Dynamic Programming for Parsing and Estimation of Stochastic Unification-Based Grammars

Stuart Geman and Mark Johnson

Brown University

ACL'02, Philadelphia

Acknowledgements: Stefan Riezler (Parc)

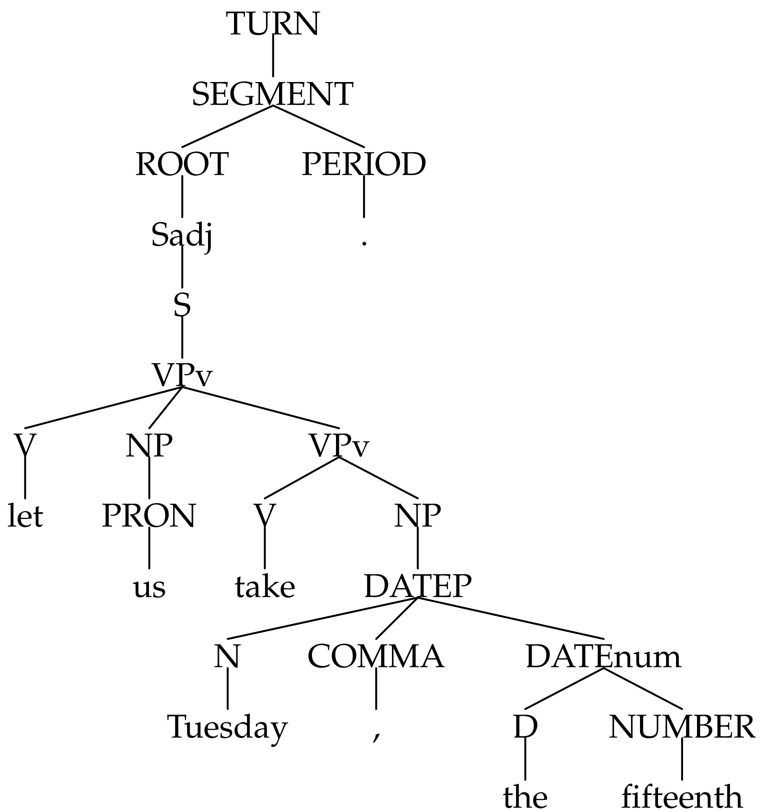
NSF grants DMS 0074276 and ITR IIS 0085940

Talk outline

- Stochastic Unification-Based Grammars (SUBGs)
- Parsing and estimation of SUBGs
- *Avoiding enumerating all parses*
  - Maxwell and Kaplan (1995) packed parse representations
  - Feature locality
  - Parse weight is a *product of functions of parse fragment weights*
  - Graphical model calculation of argmax/sum

Related work: Miyao and Tsuji (2002) "Maximum Entropy Estimation for Feature Forests" HLT

Lexical-Functional Grammar (UBG)




```
SENTENCE_ID   BAC002_E
    OBJ        ANIM  +
               CASE  ACC
               NUM   PL
               PERS  1
               PRED  PRO
               PRON-FORM  WE
             9 PRON-TYPE  PERS
    PASSIVE  –
    PRED     LET⟨2,10⟩9
    STMT-TYPE  IMPERATIVE
    SUBJ       PERS  2
               PRED  PRO
             2 PRON-TYPE  NULL
    TNS-ASP  [ MOOD  IMPERATIVE ]
    XCOMP         OBJ   ANIM  –
                        APP   NTYPE  [ NUMBER  ORD
                                       TIME  DATE ]
                              NUM   SG
                              PRED  fifteen
                              SPEC  [ SPEC-FORM  THE
                                      SPEC-TYPE  DEF ]
                        CASE  ACC
                        GEND  NEUT
                        NTYPE  [ GRAIN  COUNT
                                 PROPER  DATE
                                 TIME  DAY ]
                        NUM   SG
                        PERS  3
                     13 PRED  TUESDAY
                  PASSIVE  –
                  PRED  TAKE⟨9,13⟩
    3             SUBJ  [9]
```

