Forest Rescoring

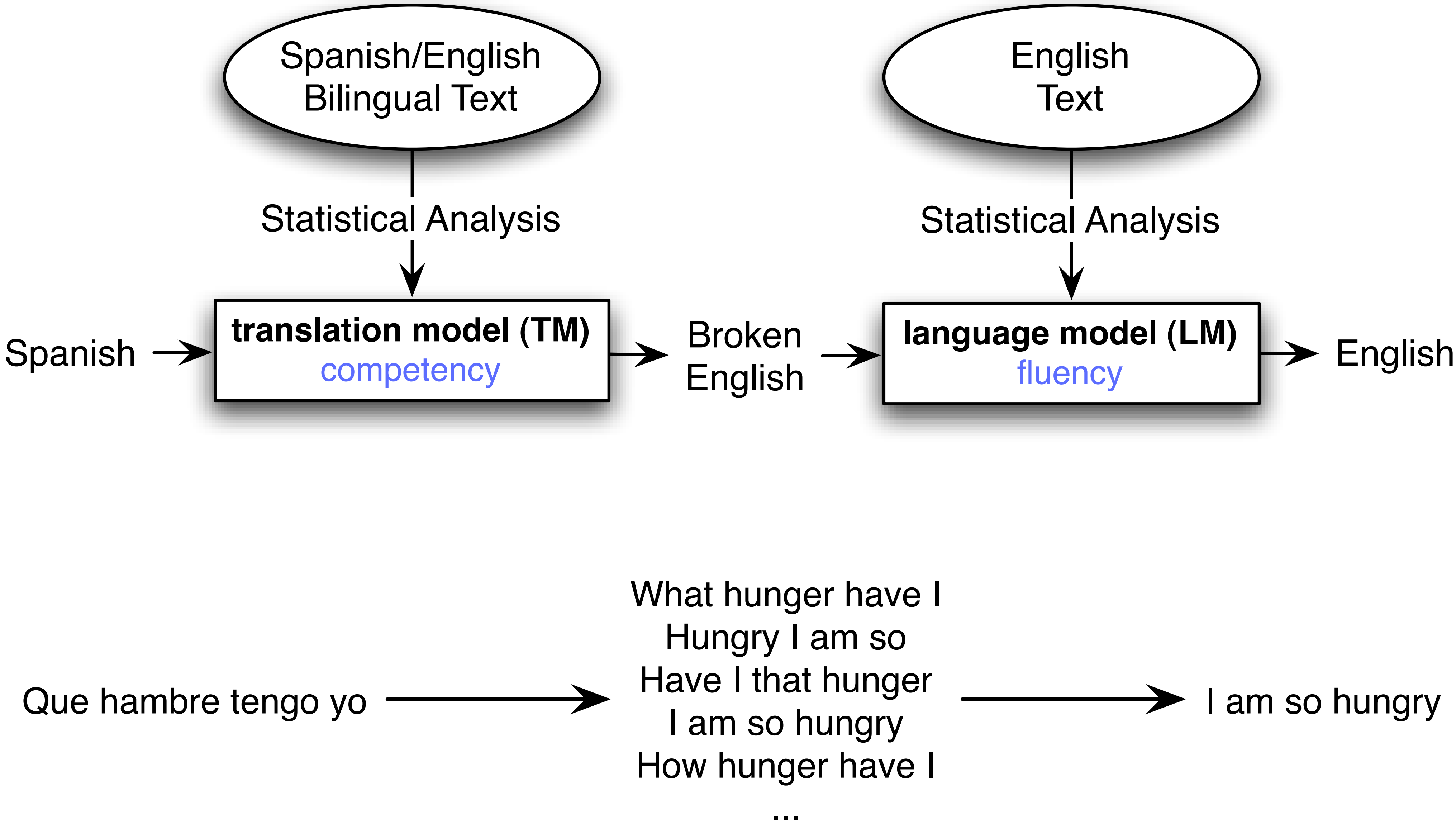
Faster Decoding with Integrated Language Models

Liang Huang
Penn University of Pennsylvania

David Chiang
ISI Information Sciences Institute USC School of Engineering

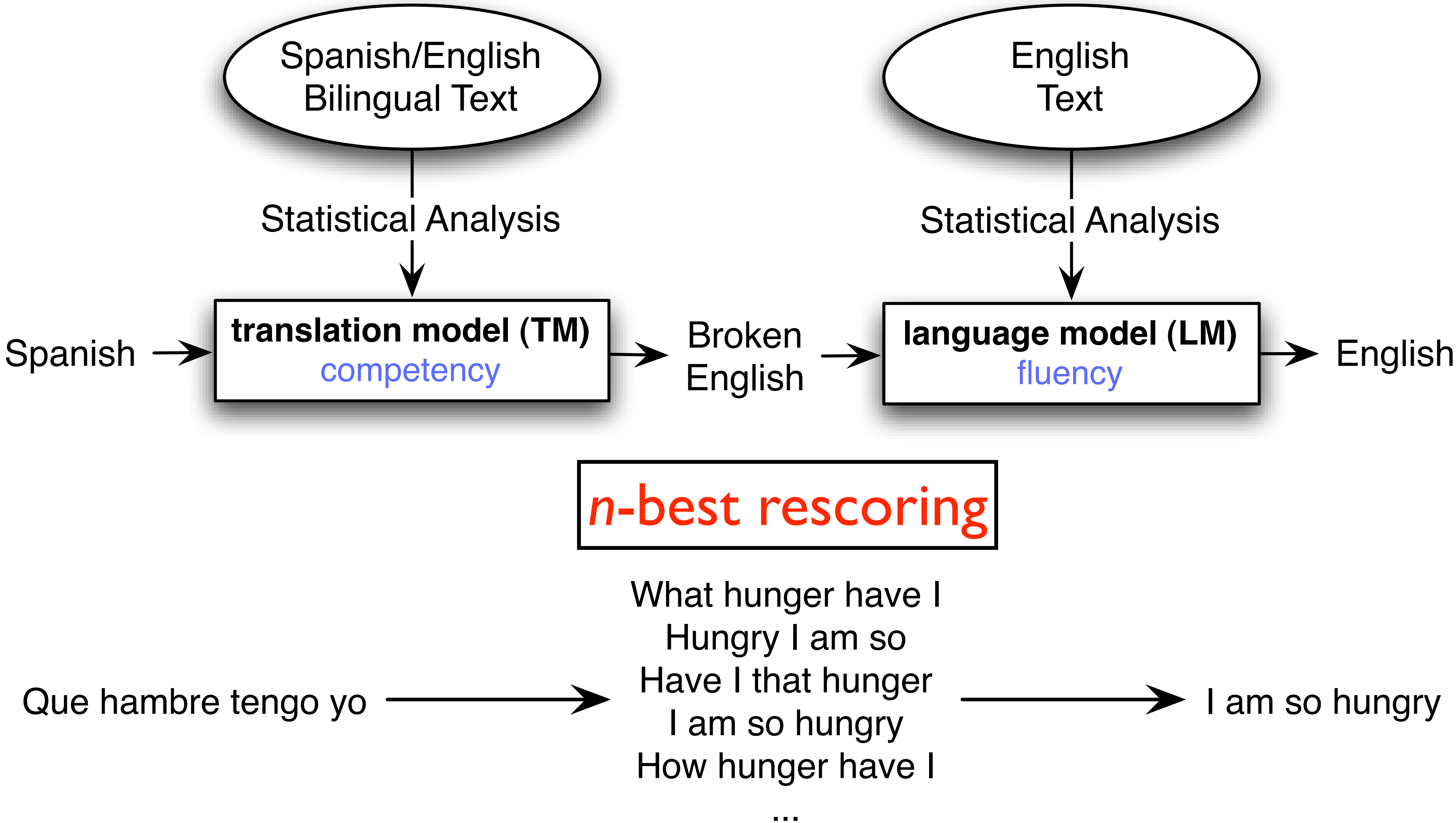
ACL 2007, Praha, Česká republika

Statistical Machine Translation



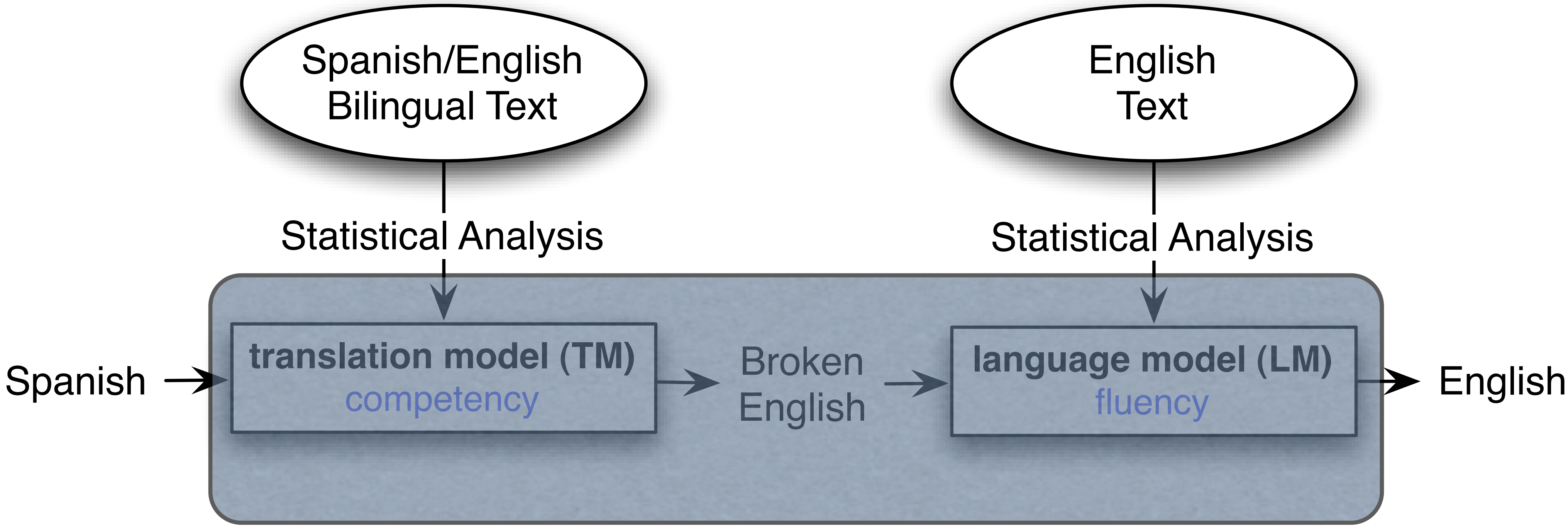
(Knight and Koehn, 2003)

