Implementation of weighted cfg parsers

Thomas Ruprecht

Chair for foundations of programming
Institute for theoretical computer science
TU Dresden

April 30th

Outline

Overview

CKY parsing

Deductive parsing

Conclusion

Overview

read binary cfg from file

G

read word from `stdin`

G, w

weighted CKY parsing or deductive parsing

print best derivation tree

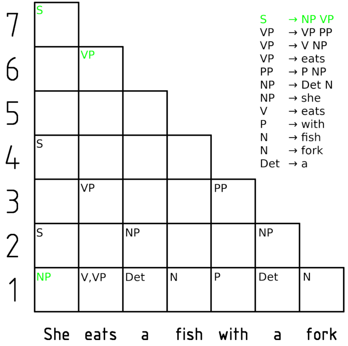
d

Outline

Weighted CKY parsing [CS70; Kas66; You67]

- requires cfg in Chomsky normal form
- fill chart from bottom to top
- each cell corresponds to span in word to parse
- add nonterminal to cell if partial derivation yields span

- store best weight along with nonterminals
- close each cell with chain rules
- store *backtrace* (rule, reference to predecessor cells)



Trougnouf, "CYK algorithm: animation showing every step of a sentence parsing", 15 January 2018, via wikimedia.org

The CKY parsing algorithm

Require: cfg $G = (N, \Sigma, P, S)$ in Cnf, word $t_1 ... t_n$ where $t_1, ..., t_n \in \Sigma$
Ensure: family of sets $(c_{i,j} \subseteq N \mid 0 \leq i < j \leq n)$ such that $A \in c_{i-1,j} \iff D_G^A(t_i ... t_j) \neq \emptyset$

function $\textsc{cky}(P, t_1 ... t_n)$
    for $1 \leq i \leq n$ **do**
        $c_{i-1,i} := \{A \mid A \to t_i \in P\}$
    for $2 \leq r \leq n$ **do**
        for $0 \leq i \leq n - r$ **do**
            $j := i + r$
            $c_{i,j} := \{A \mid i < m < j, A \to BC \in P \colon B \in c_{i,m}, C \in c_{m,j}\}$
    return $(c_{i,j} \mid 0 \leq i < j \leq n)$

The CKY parsing algorithm

Require: cfg $G = (N, \Sigma, P, S)$ in Cnf, word $t_1 ... t_n$ where $t_1, ..., t_n \in \Sigma$
Ensure: family of sets $(c_{i,j} \in N \mid 0 \leq i < j \leq n)$ such that $A \in c_{i-1,j} \iff D_G^A(t_i ... t_j) \neq \emptyset$

function $\textsc{cky}(P,\, t_1 ... t_n)$
  for $1 \leq i \leq n$ **do**
    $c_{i-1,i} := \emptyset$
    for $A \to t_i \in P$ **do**
      $c_{i-1,i} \cup= \{A\}$
  for $2 \leq r \leq n$ **do**
    for $0 \leq i \leq n - r$ **do**
      $j := i + r$
      $c_{i,j} := \emptyset$
      for $A \in N$ **do**
        for $m \in \{l+1, l+2, ..., r-1\}$ **do**
          for $A \to BC \in R$ **do**
            if $B \in c_{i,m}$ and $C \in c_{m,j}$ **then**
              $c_{i,j} \cup= \{A\}$
  return $(c_{i,j} \mid 0 \leq i < j \leq n)$