Stochastic Unification-based Grammars

- A *unification-based grammar* defines a *set of possible parses* $\mathcal{Y}(w)$ for each sentence w.
- *Features* $f_1, \ldots, f_m$ are real-valued functions on parses
  - Attachment location (high, low, argument, adjunct, etc.)
  - Head-to-head dependencies
- Probability defined by *conditional log-linear model*

$$\begin{aligned} W(y) &= \exp(\sum_{j=1}^{m} \lambda_j f_j(y)) = \prod_{j=1}^{m} \theta_j^{f_j(y)} \\ \Pr(y|w) &= W(y)/Z(w) \end{aligned}$$

where $\theta_j = e^{\lambda_j} > 0$ are *feature weights* and $Z(w) = \sum_{y \in \mathcal{Y}(w)} W(y)$ is the *partition function*.

Johnson et al (1999) "Parsing and estimation for SUBGs", Proc ACL

Estimating feature weights

- Several algorithms for *maximum conditional likelihood estimation*
  - Various iterative scaling algorithms
  - *Conjugate gradient* and other optimization algorithms
- These algorithms are *iterative*
  $\Rightarrow$ repeated *reparsing of training data*
- All of these algorithms require *conditional expectations*

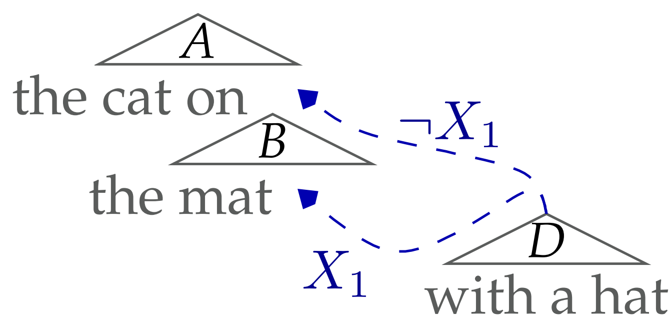
$$E[f_j|w] \;=\; \sum_{y \in \mathcal{Y}(w)} f_j(y) \Pr(y|w)$$

- Can we calculate these statistics and find the most likely parse without enumerating all parses $\mathcal{Y}(w)$? *YES* *

Maxwell and Kaplan packed parses

- A parse y consists of set of fragments $\xi \in y$ (MK algorithm)
- A fragment is in a parse when its *context function* is true
- Context functions are functions of *context variables* $X_1, X_2, \ldots$
- The variable assignment must satisfy "not no-good" functions
- Each parse is identified by a *unique context variable assignment*

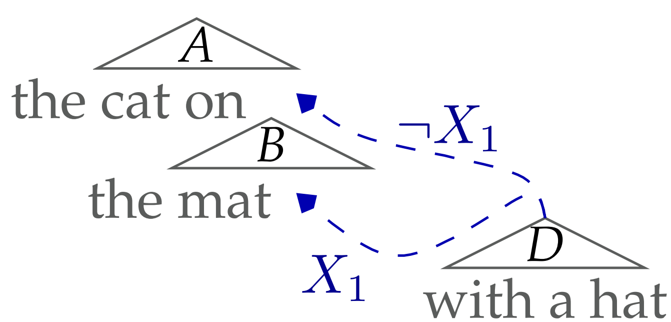
$$\begin{aligned} \xi &= \text{``\textit{the cat on the mat}''} \\ \xi_1 &= \text{``\textit{with a hat}''} \\ X_1 &\rightarrow \text{``attach } D \text{ to } B\text{''} \\ \neg X_1 &\rightarrow \text{``attach } D \text{ to } A\text{''} \end{aligned}$$