Statistical Machine Translation

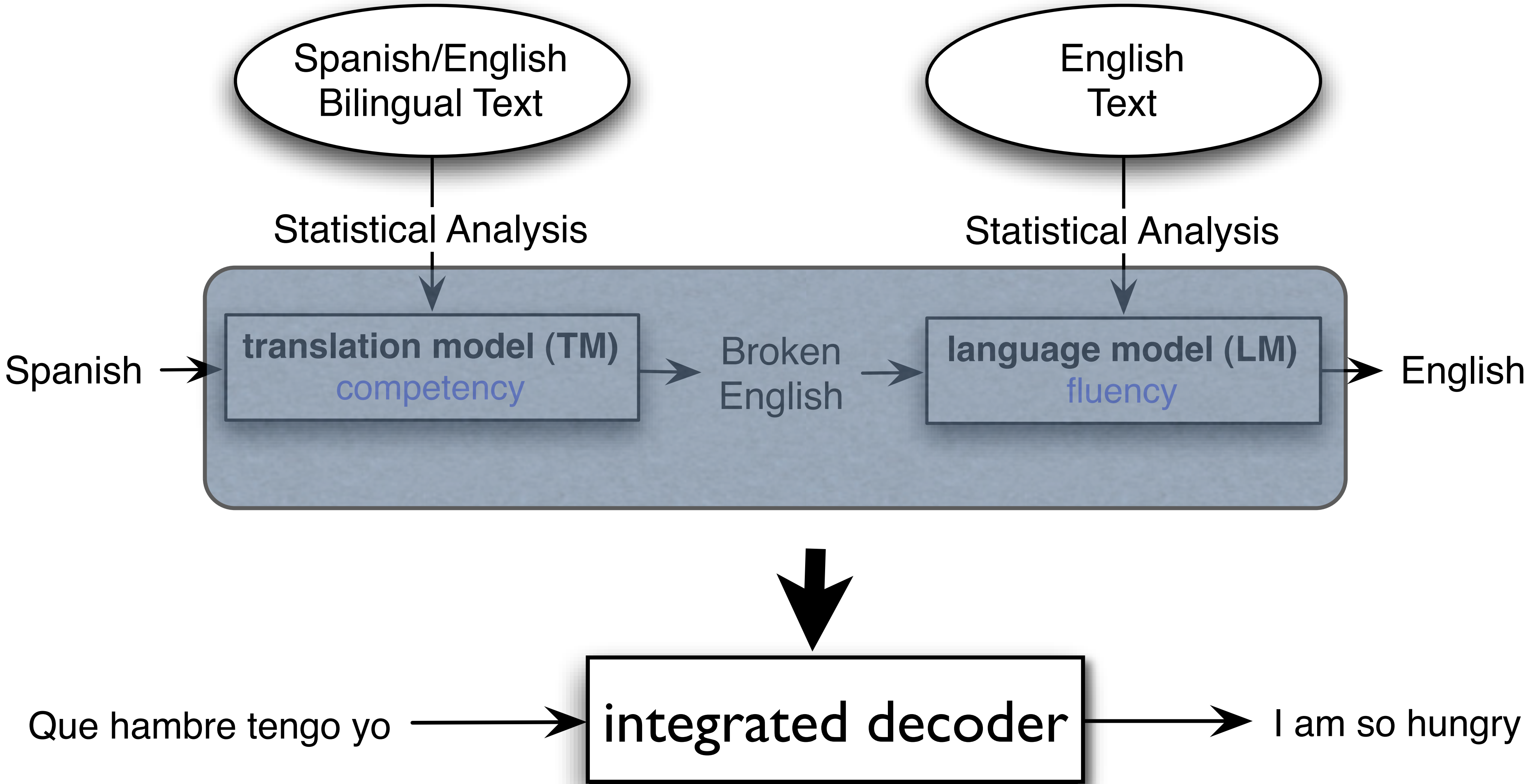


(Knight and Koehn, 2003)

