Syntactic analysis

Probabilities in syntactic parsing

Efficiency in parsing with unification grammars

PROBABILISTIC (CONTEXT-FREE) PARSING

Motivation

Ambiguity in natural language considerable
(for syntactic parsing)

Not all grammar rules are of equal right (preferences)
Some express rather uncommon patterns

Potential for speeding up

Possible remedies

First parse only with common rules
Use remaining ones only if needed

Better: every rules has a weight
Pick the overall “lightest” parse (weights are accummulated)

Potential for learning/accommodating weights

PROBABILISTIC CONTEXT-FREE GRAMMARS

Basic idea

Modification of context-free grammars

Associating a probability $P(\beta \mid X)$ with each grammar rule

Probability of expanding X using the rule $X \rightarrow \beta$ rather than some other rule

Computation

Probability of a derivation is the product of the probabilities

associated with the rules applied in the derivation

Probablility of a tree-sentence pair (T,S)

derived by n applications of context-free rules $LHS_i \rightarrow RHS_i$

$$P(T,S) = \Pi_{i=1,n} P(RHS_i \mid LHS_i)$$

PARSING WITH WEIGHTS (1)

	time 1	flies 2	like 3	an 4	arrow 5
0	NP 3 Vst 3				
1		NP 4 VP 4			
2			P 2 V 5		
3				Det 1	
4					N 8