Feature locality

- Features must be *local* to fragments: $f_j(y) = \sum_{\xi \in y} f_j(\xi)$
- May require changes to UBG to make all features local

$$\begin{aligned}
\xi &= \text{“\textit{the cat on the mat}”} \\
\xi_1 &= \text{“\textit{with a hat}”} \\
X_1 &\rightarrow \text{“attach } D \text{ to } B\text{”} \wedge (\xi_1 \text{ ATTACH}) = \text{LOW} \\
\neg X_1 &\rightarrow \text{“attach } D \text{ to } A\text{”} \wedge (\xi_1 \text{ ATTACH}) = \text{HIGH}
\end{aligned}$$



Feature locality decomposes $W(y)$

- Feature locality: the weight of a parse is the product of weights of its fragments

$$
\begin{aligned}
W(y) &= \prod_{\xi \in y} W(\xi), \qquad \text{where} \\
W(\xi) &= \prod_{j=1}^{m} \theta_j^{f_j(\xi)}
\end{aligned}
$$

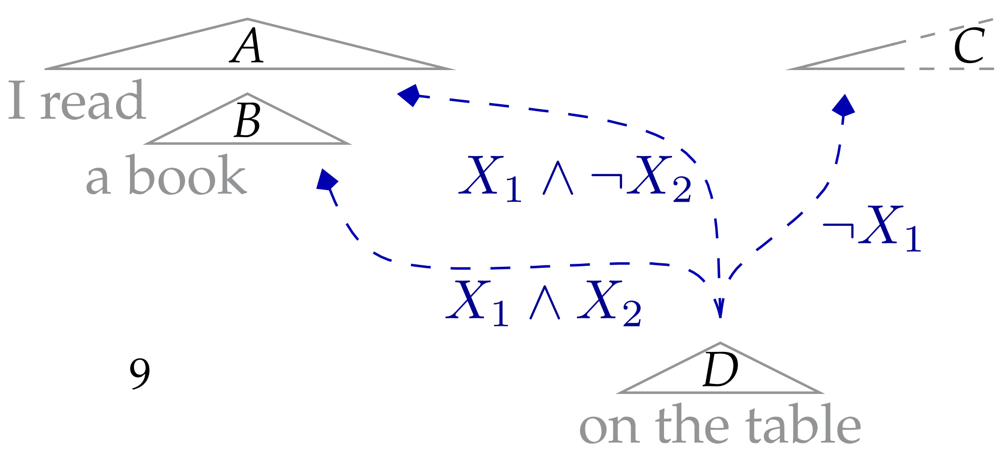
$$
\begin{aligned}
W(\xi &= \textit{“the cat on the mat”}) \\
W(\xi_1 &= \textit{“with a hat”})
\end{aligned}
$$

$$
\begin{aligned}
X_1 &\rightarrow W\,(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1\ \text{ATTACH}) = \text{LOW}\ ) \\
\neg X_1 &\rightarrow W\,(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1\ \text{ATTACH}) = \text{HIGH}\ )
\end{aligned}
$$

Not No-goods

- "Not no-goods" identify the variable assignments that correspond to parses

$$\begin{aligned} \xi &= \textit{"I read a book"} \\ \xi_1 &= \textit{"on the table"} \\ X_1 \wedge X_2 &\rightarrow \text{"attach } D \text{ to } B\text{"} \\ X_1 \wedge \neg X_2 &\rightarrow \text{"attach } D \text{ to } A\text{"} \\ \neg X_1 &\rightarrow \text{"attach } D \text{ to } C\text{"} \\ & X_1 \vee X_2 \end{aligned}$$



Identify parses with variable assignments

- *Each variable assignment uniquely identifies a parse*
- For a given sentence w, let $W'(x) = W(y)$ where y is the parse identified by x

$\Rightarrow$ Argmax/sum/expectations over parses can be computed over context variables instead