Statistical Machine Translation

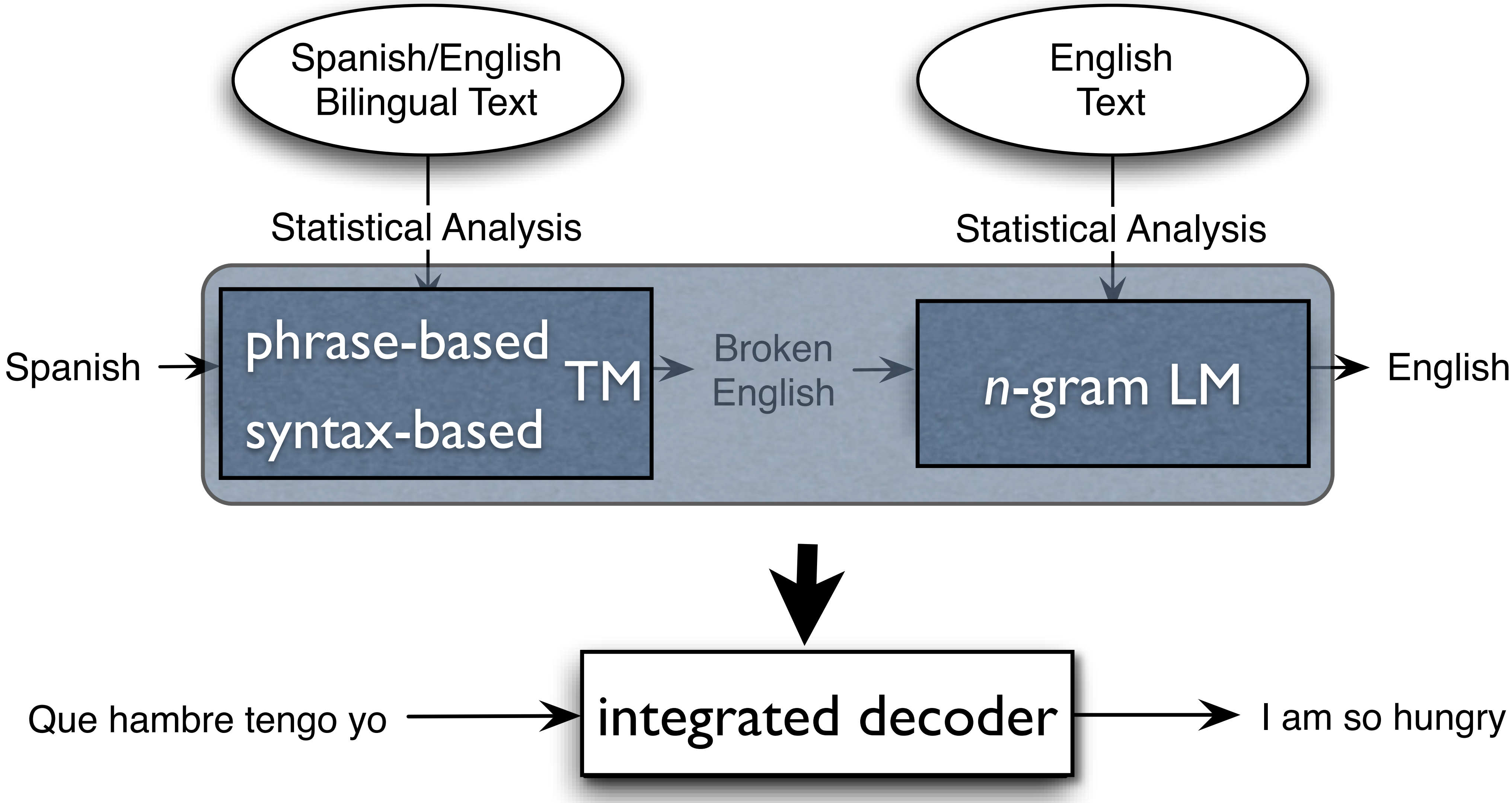


Statistical Machine Translation



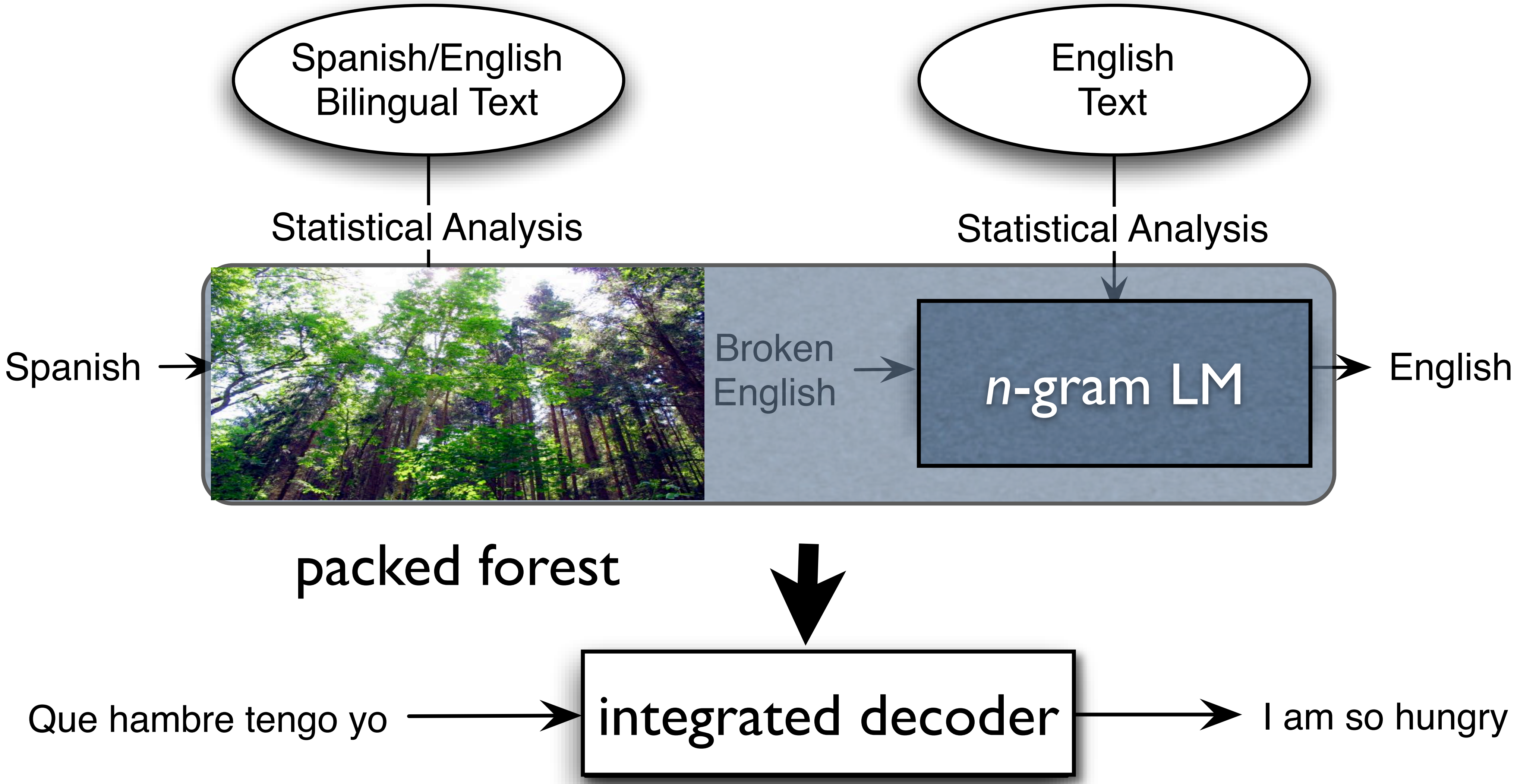
computationally challenging! ☹

Statistical Machine Translation



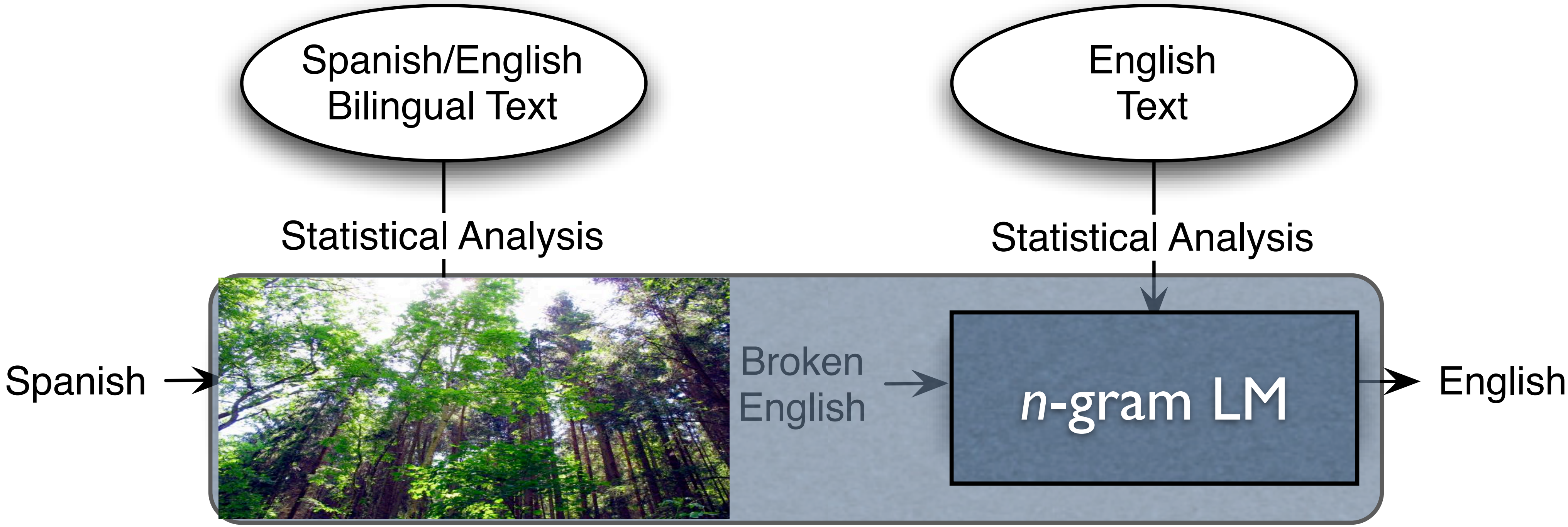
computationally challenging! ☹

Forest Rescoring

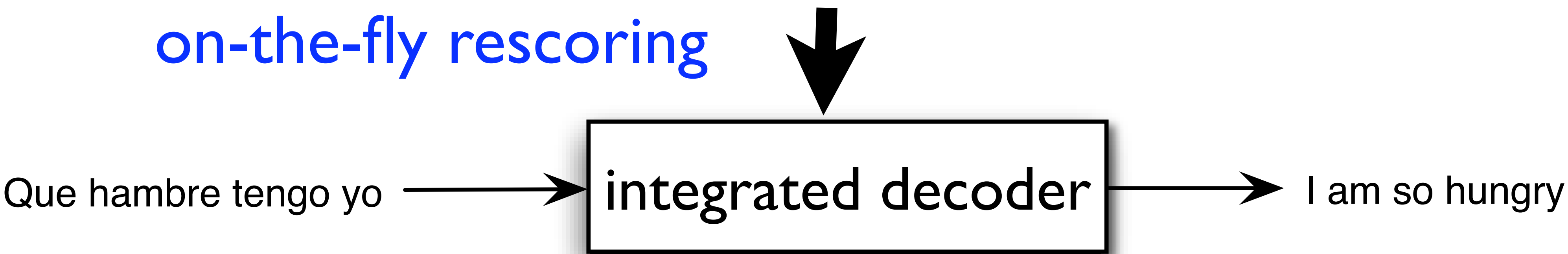


computationally challenging! ☹

Forest Rescoring

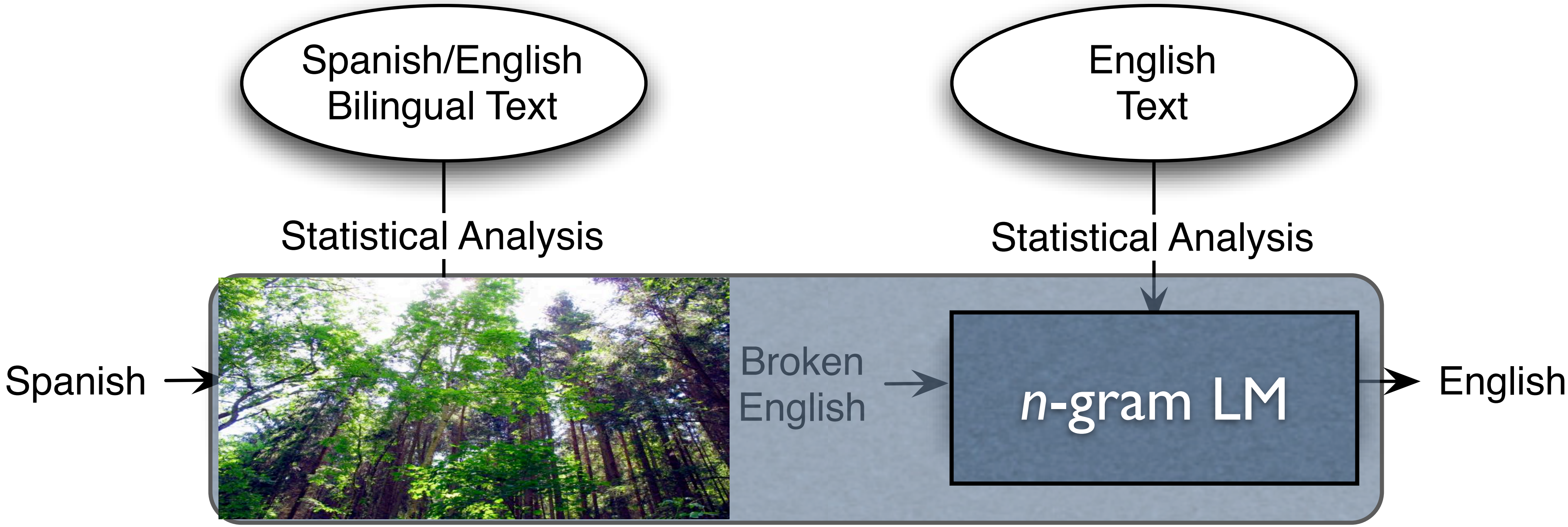


on-the-fly rescoring



computationally challenging! ☹

Forest Rescoring



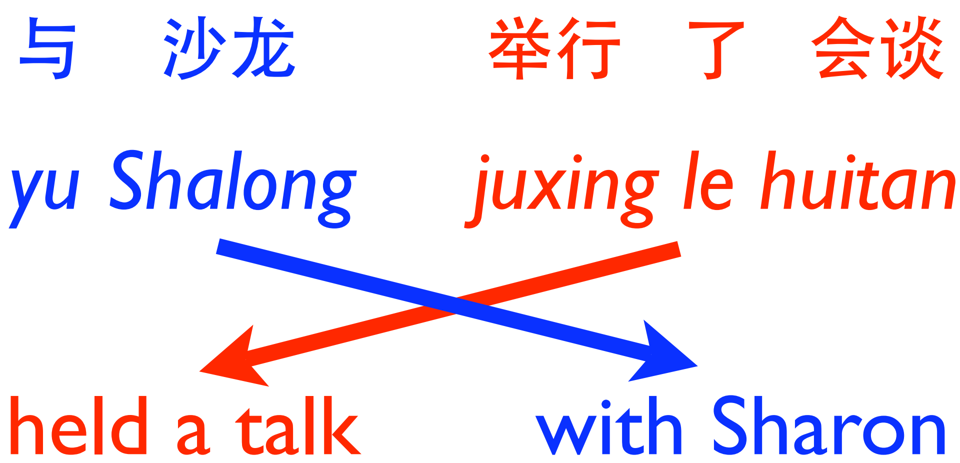
on-the-fly rescoring

significant speed-up: 10~30 times faster! ☺

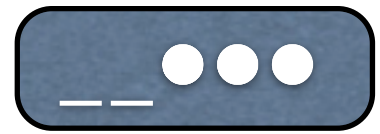
The Forest Framework

unifying phrase- and syntax-based decoding

Phrase-based Decoding

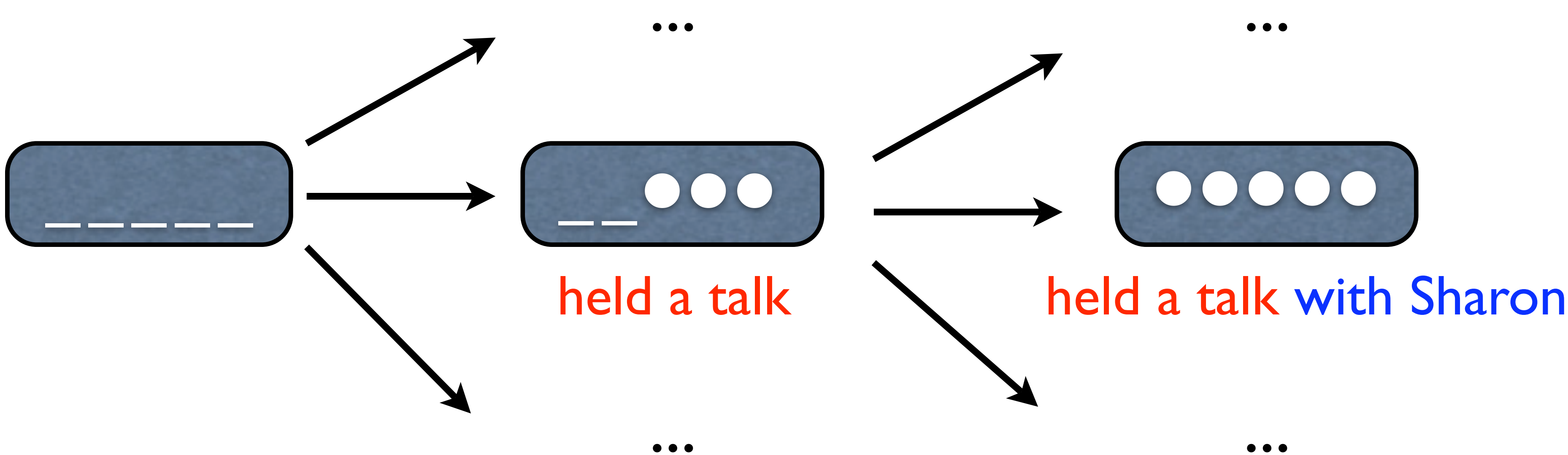


source-side: coverage vector



held a talk

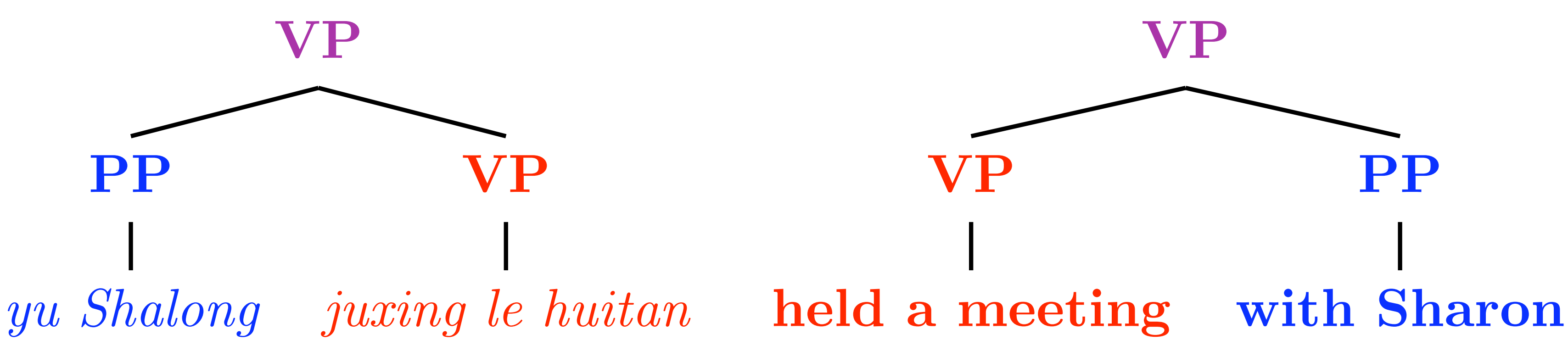
target-side: grow hypotheses strictly left-to-right



Syntax-based Translation

- synchronous context-free grammars (SCFGs)
  - context-free grammar in two dimensions
  - generating pairs of strings/trees simultaneously
  - co-indexed nonterminal further rewritten as a unit