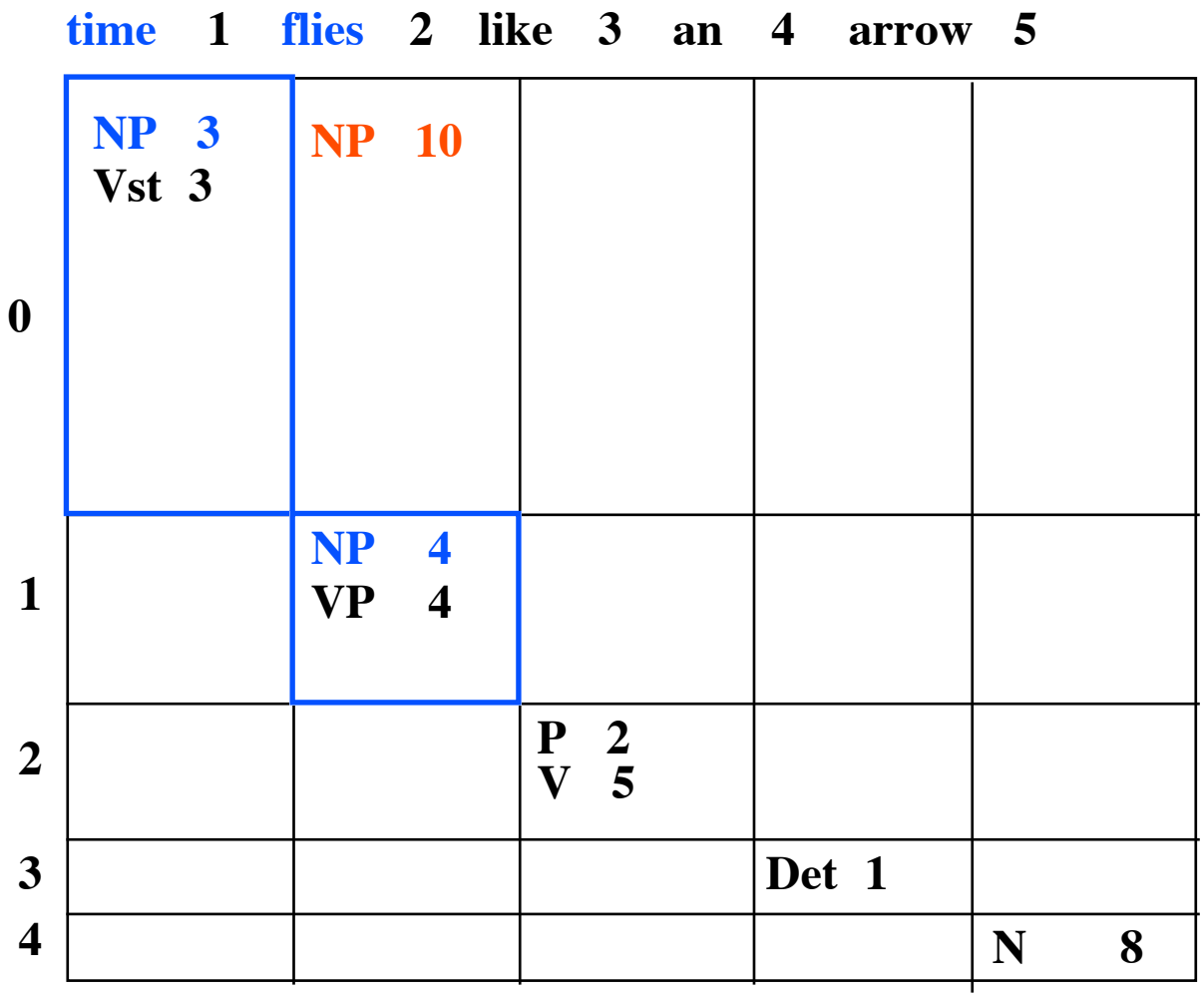
1 S -> NP VP
6 S -> Vst NP
2 S -> S PP

1 VP -> V NP
2 VP -> VP PP

1 NP -> Det N
2 NP -> NP PP
3 NP -> NP NP

0 PP -> P NP

PARSING WITH WEIGHTS (2)



1 S -> NP VP
6 S -> Vst NP
2 S -> S PP

1 VP -> V NP
2 VP -> VP PP

1 NP -> Det N
2 NP -> NP PP
3 NP -> NP NP

0 PP -> P NP

PARSING WITH WEIGHTS (3)

	time 1	flies 2	like 3	an 4	arrow 5
0	NP 3 Vst 3	NP 10 S 8 S 13	–	–	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	–	–	NP 18 S 21 VP 18
2			P 2 V 5	–	PP 12 VP 16
3				Det 1	NP 10
4					N 8

1 S -> NP VP
6 S -> Vst NP
2 S -> S PP
1 VP -> V NP
2 VP -> VP PP
1 NP -> Det N
2 NP -> NP PP
3 NP -> NP NP
0 PP -> P NP

PARSING WITH WEIGHTS (4)

	time 1	flies 2	like 3	an 4	arrow 5
0	NP 3 Vst 3	NP 10 S 8 S 13	–	–	NP 24 S 22 S 27 NP 24 S 27 S 22 S 27
1		NP 4 VP 4	–	–	NP 18 S 21 VP 18
2			P 2 V 5	–	PP 12 VP 16
3				Det 1	NP 10
4					N 8

inferior entries

1 S -> NP VP
6 S -> Vst NP
2 S -> S PP
1 VP -> V NP
2 VP -> VP PP
1 NP -> Det N
2 NP -> NP PP
3 NP -> NP NP
0 PP -> P NP

PROPERTIES OF SIMPLE PROBABILISTIC GRAMMARS (I)

Assumptions

Independent of the place

(e.g., pronouns mostly appear in subject position)

Independent of the context

(words embedding)

Independent of the structural embedding (derivation tree embedding)

Hence, statistics is a simple count how often local tree configurations occurred

PROPERTIES OF SIMPLE PROBABILISTIC GRAMMARS (II)

Benefits

Partial solution for grammar ambiguity (some idea of plausibility)

Robustness (also everything with low probabilities can be admitted)

Potential to combine probabilistic grammars with trigram models

Deficits

In the simple case, a worse language model for English than trigrams

Encodes certain biases (e.g., smaller trees are normally more probable)

Enrichments for grammar needed (most state-of-the-art parser are so)

Independent of lexical material used

LEXICALIZED CONTEXT-FREE GRAMMARS

Motivation – Exploiting lexical information

Attempts with lexical grammars

Associating syntactic categories with lexical heads

Incorporating lexical heads of mother constituent into probabilities

No corpus big enough to train (very sparse for some word combinations)