Most likely parse: $\hat{x} = \operatorname{argmax}_x W'(x)$

Partition function: $Z(w) = \sum_x W'(x)$

Expectation:* $E[f_j|w] = \sum_x f_j(x)W'(x)/Z(w)$

W' is a product of functions of X

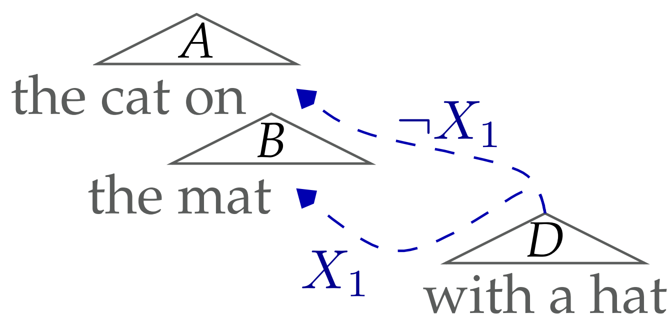
- Then $W'(X) = \prod_{A \in \mathcal{A}} A(X)$, where:
  - Each line $\alpha(X) \rightarrow \xi$ introduces a term $W(\xi)^{\alpha(X)}$
  - A "not no-good" $\eta(X)$ introduces a term $\eta(X)$

$$
\begin{array}{cccccc}
 & & \vdots & & & \vdots \\
\alpha(X) & \rightarrow & \xi & & \times & W(\xi)^{\alpha(X)} \\
 & & \vdots & & \times & \vdots \\
 & & & \eta(X) & \times & \eta(X) \\
 & & \vdots & & \times & \vdots
\end{array}
$$

$\Rightarrow$ W' is a *Markov Random Field* over the context variables X

W' is a product of functions of X

$$
\begin{aligned}
W'(X_1) \;=\; & \quad W(\xi \;=\; \text{“\textit{the cat on the mat}”}) \\
& \times\; W(\xi_1 \;=\; \text{“\textit{with a hat}”}) \\
& \times\; W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{low}\right)^{X_1} \\
& \times\; W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{high}\right)^{\neg X_1}
\end{aligned}
$$



Product expressions and graphical models

- MRFs are products of terms, each of which is a function of (a few) variables
- Graphical models provide *dynamic programming algorithms* for Markov Random Fields (MRF) (Pearl 1988)
- These algorithms implicitly *factorize the product*
- They generalize the Viterbi and Forward-Backward algorithms to arbitrary graphs (Smyth 1997)

$\Rightarrow$ Graphical models provide dynamic programming techniques for parsing and training Stochastic UBGs

Factorization example

$$
\begin{aligned}
W'(X_1) \;=\; & \quad W(\xi = \text{“\textit{the cat on the mat}”}) \\
\times & \quad W(\xi_1 = \text{“\textit{with a hat}”}) \\
\times & \quad W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{low}\right)^{X_1} \\
\times & \quad W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{high}\right)^{\neg X_1}
\end{aligned}
$$

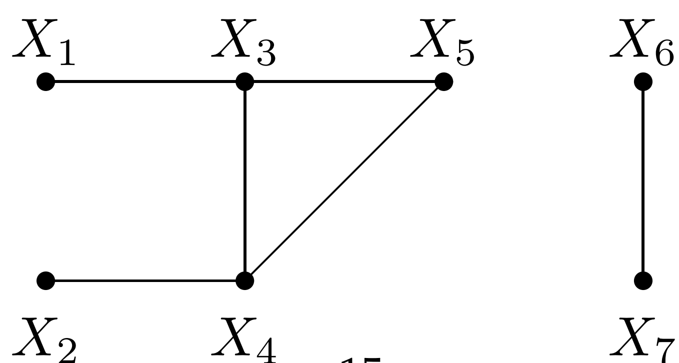
$$
\begin{aligned}
\max_{X_1} W'(X_1) \;=\; & \quad W(\xi = \text{“\textit{the cat on the mat}”}) \\
\times & \quad W(\xi_1 = \text{“\textit{with a hat}”}) \\
\times & \quad \max_{X_1} \begin{pmatrix} W\left(\text{“attach } D \text{ to } B\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{low}\right)^{X_1}, \\ W\left(\text{“attach } D \text{ to } A\text{”} \;\wedge\; (\xi_1\ \textsc{attach}) = \textsc{high}\right)^{\neg X_1} \end{pmatrix}
\end{aligned}
$$