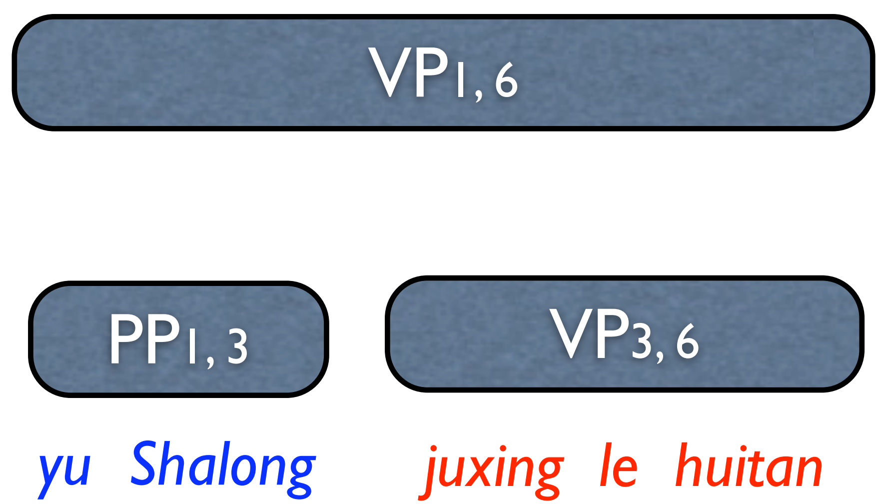
$$
\begin{array}{lcll}
\mathbf{VP} & \rightarrow & \mathbf{PP}^{(1)}\ \mathbf{VP}^{(2)}, & \mathbf{VP}^{(2)}\ \mathbf{PP}^{(1)} \\
\mathbf{VP} & \rightarrow & \textit{juxing le huitan}, & \textbf{held a meeting} \\
\mathbf{PP} & \rightarrow & \textit{yu Shalong}, & \textbf{with Sharon}
\end{array}
$$



Translation as Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
  - build the corresponding target sub-strings in parallel

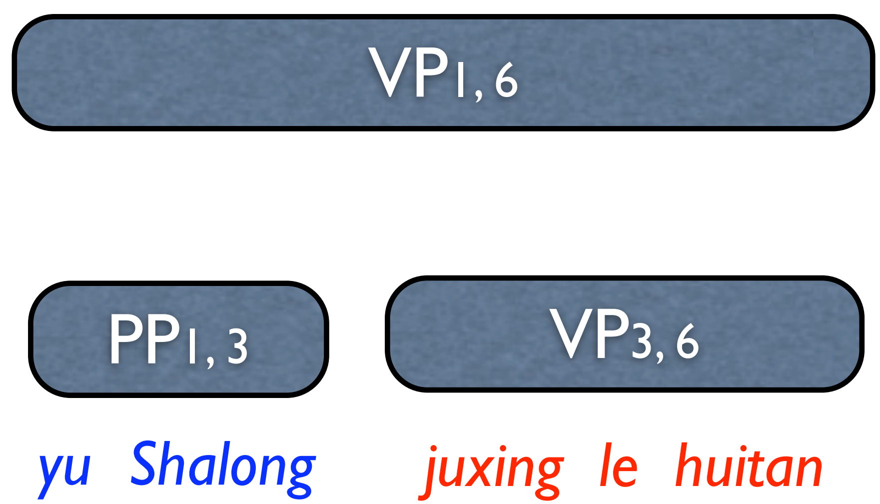
$$
\begin{array}{lcl}
\mathbf{VP} & \rightarrow & \mathbf{PP}^{(1)}\ \mathbf{VP}^{(2)}, \\
\mathbf{VP} & \rightarrow & \textit{juxing le huitan}, \\
\mathbf{PP} & \rightarrow & \textit{yu Shalong},
\end{array}
$$



Translation as Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
  - build the corresponding target sub-strings in parallel

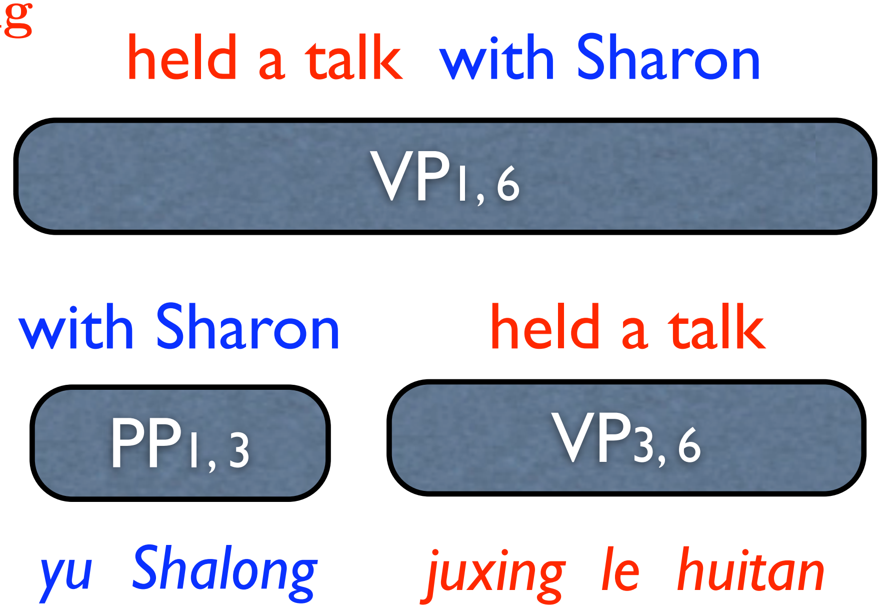
$$
\begin{array}{lcll}
\mathbf{VP} & \rightarrow & \mathbf{PP}^{(1)}\ \mathbf{VP}^{(2)}, & \mathbf{VP}^{(2)}\ \mathbf{PP}^{(1)} \\
\mathbf{VP} & \rightarrow & \textit{juxing le huitan}, & \textbf{held a meeting} \\
\mathbf{PP} & \rightarrow & \textit{yu Shalong}, & \textbf{with Sharon}
\end{array}
$$