Choice of head sometimes linguistically controversial

Exploiting structural information

Preference for right-branching structures in English

Basic idea

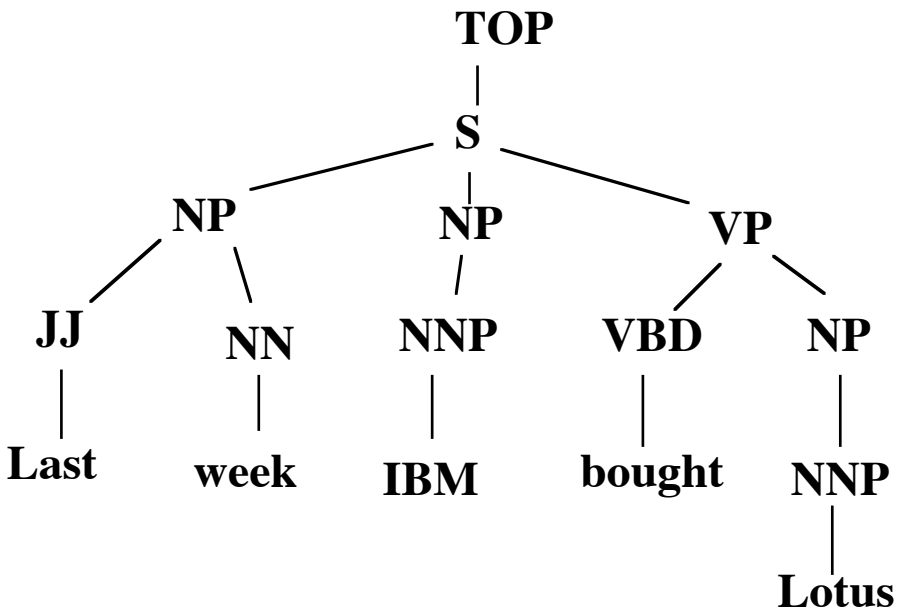
Associating a word *w* and a part-of-speech (POS) tag *t* with each nonterminal *X*

Nonterminals then become *X*(*w*,*t*)

LEXICALIZED CONTEXT-FREE GRAMMAR

AN EXAMPLE (1)