The CKY parsing algorithm + weights

Require: weighted cfg $G = (N, \Sigma, P, S, \mu)$ in Cnf, word $t_1 ... t_n$ where $t_1, ..., t_n \in \Sigma$
Ensure: family of mappings $(c_{i,j} \colon N \to \mathbb{R} \mid 0 \leq i < j \leq n)$ such that
$c_{i-1,j}(A) = \max\{\mu(d) \mid d \in D_G^A(t_i ... t_j)\} \cup \{0\}$

function CKY(P, μ, $t_1 ... t_n$)
  for $1 \leq i \leq n$ **do**
    $c_{i-1,i} := A \mapsto \max\{\mu(A \to t_i) \mid A \to t_i \in P\} \cup \{0\}$
  for $2 \leq r \leq n$ **do**
    for $0 \leq i \leq n - r$ **do**
      $j := i + r$
      $c_{i,j} := A \mapsto \max\{\mu(A \to BC) \cdot c_{i,m}(B) \cdot c_{m,j}(C) \mid i < m < j, A \to BC \in P\} \cup \{0\}$
  return $(c_{i,j} \mid 0 \leq i < j \leq n)$

The CKY parsing algorithm + weights

Require: weighted cfg (N, Σ, P, S, μ) in Cnf, word $t_1 \ldots t_n$ where $t_1, \ldots, t_n \in \Sigma$
Ensure: family of mappings $(c_{i,j} \colon N \to \mathbb{R} \mid 0 \leq i < j \leq n)$ such that

$$c_{i-1,j}(A) = \max\{\mu(d) \mid d \in D_G^A(t_i \ldots t_j)\} \cup \{0\}$$

function $\textsc{cky}(P, \mu, t_1 \ldots t_n)$
  $(c_{i,j,A} := 0 \mid 0 \leq i < j \leq n, A \in N)$
  for $1 \leq i \leq n$ **do**
    for $A \to t_i \in P$ **do**
      $c_{i-1,i,A} := \max\{c_{i-1,i,A}, \mu(A \to t_i)\}$
  for $2 \leq r \leq n$ **do**
    for $0 \leq i \leq n - r$ **do**
      $j := i + r$
      for $A \in N$ **do**
        for $m \in \{i+1, i+2, \ldots, j-1\}$ **do**
          for $A \to BC \in R$ **do**
            $c_{i,j,A} := \max\{c_{i,j,A}, \mu(A \to BC) \cdot c_{i,m,B} \cdot c_{m,j,C}\}$
  return $(c_{i,j} := A \mapsto c_{i,j,A} \mid 0 \leq i < j \leq n)$

The CKY parsing algorithm + weights + chain rules

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 ... t_n$ where $t_1, ..., t_n \in \Sigma$
Ensure: family of mappings $(c_{i,j} \colon N \to \mathbb{R} \mid 0 \le i < j \le n)$ such that

$$c_{i,j}(A) = \max\{\mu(d) \mid d \in D_G^A(t_i ... t_j)\} \cup \{0\}$$

function $\textsc{cky}(P,\ \mu,\ t_1 ... t_n)$
    $(c_{i,j,A} := 0 \mid 0 \le i < j \le n, A \in N)$
    for $1 \le i \le n$ **do**
        for $A \to t_i \in P$ **do**
            $c_{i-1,i,A} := \max\{c_{i-1,i,A}, \mu(A \to t_i)\}$
        $c' = \textsc{unary\_closure}(P, \mu, (c_{i-1,i,A} \mid A \in N))$
        $(c_{i-1,i,A} := c'_A \mid A \in N)$
    for $2 \le r \le n$ **do**
        for $0 \le i \le n - r$ **do**
            $j := i + r$
            for $A \in N$ **do**
                for $m \in \{i+1, i+2, ..., j-1\}$ **do**
                    for $A \to BC \in R$ **do**
                        $c_{i,j,A} := \max\{c_{i,j,A}, \mu(A \to BC) \cdot c_{i,m,B} \cdot c_{m,j,C}\}$
            $c' = \textsc{unary\_closure}(P, \mu, (c_{i,j,A} \mid A \in N))$
            $(c_{i,j,A} := c'_A \mid A \in N)$
    return $(c_{i,j} := A \mapsto c_{i,j,A} \mid 0 \le i < j \le n)$