Translation as Parsing

- translation with SCFGs => monolingual parsing
- parse the source input with the source projection
  - build the corresponding target sub-strings in parallel

$$\begin{array}{lcll}
\mathbf{VP} & \rightarrow & \mathbf{PP}^{(1)}\ \mathbf{VP}^{(2)}, & \mathbf{VP}^{(2)}\ \mathbf{PP}^{(1)} \\
\mathbf{VP} & \rightarrow & \textit{juxing le huitan}, & \textbf{held a meeting} \\
\mathbf{PP} & \rightarrow & \textit{yu Shalong}, & \textbf{with Sharon}
\end{array}$$

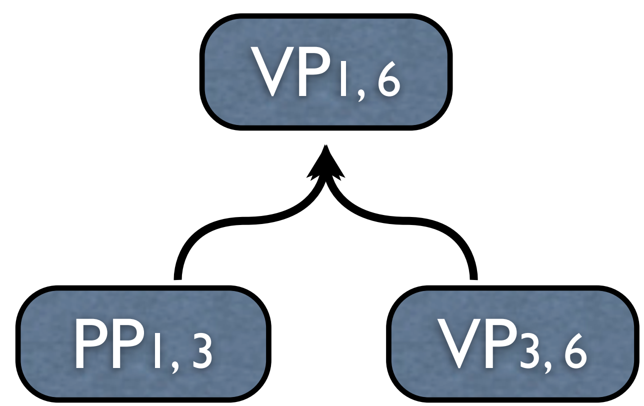


Packed Forest

- a compact representation of all translations
- has a structure of hypergraph (graph is a special case)

phrase-based: graph

syntax-based: hypergraph

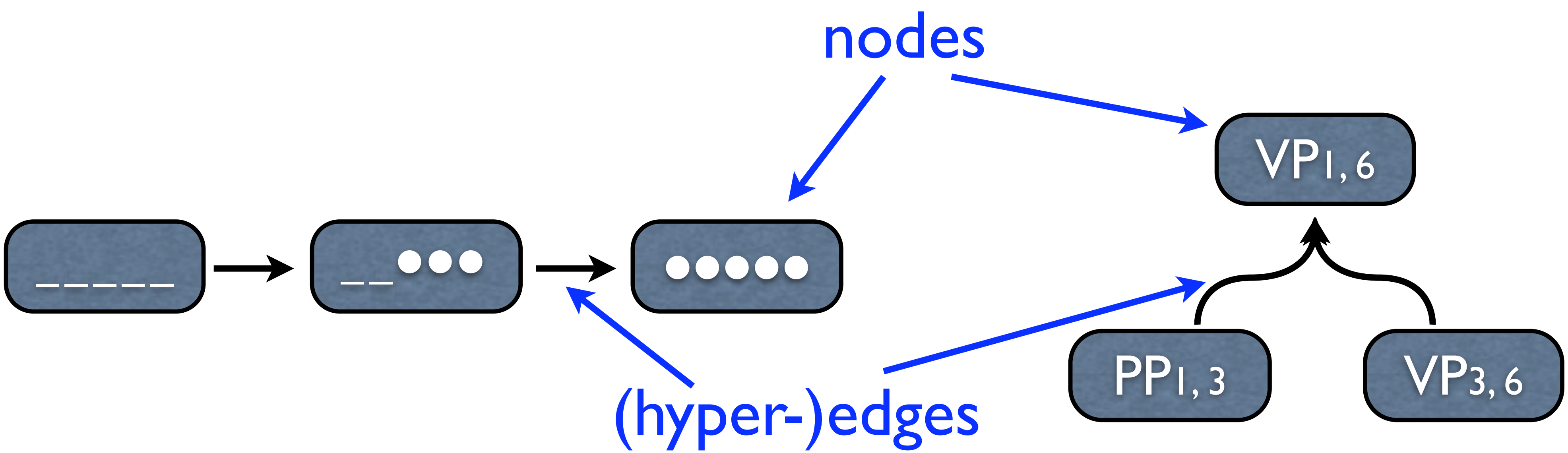


Packed Forest

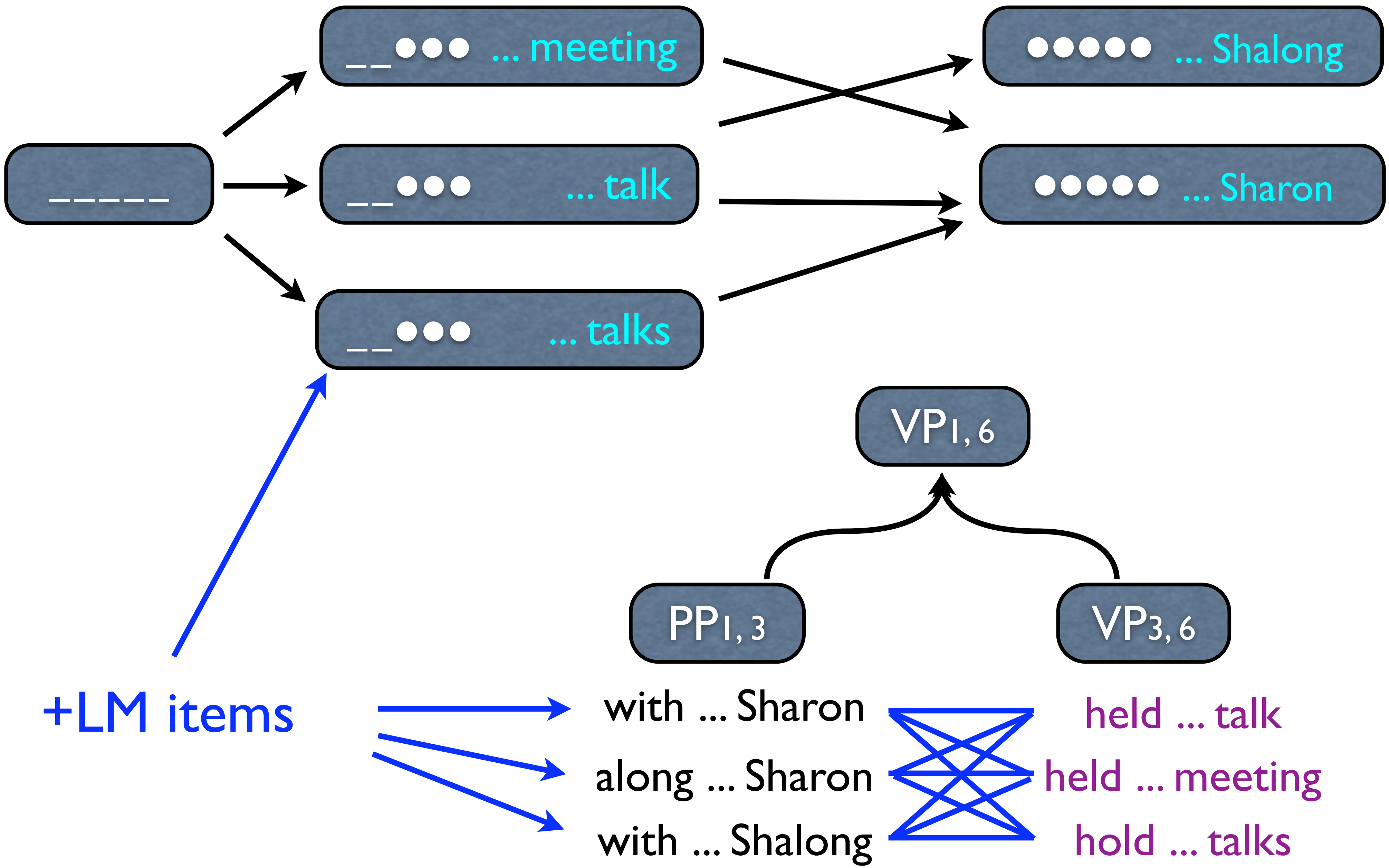
- a compact representation of all translations
- has a structure of hypergraph (graph is a special case)