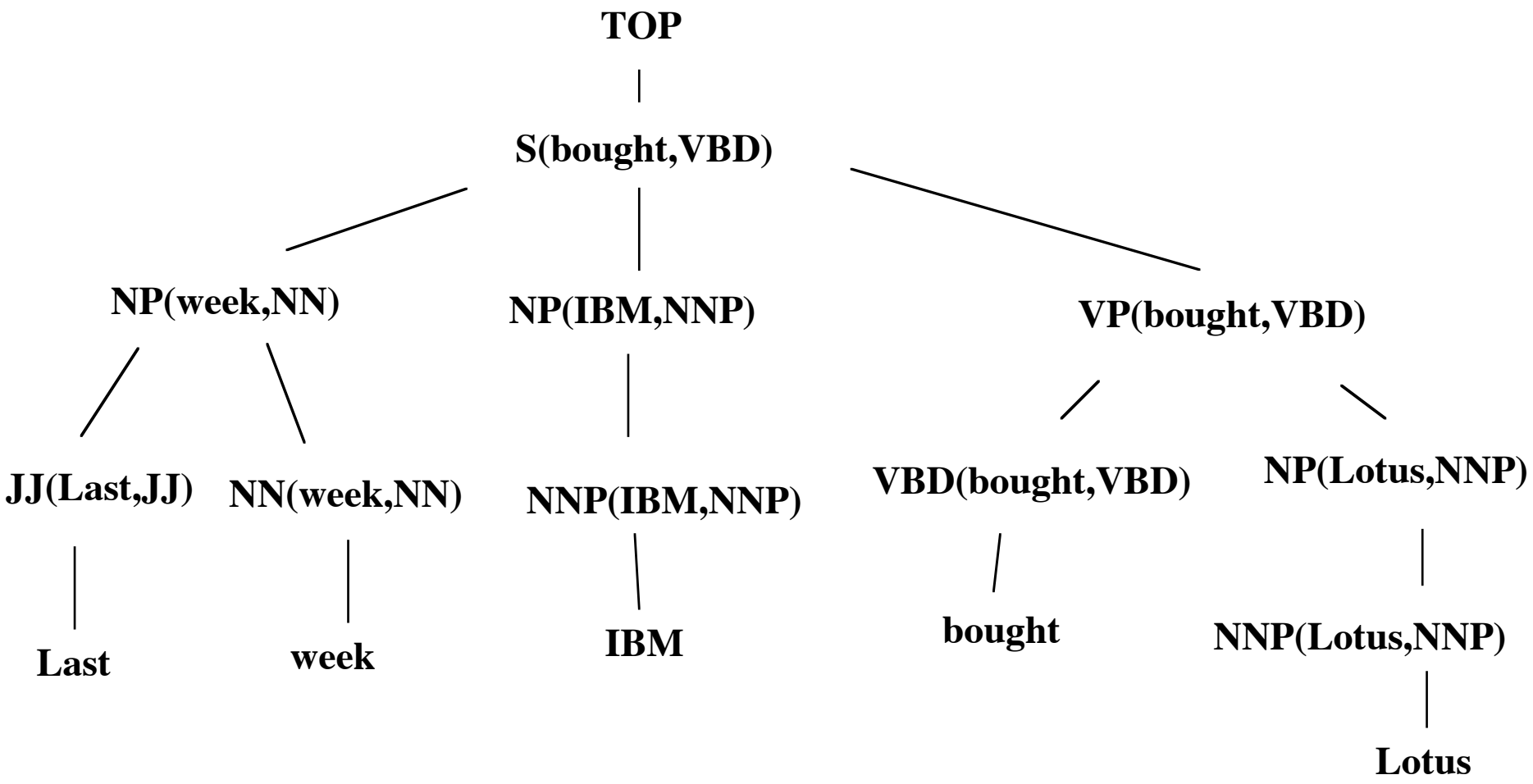
A non-lexicalized parse tree



A list of derivation rules

Internal rules	Lexical rules
TOP → S	JJ → Last
S → NP NP VP	NN → week
NP → JJ NN	NNP → IBM
NP → NNP	VBD → bought
VP → VBD NP	NNP → Lotus
NP → NNP	

LEXICALIZED CONTEXT-FREE GRAMMAR
AN EXAMPLE (2)



LEXICALIZED CONTEXT-FREE GRAMMAR AN EXAMPLE (3)

Internal rules

TOP → S(bought,VBD)

S(bought,VBD) → NP(week,NN)NP(IBM,NNP) VP(bought,VBD)

NP(week,NN) → JJ(Last,JJ) NN(week,NN)

NP(IBM,NNP) → NNP(IBM,NNP)

VP(bought,VBD) → VBD(bought,VBD) NP(Lotus,NNP)

NP(Lotus,NNP) → NNP(Lotus,NNP)

Lexical rules

JJ(Last,JJ) → Last

NN(week,NN) → week

NNP(IBM,NNP) → IBM

VBD(bought,VBD) → bought

NNP(Lotus,NNP) → Lotus

CONSEQUENCES OF THE LEXICALIZATION

Problem

Expanding the number of non-terminals increases the number of rules

Parameter estimation of the probabilities severely affected

Data for maximum likelihood becomes very sparse

Example

P(NP(week,NN) NP(IBM,NNP) VP(bought,VBD)) | S(bought,VBD) =

$$\frac{\textit{Count}(\text{S(bought,VBD)} \rightarrow \text{NP(week,NN) NP(IBM,NNP) VP(bought,VBD))}}{\textit{Count}(\text{S(bought,VBD)})}$$

Solution

Breaking down the right hand side of the rules into a sequence of smaller steps

Size of steps should be small enough for the parameter estimation to be feasible

The independence assumption should be linguistically plausible

BREAKING DOWN RULES

Idea

Centered around the *head child* of a rule

Partitioning a rule into its head and its left and right modifiers

Probability of the rule becomes product of the probabilities of the head and all modifiers