Dependency structure graph $G_{\mathcal{A}}$

$$Z(w) \;=\; \sum_x W'(x) \;=\; \sum_x \prod_{A \in \mathcal{A}} A(x)$$

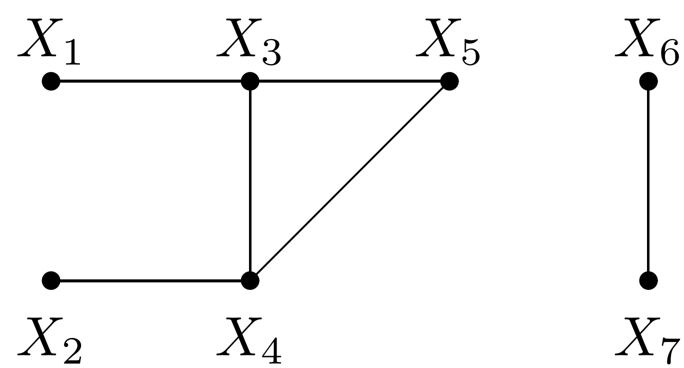
- $G_{\mathcal{A}}$ is the *dependency graph* for $\mathcal{A}$
  - context variables X are vertices of $G_{\mathcal{A}}$
  - $G_{\mathcal{A}}$ has an edge (X_i, X_j) if both are arguments of some $A \in \mathcal{A}$

$$A(X) = a(X_1, X_3)b(X_2, X_4)c(X_3, X_4, X_5)d(X_4, X_5)e(X_6, X_7)$$



Graphical model computations

$$
\begin{aligned}
Z &= \textstyle\sum_x a(x_1,x_3)b(x_2,x_4)c(x_3,x_4,x_5)d(x_4,x_5)e(x_6,x_7) \\
Z_1(x_3) &= \textstyle\sum_{x_1} a(x_1,x_3) \\
Z_2(x_4) &= \textstyle\sum_{x_2} b(x_2,x_4) \\
Z_3(x_4,x_5) &= \textstyle\sum_{x_3} c(x_3,x_4,x_5)Z_1(x_3) \\
Z_4(x_5) &= \textstyle\sum_{x_4} d(x_4,x_5)Z_2(x_4)Z_3(x_4,x_5) \\
Z_5 &= \textstyle\sum_{x_5} Z_4(x_5) \\
Z_6(x_7) &= \textstyle\sum_{x_6} e(x_6,x_7) \\
Z_7 &= \textstyle\sum_{x_7} Z_6(x_7) \\
Z &= Z_5 Z_7 \\
&= \left(\textstyle\sum_{x_5} Z_4(x_5)\right)\left(\textstyle\sum_{x_7} Z_6(x_7)\right)
\end{aligned}
$$



See: Pearl (1988) *Probabilistic Reasoning in Intelligent Systems*

Graphical model for Homecentre example

Use a damp, lint-free cloth to wipe the dust and dirt buildup from the scanner plastic window and rollers.

Computational complexity

- Polynomial in $m =$ the *maximum number of variables in the dynamic programming functions* $\geq$ the number of variables in any function A

- m depends on the ordering of variables (and G)

- Finding the variable ordering that minimizes m is NP-complete, but there are good heuristics

$\Rightarrow$ Worst case exponential (no better than enumerating the parses), but average case might be much better

  – Much like UBG parsing complexity

Conclusion

- There are DP algorithms for parsing and estimation from packed parses that avoid enumerating parses
  - Generalizes to all Truth Maintenance Systems (not grammar specific)
- Features must be local to parse fragments
  - May require adding features to the grammar
- Worst-case exponential complexity; average case?
- Makes available techniques for graphical models to packed parse representations
  - MCMC and other sampling techniques

Future directions

- Reformulate “hard” grammatical constraints as “soft” stochastic features
  - Underlying grammar permits all possible structural combinations
  - Grammatical constraints reformulated as stochastic features
- Is this computation tractable?
- Comparison with Miyao and Tsujii (2002)