phrase-based: graph

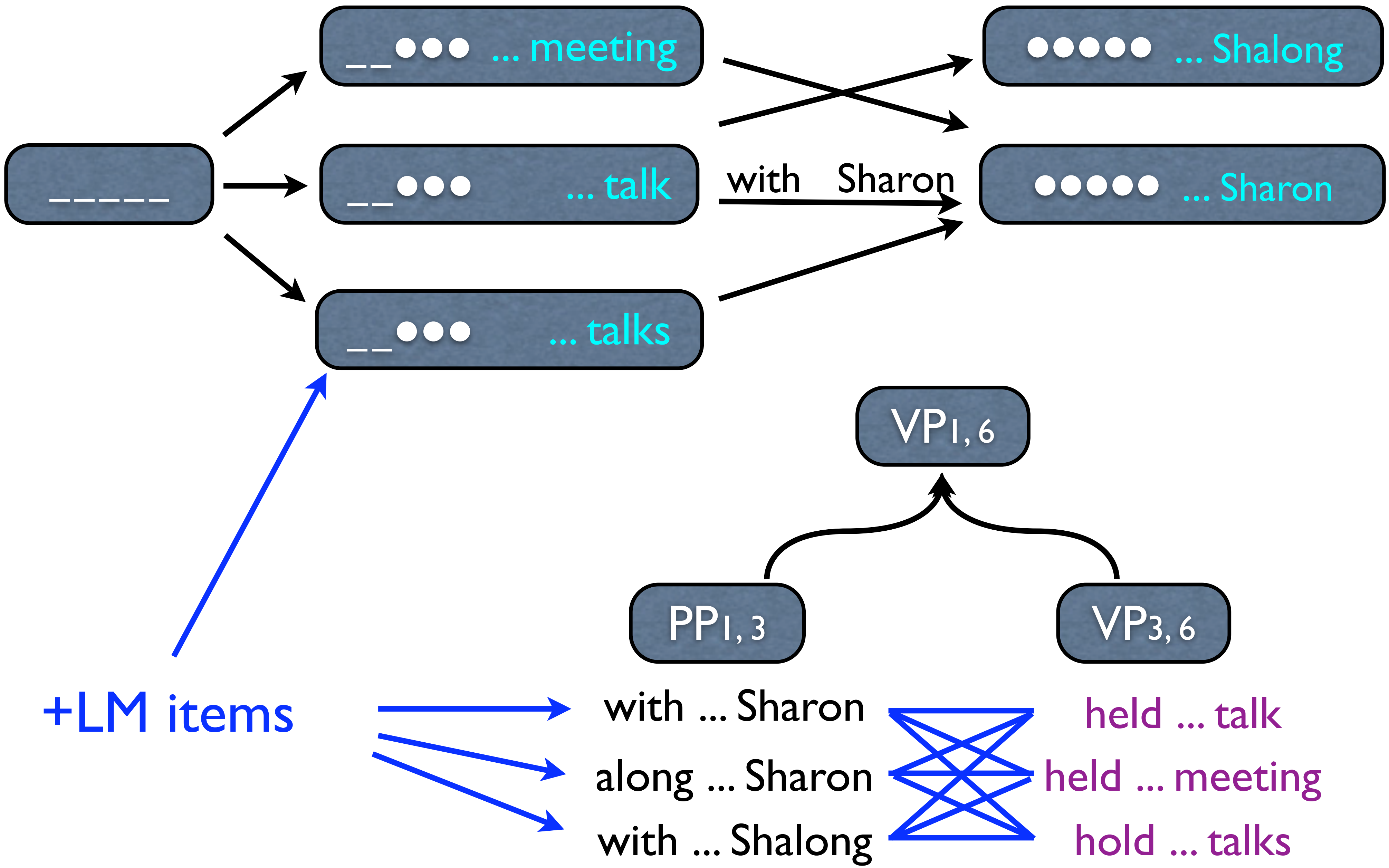
syntax-based: hypergraph



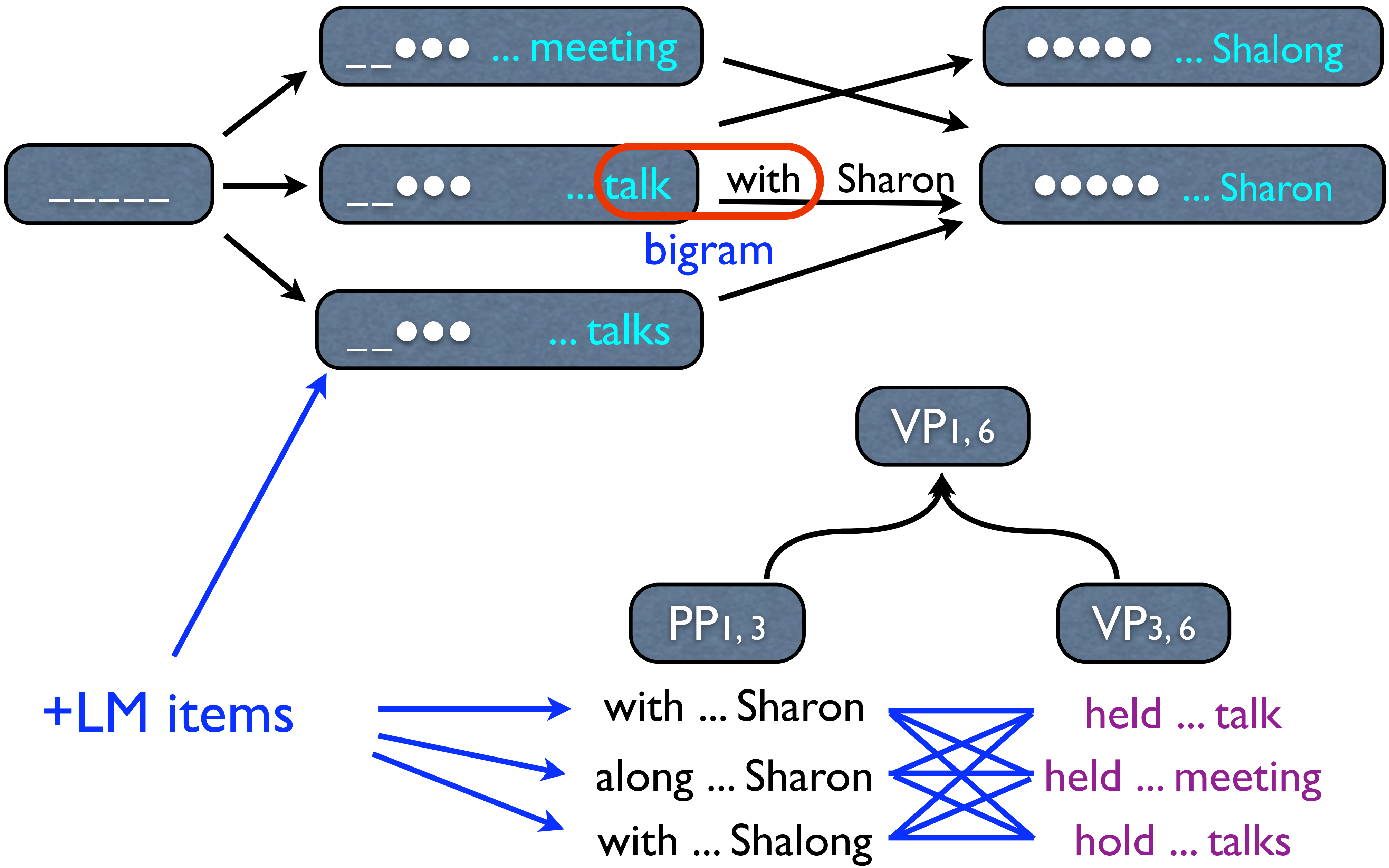
Adding a Bigram Model



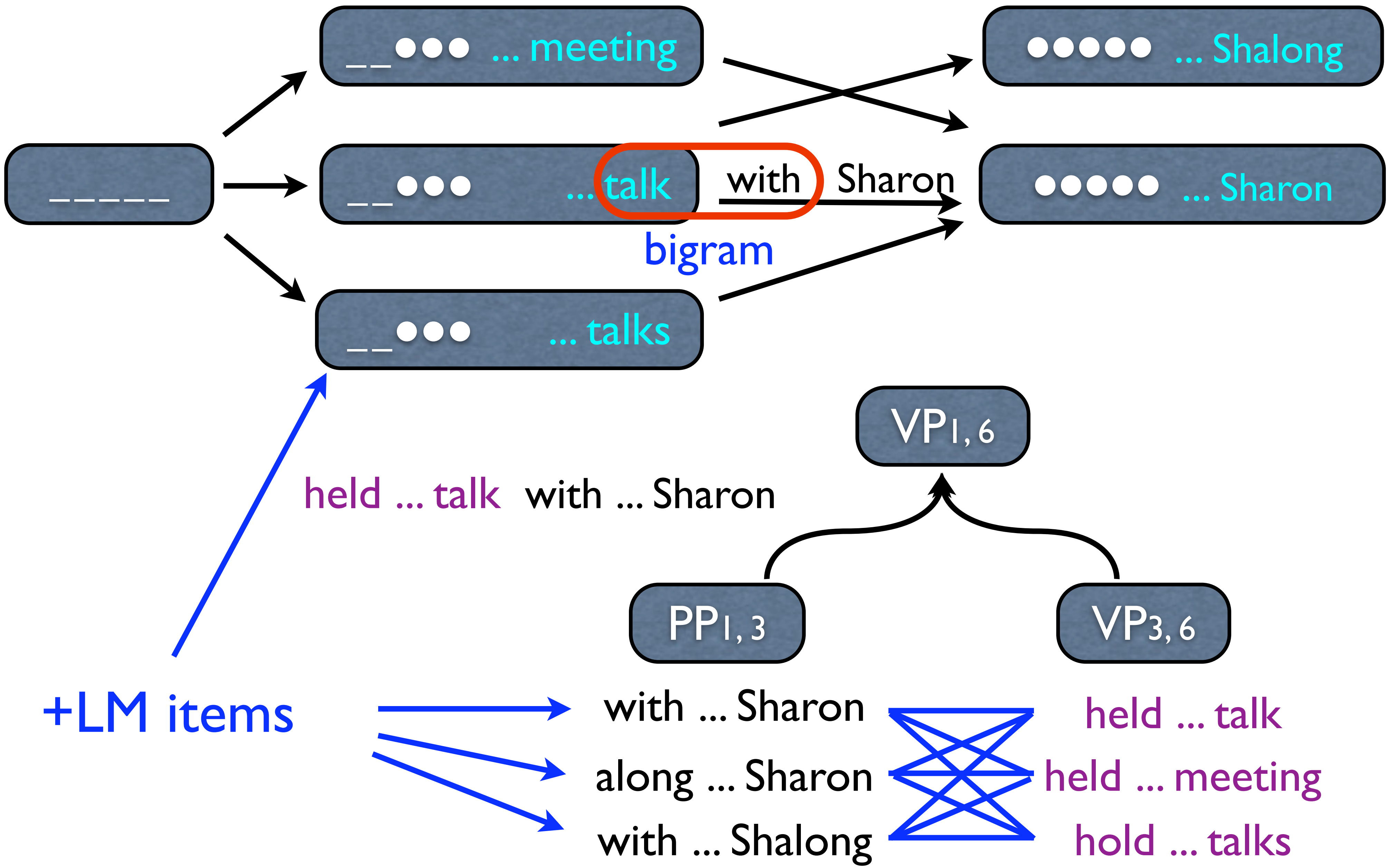
Adding a Bigram Model



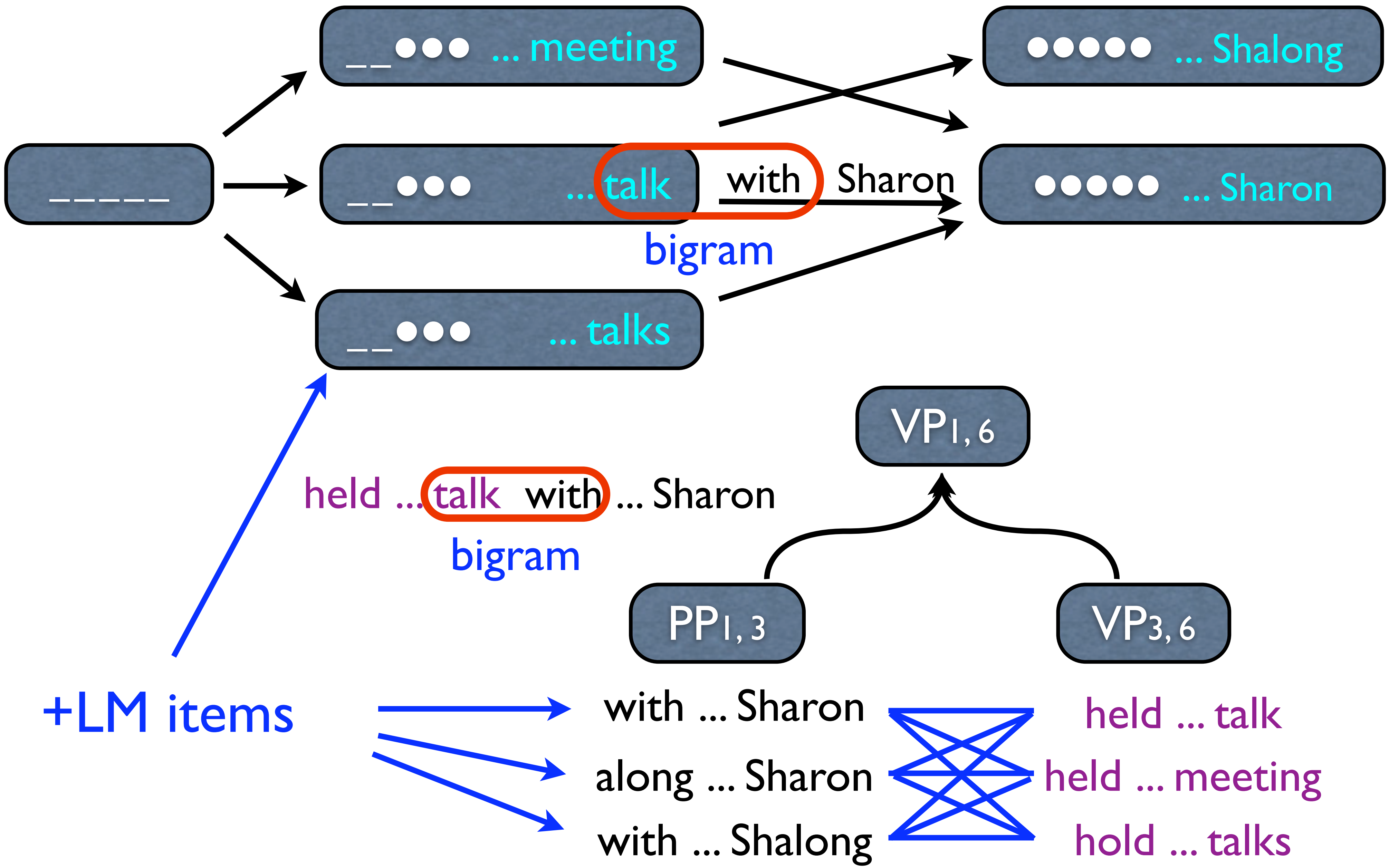
Adding a Bigram Model



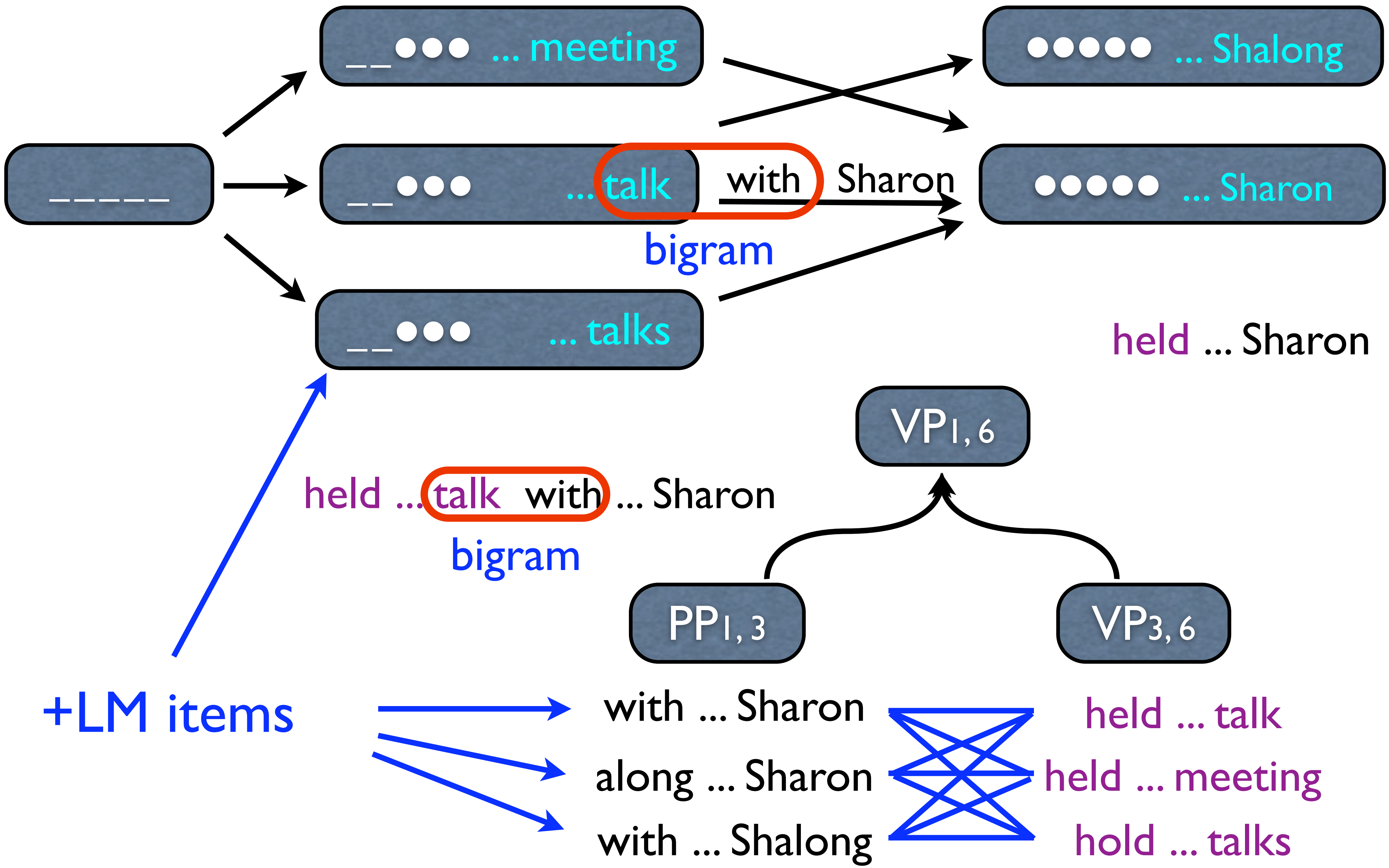
Adding a Bigram Model



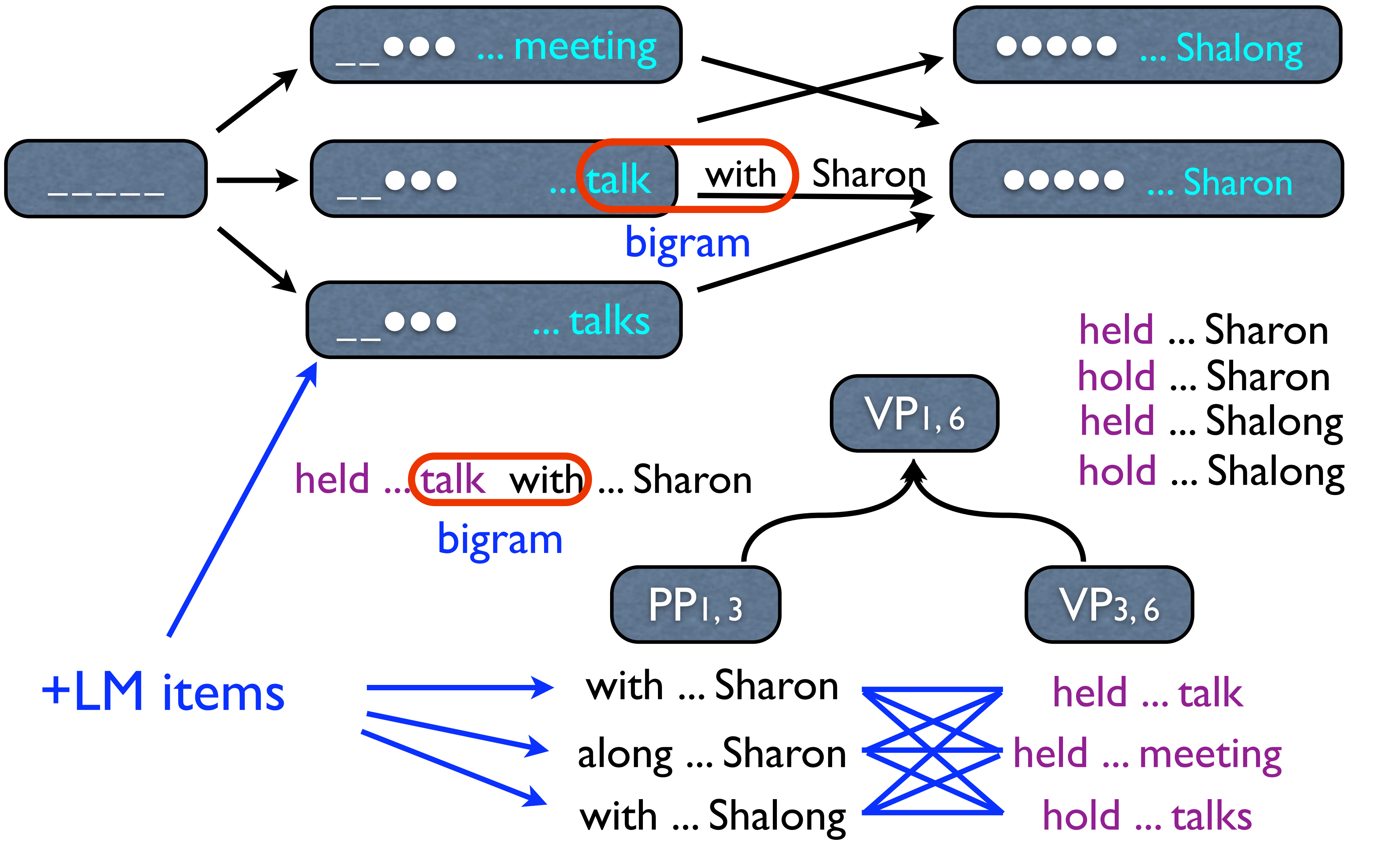
Adding a Bigram Model



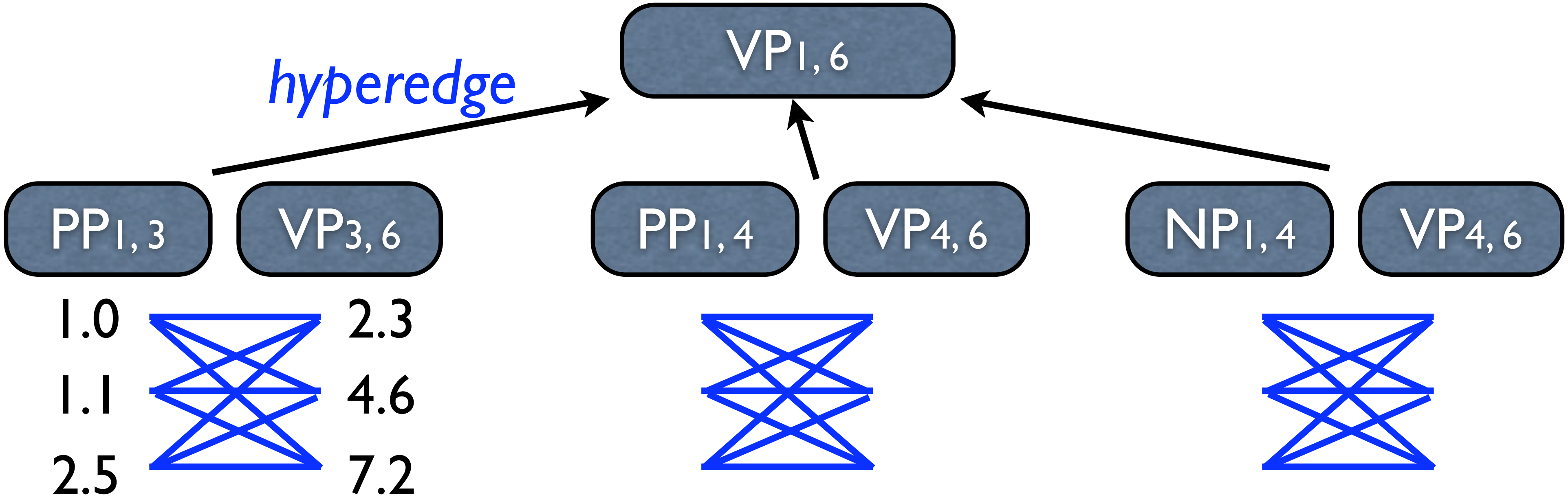
Adding a Bigram Model



Adding a Bigram Model

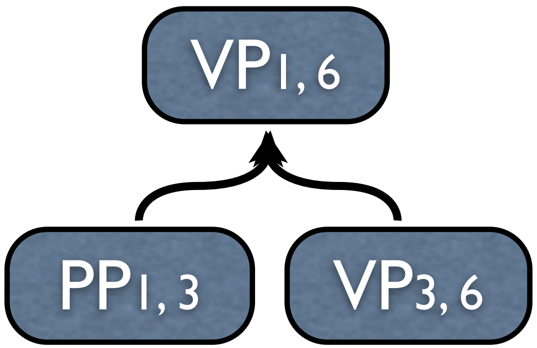


Conventional Beam Search



- beam search: only keep top-*k* +LM items at each node
- but there are many ways to derive each node
- can we avoid enumerating all combinations?
  - best-first enumeration?

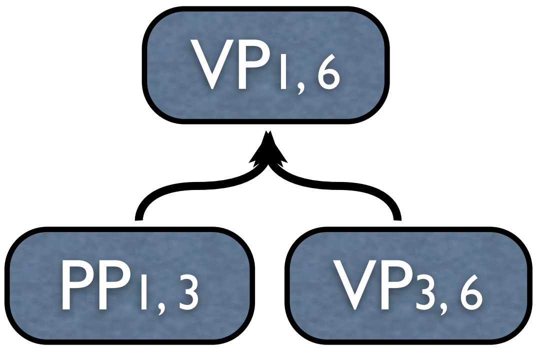
Cube Pruning



monotonic grid?

		$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
		1.0	3.0	8.0
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{meeting}})$	1.0	2.0	4.0	9.0
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{talk}})$	1.1	2.1	4.1	9.1
$(\mathrm{VP}_{3,6}^{\text{hold} \star \text{conference}})$	3.5	4.5	6.5	11.5

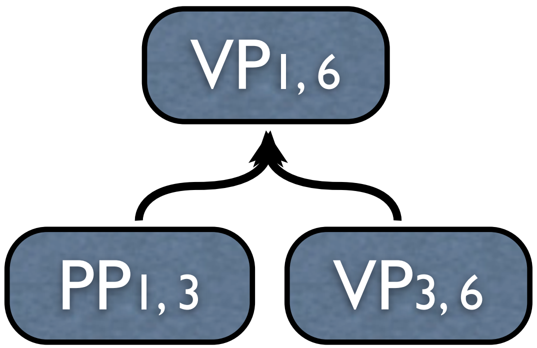
Cube Pruning



non-monotonic grid
due to LM combo costs

		$(\mathrm{PP}\,^{\text{with} \,\star\, \text{Sharon}}_{1,3})$	$(\mathrm{PP}\,^{\text{along} \,\star\, \text{Sharon}}_{1,3})$	$(\mathrm{PP}\,^{\text{with} \,\star\, \text{Shalong}}_{1,3})$
		1.0	3.0	8.0
$(\mathrm{VP}\,^{\text{held} \,\star\, \text{meeting}}_{3,6})$	1.0	2.0 + 0.5	4.0 + 5.0	9.0 + 0.5