The CKY parsing algorithm + weights + chain rules

function $\textsc{unary\_closure}(P,\ \mu,\ (c_A \in \mathbb{R} \mid A \in N))$

  $queue := \{(A, c_A) \in N \times \mathbb{R} \mid A \in N, c_A \neq 0\}$

  $(c_A := 0 \mid A \in N)$

  while $queue \neq \emptyset$ **do**

    $(B, w) := \operatorname{argmax}_{(B,w) \in queue}\ w$

    $queue \setminus= \{(B, w)\}$

    if $c_B < w$ **then**

      $queue \cup= \{(A, \mu(A \to B) \cdot w) \mid A \to B \in P\}$

      $c_B := w$

  return $(c_A \mid A \in N)$

About backtraces

- only best derivation:
  - store at most *one* backtrace per span and nonterminal
  - update when weight is updated
- recursively read trees from backtraces:

Require: family of backtraces $(b_{i,j,A} \mid 0 \leq i < j \leq n, A \in N)$, each otf. $\perp$, or $A \to t$, or $(A \to B, i, j)$ or $(A \to BC, i, m, m, j)$

function $\textsc{first\_tree}(b,\ i,\ j,\ A)$
  if $b_{i,j,A}$ otf. $A \to t$ **then return** $A \to t$
  else if $b_{i,j,A}$ otf. $(A \to B, i, j)$ **then return** $(A \to B)(\textsc{first\_tree}(b, i, j, B))$
  else if $b_{i,j,A}$ otf. $(A \to BC, i, m, m, j)$ **then**
    return $(A \to BC)(\textsc{first\_tree}(b, i, m, B), \textsc{first\_tree}(b, m, j, C))$

Let's talk about data structures

- access to grammar rules depends on loops:
  - access by first nonterminal on rhs `Map<Nt, Set<(Rule, Wt)>>`
  - for some, that be no concern `Set<(Rule, Wt)>`
- weights for each nonterminal and span:
  - usually in a $(\frac{|w|\cdot(|w|+1)}{2} \cdot |N|)$-dimensional vector (dense) `Vec<Wt>`
  - or hashmap (sparse) `Map<(Int, Nt, Int), Wt>`
- storing backtraces:
  - each backtrace: applied rule and references to cells
    `Bt = Bin(Rule, [Int; 4])+ Chain(Rule, [Int; 2])+ Term(Rule)`
  - backtraces for each chart cell and nonterminal `Vec<Set<Bt>>`
  - or do not store them at all

Outline

Overview

CKY parsing

Deductive parsing

Conclusion

Deduction systems [Ned03]

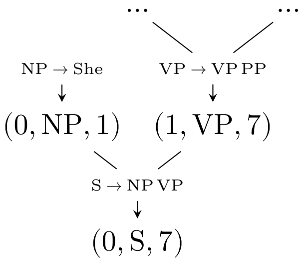
- rule-based system
- derive consequence (c) from antecedents ($a_1, ..., a_k$) for some $k \in \mathbb{N}$
- compute weight of consequence using weight of antecedents ($w_1, ..., w_k$)
- side condition b

$$\frac{a_1 : w_1, ..., a_k : w_k}{c\colon f(w_1,...,w_k)}\ b$$

Deduction system for parsing weighted cfg [Ned03]

- item (i, A, j) for each nonterminal A spanning $t_i \ldots t_j$
- predict initial items $\frac{}{(i-1,A,i)\colon\, \mu(A \longrightarrow t_i)}\ A \to t_i \in P \wedge w = t_1 \ldots t_i \ldots t_n$
- combine items $\frac{(i_0,B_1,i_1)\colon\, w_1,(i_1,B_2,i_2)\colon\, w_2,\ldots,(i_{k-1},B_k,i_k)\colon\, w_k}{(i_0,A,i_k)\colon\, \mu(A \to B_1 \ldots B_k) \cdot w_1 \cdots w_k}\ A \to B_1 \ldots B_k \in P$
- goal item: $(0, S, |w|)$

- deduction system $\rightsquigarrow$ weighted hypergraph
  - edge from antecedents to consequence
  - can be explored with respect weight
  - hyperpaths to goal item correspond to parse trees