Extensions

Incorporating "distance features"

(weakening the independence assumption between modifiers)

Complement/Adjunct distinction and subcategorization

Traces and wh-movement

FURTHER EXTENSIONS

Preferences

Right-branching structures (dependencies between adjacent words)

Dependencies do not cross a verb

Refinements

Nonrecursive NPs

Coordination

Punctuation

Sentences with empty subjects

RESULTS

Dependency accuracy for major subtypes of dependencies

	Recall	Precision
Complement to a verb	**93,76%**	**92,96%**
Other complements	**94,47%**	**94,12%**
PP modification	**82,29%**	**81,51%**
Coordination	**61,47%**	**62,20%**
Modification within Base-NPs	**93,20%**	**92,59%**
Modification to NPs	**73,20%**	**75,49%**
Sentential head	**94,99%**	**94,99%**
Adjunct to a verb	**75,11%**	**78,44%**

PARALLEL PARSING WITH CYK - MOTIVATIONS

State-of-the-art-analysis

Weighted context-free grammar (CFG), that is learned from a treebank

Complexity O(|*G*|n^3), the grammar constant |G| typically dominates the runtime

Thousands of nonterminal symbols and millions of context-free rules, $n \approx 20$

Number of processing cores doubles every 2nd year, clock frequency ≈ 3 GHz

Basic insights about parallelization

Over grammar rules rather than chart cells

Some parallelization options are architecture dependent

Understanding of programming model and hardware needed

Speed-up between 14 and 26 depending, depending on processor unit

CYK - SEQUENTIAL VERSION

```
Algorithm: parse(sen, lex, gr)
Input: sen /* the input sentence */
       lex /* the lexicon */
       gr /* the grammar */
Output: tree /* the most probable parse tree */

1      scores[][][] = initScores();
2      nW ords = readSentence(sen);
3      lexiconScores(scores, sen, nW ords, lex);
4      for length = 2 to nW ords
5            binaryRelax(scores, nW ords, length, gr);
6             unaryRelax(scores, nW ords, length, gr);
7      tree = backtrackBestParseTree(scores);
8      return tree;
```

```
Algorithm: binaryRelax(scores, nW ords, length, gr)
Input: scores /* the 3-dimensional scores */
       nW ords /* the number of total words */
       length /* the current span */
       gr /* the grammar */
Output: None
1     for start = 0 to nW ords - length
2           end = start + length;
3           foreach symbol ∈ gr
4                 max = FLOAT MIN;
5                 foreach rule r per symbol // defined by gr
6                       // r is "symbol ⇒ l sym r sym"
7                       for split = start + 1 to end ? 1
8                       // calculate score
9                       lscore = scores[start][split][l sym];
10                      rscore = scores[split][end][r sym];
11                      score = rule score + lscore + rscore;
12                      // maximum reduction
13                      if score > max
14                            max = score;
15                scores[start][end][symbol] = max;
```

USE OF SPECIAL HARDWARE

Hardware properties

Graphics Processor Units (GPUs), millions of operations executed in parallel

Processing cores called *stream processors* (SP), organized hierarchically

Compute Unified Device Architecture (CUDA) - for other applications

Single Instruction Multiple Threads (SIMT). executed in bundles (called *warps*)

A thread cannot advance to the next instruction if other threads

in the same warp have not yet completed their own execution

Hardware usage techniques

Inside a warp, if some threads follow different execution paths than others,

the execution of the threads with different paths is serialized - avoid it!