$(\mathrm{VP}\,^{\text{held} \,\star\, \text{talk}}_{3,6})$	1.1	2.1 + 0.3	4.1 + 5.4	9.1 + 0.3
$(\mathrm{VP}\,^{\text{hold} \,\star\, \text{conference}}_{3,6})$	3.5	4.5 + 0.6	6.5 +10.5	11.5 + 0.6

Cube Pruning

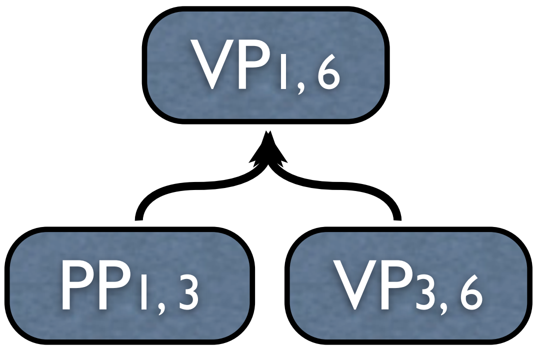


bigram (meeting, with)

non-monotonic grid
due to LM combo costs

	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
	1.0	3.0	8.0
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.0 + 0.5	4.0 + 5.0	9.0 + 0.5
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.1 + 0.3	4.1 + 5.4	9.1 + 0.3
$(\mathrm{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	4.5 + 0.6	6.5 +10.5	11.5 + 0.6

Cube Pruning



non-monotonic grid
due to LM combo costs

	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
	1.0	3.0	8.0
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.5	9.0	9.5
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.4	9.5	9.4
$(\mathrm{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate

		$(\mathrm{PP}^{\text{with} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{along} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{with} \star \text{Shalong}}_{1,3})$
		1.0	3.0	8.0
$(\mathrm{VP}^{\text{held} \star \text{meeting}}_{3,6})$	1.0	2.5	9.0	9.5
$(\mathrm{VP}^{\text{held} \star \text{talk}}_{3,6})$	1.1	2.4	9.5	9.4
$(\mathrm{VP}^{\text{hold} \star \text{conference}}_{3,6})$	3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate
- push the two successors