Weighted deductive parsing algorithm

Require: weighted binary cfg (N, Σ, P, S, μ), word $t_1 \ldots t_n$ where $t_1, \ldots, t_n \in \Sigma$
Ensure: family of mappings $(c_{i,j} \colon N \to \mathbb{R} \mid 0 \le i < j \le n)$ such that
$c_{i-1,j}(A) = \max\{\mu(d) \mid d \in D_G^A(t_i \ldots t_j)\} \cup \{0\}$

function DEDUCE(P, μ, $t_1 \ldots t_n$)
  $queue := \{(i-1, A, i, \mu(A \to t_i)) \mid 1 \le i \le n, A \to t_i \in P\}$
  $(c_{i,j,A} := 0 \mid 0 \le i < j \le n, A \in N)$
  while $queue \ne \emptyset$ **do**
    $(i, A, j, w) := \operatorname{argmax}_{(i,A,j,w) \in queue} w$
    $queue \setminus= \{(i, A, j, w)\}$
    if $c_{i,j,A} = 0$ **then**
      $c_{i,j,A} := w$
      $queue \cup= \{(i, A', j', \mu(A' \to AC) \cdot w \cdot c_{j,j',C}) \mid A' \to AC \in P\}$
      $queue \cup= \{(i', A', j, \mu(A' \to BA) \cdot c_{i',i,B} \cdot w) \mid A' \to BA \in P\}$
      $queue \cup= \{(i, A', j, \mu(A' \to A) \cdot w) \mid A' \to A \in P\}$
  return $(c_{i,j} := A \mapsto c_{i,j,A} \mid 0 \le i < j \le n)$

Let's talk about data structures … again

- access of grammar from each rhs nonterminal `Map<Nt, Set<(Rule, Wt)>>`
- each item may need to sotre a backtrace `(Int, Nt, Int, Wt, Bt)`
- storing the found items and their weights:
  - access from left `Map<(Int, Nt), Set<(Int, Nt, Int, Wt)>>`
  - access from right `Map<(Nt, Int), Set<(Int, Nt, Int, Wt)>>`
- storing backtraces:
  - store applied rule and antecedent items
    `Bt = Bin(Rule, [Int; 4])+ Chain(Rule, [Int; 2])+ Term(Rule)`
  - set of backtraces for item `Map<(UInt, Nt, UInt), Set<Bt>>`
  - or do not store them at all

Outline

General Comments and Tips

- order of loops in CKY algorithm doesn't matter that much, *but*[1]:
  - may be used to cache-optimize,
  - may lead to other optimizations
- deductive parsers may not need to expand the whole search space
- try to think about efficient access in your data structures
  - don't search in lists
  - indexed access: maps
  - check if you *really* need sets/maps
  - flat data structures are faster than stacked heap allocations
- try not to over-engineer it

[1] Bodenstab [Bod09] discusses this in detail.

[Bod09] Nathan Bodenstab. "Efficient Implementation of the cky algorithm". In: *Computational Linguistics, Final Project Paper* (2009).

[CS70] John Cocke and J. T. Schwartz. *Programming languages and their compilers: Preliminary notes*. Tech. rep. Version 2nd. Courant Institute of Mathematical Sciences, New York University, Apr. 1970.

[HC05] Liang Huang and David Chiang. "Better k-best parsing". In: *Proceedings of the Ninth International Workshop on Parsing Technology*. Association for Computational Linguistics. 2005, pp. 53–64.

[Kas66] T. Kasami. *An efficient recognition and syntax-analysis algorithm for context-free languages*. Tech. rep. R-257. AFCRL, Mar. 1966.

[Ned03] Mark-Jan Nederhof. "Weighted deductive parsing and Knuth's algorithm". In: *Computational Linguistics* 29.1 (2003), pp. 135–143.

[You67] Daniel H. Younger. "Recognition and parsing of context-free languages in time n3". In: *Information and Control* 10.2 (Feb. 1967), pp. 189–208. DOI: 10.1016/s0019-9958(67)80007-x.