Global memory access is expensive, but shared data within the same block

CANDIDATES FOR PARALLELIZATION

Principled alternatives

Loop in the CYK top level inappropriate

All loops in BinaryRelax are good candidates, in principle

1. Symbols to threads (first loop in* BinaryRelax*)

Does not provide enough paralellism

load imbalance – each symbol has a varying number of rules associated with it

2. Rules to threads (nested loops lines 3 & 5 in* BinaryRelax*)

Avoids disadvantages of symbol mapping

Disadvantage - synchronization needed for rules with same parent symbol

3. Exploiting granularity in hardware (line 3 to thread blocks & line 5 to threads)

Each symbol to a thread block, rules associated to local threads

Solves all the disadvantages, no synchronization needed

MAPPING RULES TO THREADS

Algorithm: threadBasedRulesBR(scores, nW ords, length, gr)

```
…
1    for start = 0 to nW ords - length in parallel
2         end = start + length;
3         foreach rule r ∈ gr in parallel
4              shared ... int.sh.max[NUM.SYMBOL] = FLOAT MIN;


11                  // local maximum reduction
12                  if score > local.max
13                       local.max = score;
14             atomicMax(&sh max[symbol], local max);
15   // global maximum reduction
16   foreach symbol ∈ gr in parallel
17   atomicMax(&scores[start][end][symbol], sh max[symbol]);
```

MAPPING TO BLOCKS AND THREADS

Algorithm: blockBasedRulesBR(scores, nW ords, length, gr)

```
…
1    for start = 0 to nW ords - length in parallel
2        end = start + length;
3        foreach symbol ∈ gr in parallel
4            shared ... int.sh.max[NUM.SYMBOL] = FLOAT MIN;
5            foreach rule r per symbol in parallel



11               // local maximum reduction
12               if score > local.max
13                   local.max = score;
14           atomicMax(&sh max[symbol], local.max);
15   // global maximum reduction
16   foreach symbol ∈ gr in parallel
17       atomicMax(&scores[start][end][symbol], sh.max);
```

FURTHER COMPLEMENTARY MEASURES

1. Span level parallelism (first loop in* BinaryRelax*)

Spans in same level of the chart are independent of each other

2. Amomic operations (lines 14 & 17 in* threadBasedRulesBR*)

Creating shared variables, percolating down references

3. Amomic operations on shared memory (lines 15 & 18 in* blockBasedRulesBR*)

Only one shared variable per thread block

Requires only a fraction of shared memory,

costly operation on global memory performed only once

4. Reducing global memory access - adapt memory access pattern (right instead of left)

```
for split = start + 1 to end - 1
    lscore = scores[start][split][l sym];
    rscore = scores[split][end][r sym];
    score = rule score + lscore + rscore;
```

```
for k = 1 to len - 1
    lscore = sh scores L[k][unique l sym];
    rscore = sh scores R[k][unique r sym];
    score = rule score + lscore + rscore;
```

FURTHER EFFICIENCY-EMPHASIZING MEASURES

Agenda-based parsing

Best-first and A* variants, depending on admissibility of heuristics

Also local variations, to keep agenda managing effort low

Beam search parsing

Local pruning, based on learning and posterior probabilities

Course-to-fine parsing (multiple pass parsing)

First-step parsing with a course grammar, to obtain good parameters

Builfing a course grammar out of a fine one is not easy

Chart constraints

Skipping entire chart cells on the basis of trained tagging data (start or end)

CCG parsing

Considerably harder than CFG; supported by supertagging, A* methods

UNIFICATION-BASED PARSING

Motivation

Rule explosion for expressing features as subcategories (e.g., for agreement)

Categories are generalized into feature strucures

Modifications in processing

Make all elements of the grammar components feature structures

Substitute *unification* and *equivalence* tests for category comparison

***Unify* category of passive edges with *argument position* of active edges**

Test *spanning passive* edges for compatibility against start symbol *S*

UNIFICATION-BASED PARSING – EFFICIENCY

Motivation

90+% of parsing time typically go to directed acyclic graph manipulation

Observations

Most unifications fail: predict failure cheaply, where possible

- ***rule filter*: rule feeding relations**
- ***quick check*: most likely failure paths**

Lexicalization: argument positions in rules may be highly underpecified

- ***head driven*: instantiate right-hand side bidirectionally, starting from head**

Many unifications fail very early: *copy* more expensive than *unify*

- **memory is expensive: redo a couple of unifications instead of copying**

TECHNIQUES FOR EFFICIENT PARSING (I)

