi tính không thay đổi của văn phạm và luật từ tố nên m là một hằng số. Như vậy, có thể xem thuật toán Earley thực hiện trên một bảng có kích thước $n \times m$. Do đó, độ phức tạp không gian của thuật toán Earley là $\mathcal{O}(n \times m)$.

7 Tính nhập nhằng

Trong cả hai thuật toán, tính nhập nhằng có ảnh hưởng tới tốc độ xây dựng không gian trạng thái cũng như tốc độ xây dựng cây cú pháp từ các trạng thái đã được tạo ra. Tính nhập nhằng gây ra bởi trường hợp một ký hiệu không kết thúc có thể đưa ra nhiều luật cú pháp, hoặc một từ có nhiều hơn một từ loại (POS). Tính nhập nhằng ảnh hưởng trực tiếp tới bộ nhớ cần sử dụng để lưu trữ các trạng thái có thể. Về việc xây dựng cây cú pháp, có thể xem xét tính nhập nhằng thông qua độ phức tạp về thời gian và không gian. Bản chất của việc đưa ra cây cú pháp (cho cả 2 thuật toán) là duyệt sâu. Độ phức tạp về thời gian là $\mathcal{O}(b \times d)$, độ phức tạp về không gian là $\mathcal{O}(b \times d)$. Trong đó, b là số văn phạm (hoặc luật từ tố) con tối đa của một văn phạm, d là độ dài của câu.

8 Khó khăn và giải quyết

Trong quá trình xây dựng bộ phân tích, tác giả cũng gặp một số khó khăn.

Ở các thuật toán CKY cũng như Earley nguyên thủy chỉ đề cập tới chuyện xem xét một câu có đúng trong miền văn phạm và luật từ tố đã cho hay không chứ không

đưa ra cách thức để đưa ra cây cú pháp. Cấu trúc dữ liệu thiết kế không hợp lý khó có thể lần ngược để xây dựng cây cú pháp. Ý tưởng đưa ra là sử dụng đệ quy để đưa ra cây cú pháp. Theo cách đó, tại mỗi thời điểm, trạng thái dẫn tới trạng thái hiện tại sẽ được lưu lại. Đây là kiểu dữ liệu dạng cây xây dựng ngược, từ lá lên gốc. Sau đó, cây cú pháp được đưa ra bởi phép duyệt cây tiền thứ tự (pre-order).

Ngoài ra, trong thuật toán Earley, các luật (văn phạm, luật từ tố) có sự biến đổi thứ tự dấu •. Mặt khác, các luật ở những cột trước vẫn cần dùng tới ở cột sau. Vấn đề nảy sinh là các kiểu dữ liệu cơ bản của Python đều ở dạng *mutable* (ngoại trừ tuple, tuy nhiên tuple không thể thêm bớt về sau). Điều đó có nghĩa, nếu không tạo đối tượng khác cho luật ở các cột sau mà vẫn dùng lại luật lấy từ cột trước để biến đổi thì đương nhiên luật ở cột trước cũng bị biến đổi theo. Sự biến đổi bất thường này sẽ làm cho kết quả của thuật toán không còn đúng nữa. Theo quan điểm của tác giả, khó khăn lớn nhất khi cài đặt thuật toán Earley dùng ngôn ngữ Python là tính *mutable* của các kiểu dữ liệu có sẵn trong Python. Để giải quyết khó khăn này, tác giả thao tác sự biến đổi trên các bản sao của dữ liệu. Ví dụ, sao chép một danh sách có thể dùng cú pháp *listCopy = list[:]* hoặc đưa danh sách về dạng tuple *listImmutable = tuple(list)*.

Kết luận

Trong phạm vi một bài tập lớn và khoảng thời gian cho phép, tác giả đã cố gắng xây dựng bộ phân tích cú pháp với hai thuật toán CKY và Earley. Yêu cầu bài toán đặt ra phù hợp với sinh viên trình độ nhập môn về Xử lý ngôn ngữ tự nhiên. Đối với bản thân tác giả, đây là một bài toán hay, có ích trong rèn luyện tư duy tổ chức dữ liệu và về tính *mutable* của các kiểu dữ liệu trong Python. Qua bài toán, tác giả có dịp hiểu thêm về các phương pháp phân tích cú pháp cho một câu văn bản. Ngoài hai phương pháp được cài đặt, còn có một số thuật toán khác như *Chart parsing* kết hợp hai phương pháp top–down và bottom–up hay một số thuật toán phân tích cú pháp kết hợp xác suất hoặc học máy.

Xử lý tiếng Việt đang là vấn đề được cộng đồng trong nước quan tâm. Do đó, tác giả đề xuất bài toán phân tích cú pháp cho câu tiếng Việt trong phiên bản tiếp theo của bài tập này. Cú pháp của câu tiếng Việt có nhiều khác biệt so với câu tiếng Anh nên các thuật toán được thiết kế cho tiếng Anh chưa hoàn toàn tốt khi áp dụng để phân tích cú pháp cho câu tiếng Việt. Có thể kể đến một số thuật toán cải tiến áp dụng cho tiếng Việt như trong [DLM04] hay [LH08].

Tài liệu

[VINHNV11] Nguyễn Văn Vinh, *Syntactic Parsing*, Slide NLP Course, 2011

[EAR70] Jay Earley,*An efficient context-free parsing algorithm*, Magazine Communications of the ACM, 1970

[JUL08] Julia Hockenmaier, *Parsing algorithms for CFGs*, University of Illinois at Urbana-Champaign, 2008

[SCF10] Scott Farrar, *Introduction Earley Algorithm*, University of Washington, 2010

[DLM04] Nguyễn Gia Định, Trần Thanh Lương, Lê Viết Mẫn, *Một số cải tiến giải thuật Earley cho việc phân tích cú pháp trong xử lý ngôn ngữ tự nhiên*, Tạp chí khoa học, Đại học Huế, 2004

[LH08] Đỗ Bá Lâm, Lê Thanh Hương, *Cải tiến giải thuật Earley trong phân tích cú pháp tiếng Việt*, đề tài VLSP, 2008