Pre-compiling the lexicon

Expansion and application of lexical rules done off-line

Parts of feature structures for internal use of lexical rules deleted

Loading a compiled file for data about each stem, caching most frequent ones

Improvements in unification

Destructive, but reversible check testing compatibility between structures

Output structure built only in case of success

Reusing parts of the input structure in building the output structure

Disjunctions re-expressed in the type hierarchy or in disjunctive normal forms

1.4 - 3 times more rules German VERBMOBIL

&

2-5 overall speed-up Japanese grammars

TECHNIQUES FOR EFFICIENT PARSING (II)

Pre-compiling type unification

50% of unification and copying time for computing greatest lower bound (GLB)

6,000 types in VERBMOBIL result in 36,000,000 possible GLBs

Only GLBs of the 0.5-2.0% successful unifications needs to be stored in tables

Computing a unique key for each combination, storing in a hash table

Off-line computation expensive (naively 50 hours)

Exploiting symmetry and hierarchical 'consistency' reduces this to 1 hour

Pre-compiling rule filters

Quick checking operations that avoid using unification

Filter realized as a three-dimensional boolean array (unary, binary schemas)

Off-line computation of rule filters < 1 minute for all three languages

Rule out 50-60% of failed unifications, saving 45% of parsing time

TECHNIQUES FOR EFFICIENT PARSING (III)

Dynamic unification filtering

Unification failure reasons unevenly distributed (CAT are frequent cases)

Quick check of most frequent failure points , stored as a feature path

Saving the paths with the highest failure rate in off-line parses

13 to 22 paths for the 3 languages, some very long and unintuitive

Avoids almost all unsuccessful unification, adds 75% savings to rule filtering

Reducing feature structure size via restrictors

Large (theoretically-motivated) structures without effect on searching

All relevant information in SYNSEM feature of mother node

Paths specifications to guide deletion (positive and negative restrictors)

Speed gain of 30% for German and 45% for English

TECHNIQUES FOR EFFICIENT PARSING (IV)

Limiting the number of initial chart items

Number of lexical items per stem may increase parsing hypotheses

Cooccurrence requirements of items for plausible readings motivate deletions

(e.g., prefixed verbs reuiqre a separable prefix, and vice-versa)

Global operation context (the whole chart) – realized by exclusive-or operations

Example: “Ich komme morgen an” – only 8 of 97 readings of “komme” remain

“Der Mann wartet an der Tür” only prepositional readings for “an”

Computing best partial analyses

Deficient or spontaneous input may not yield a successful parse

Best partial results are maintained and combined

Best paths are (sub)trees with utterance state, second best are lexical items

Combination of partial parses that overarches the full utterance (minimal costs)

RESULTS

Combined effect

Deep linguistic analysis coupled with speech processing

Overall speed-up by a factor of 10 to 25

	German	*English*	*Japanese*
# sentences	**5106**	**1261**	**1917**
# words	**7**	**6.7**	**7.2**
# lexical entries	**40.9**	**25.6**	**69.8**
# chart items	**1024**	**234**	**565**
# results	**5.8**	**12.4**	**53.6**
time first	**1.46 s**	**0.24s**	**0.9 s**
time overall	**4.53 s**	**1.38 s**	**4.42 s**

REDUCING CONTEXT-FREE PARSING BY CONSTRAINTS (Roark, Hollingshead, Bodenstab)

Motivation

Many options of compositions in a chart "linguistically/lexically implausible"

Constraints about prominent positions of words found out cheaply

General approach

Evidence about start and end of a constituent (as possible positions) crucial

Incompatibilities of words with these positions derived in fast pre-processing

Incompatibilities used to cut-off options in chart processing

Considerable savings can be obtained / proofs on complexity bounds

Demonstrated for rather differently structured languages (English, Chinese)

RESULTS FOR CONSTRAINS ON WORDS

Table 1
Statistics on extracted word classes for English (Sections 2–21 of the Penn WSJ treebank) and Chinese (articles 1–270 and 400–1151 of the Penn Chinese treebank).

	Corpus totals		Begin class		End class		Unary class	
	Strings	Words	B	$\overline{B}$	E	$\overline{E}$	U	$\overline{U}$
English								
Count	39,832	950,028	430,841	439,558	223,544	646,855	105,973	844,055
Percent			49.5	50.5	25.7	74.3	11.2	88.8
Chinese								
Count	18,086	493,708	188,612	269,000	165,591	292,021	196,732	296,976
Percent			41.2	58.8	36.2	63.8	39.9	60.1

BASIC SEARCH RESULTS FOR ENGLISH

Table 5
English test set results (WSJ Section 23) for the CONSTRAINEDCYK algorithm with both left- and right-binarized Markov order-2 grammars.

	Constraints	F_1	Precision	Recall	Seconds	Speed-up
Right	None (baseline CYK)	71.7	74.6	69.0	628	
	Unary(40)	72.1	75.5	69.0	525	1.2x
	GHP(40)	75.8	79.0	72.8	75	8.4x
	GHP(40) + Unary(40)	76.1	79.8	72.7	57	11.0x
Left	None (baseline CYK)	72.0	74.8	69.4	1,063	
	Unary(40)	72.4	75.8	69.4	910	1.2x
	GHP(40)	76.1	79.3	73.2	106	10.1x
	GHP(40) + Unary(40)	76.4	80.0	73.1	60	17.9x

BASIC SEARCH RESULTS FOR CHINESE

Table 6
Chinese test set results (PCTB Sections 271–300) for the CONSTRAINEDCYK algorithm with both left- and right-binarized Markov order-2 grammars.

	Constraints	F_1	Precision	Recall	Seconds	Speed-up
Right	None (baseline CYK)	60.5	64.6	56.9	157	
	Unary(20)	62.3	67.6	57.8	141	1.1x
	GHP(20)	66.9	71.4	63.0	17	9.1x
	GHP(20) + Unary(20)	68.9	74.4	64.1	15	10.2x
Left	None (baseline CYK)	60.4	64.5	56.9	269	
	Unary(20)	62.1	67.2	57.8	234	1.2x
	GHP(20)	66.0	70.5	62.1	33	8.2x
	GHP(20) + Unary(20)	68.0	73.5	63.2	23	11.7x

COMPETITIVE TEST RESULTS FOR ENGLISH

Table 7
English test set results (WSJ Section 23) applying sentence-level high precision and unary constraints to three parsers with parameter settings tuned on development data.

Parser	F_1	Precision	Recall	Seconds	Speed-up
BUBS (2010)	88.4	88.5	88.3	586	
+ Unary(100)	88.5	88.7	88.3	486	1.2x
+ HP(0.9)	88.7	88.9	88.6	349	1.7x
+ HP(0.9) + Unary(100)	88.7	89.0	88.4	283	2.1x
Charniak (2000)	89.7	89.7	89.6	1,116	
+ Unary(100)	89.8	89.8	89.7	900	1.2x
+ HP(0.8)	89.8	90.0	89.6	716	1.6x
+ HP(0.8) + Unary(100)	89.7	90.0	89.5	679	1.6x
Berkeley (2007)	90.2	90.3	90.0	564	
+ Unary(125)	90.1	90.3	89.9	495	1.1x
+ HP(0.7)	90.2	90.4	90.0	320	1.8x
+ HP(0.7) + Unary(125)	90.2	90.4	89.9	289	2.0x

COMPETITIVE TEST RESULTS FOR CHINESE

Table 8
Chinese test set results (PCTB articles 271–300) applying sentence-level high-precision and unary constraints to two parsers with parameter settings tuned on development data.

Parser	F_1	Precision	Recall	Seconds	Speed-up
BUBS (2010)	79.5	79.5	79.1	169	
+ Unary(50)	80.7	82.1	79.4	153	1.1x
+ HP(0.8)	81.1	81.5	80.7	75	2.3x
+ HP(0.8) + Unary(50)	81.8	83.1	80.5	44	3.8x
Berkeley (2007)	83.9	84.5	83.3	141	
+ Unary(50)	84.5	85.9	83.0	125	1.1x
+ HP(0.7)	84.5	85.1	83.8	64	2.2x
+ HP(0.7) + Unary(50)	84.7	86.1	83.4	57	2.5x