		$(\mathrm{PP}^{\text{with} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{along} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{with} \star \text{Shalong}}_{1,3})$
		1.0	3.0	8.0
$(\mathrm{VP}^{\text{held} \star \text{meeting}}_{3,6})$	1.0	2.5	9.0	9.5
$(\mathrm{VP}^{\text{held} \star \text{talk}}_{3,6})$	1.1	2.4	9.5	9.4
$(\mathrm{VP}^{\text{hold} \star \text{conference}}_{3,6})$	3.5	5.1	17.0	12.1

Cube Pruning

k-best parsing
(Huang and Chiang, 2005)

- a priority queue of candidates
- extract the best candidate
- push the two successors

	$(\mathrm{PP}^{\text{with} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{along} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{with} \star \text{Shalong}}_{1,3})$
	1.0	3.0	8.0
$(\mathrm{VP}^{\text{held} \star \text{meeting}}_{3,6})$ 1.0	2.5	9.0	9.5
$(\mathrm{VP}^{\text{held} \star \text{talk}}_{3,6})$ 1.1	2.4	9.5	9.4
$(\mathrm{VP}^{\text{hold} \star \text{conference}}_{3,6})$ 3.5	5.1	17.0	12.1

Cube Pruning

items are popped out-of-order

solution: keep a buffer of pop-ups

2.5 2.4 5.1

	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{along} \star \text{Sharon}})$	$(\mathrm{PP}_{1,3}^{\text{with} \star \text{Shalong}})$
	1.0	3.0	8.0
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{meeting}})$ 1.0	2.5	9.0	9.5
$(\mathrm{VP}_{3,6}^{\text{held} \star \text{talk}})$ 1.1	2.4	9.5	9.4
$(\mathrm{VP}_{3,6}^{\text{hold} \star \text{conference}})$ 3.5	5.1	17.0	12.1

Cube Pruning

items are popped out-of-order

solution: keep a buffer of pop-ups

2.5 2.4 5.1

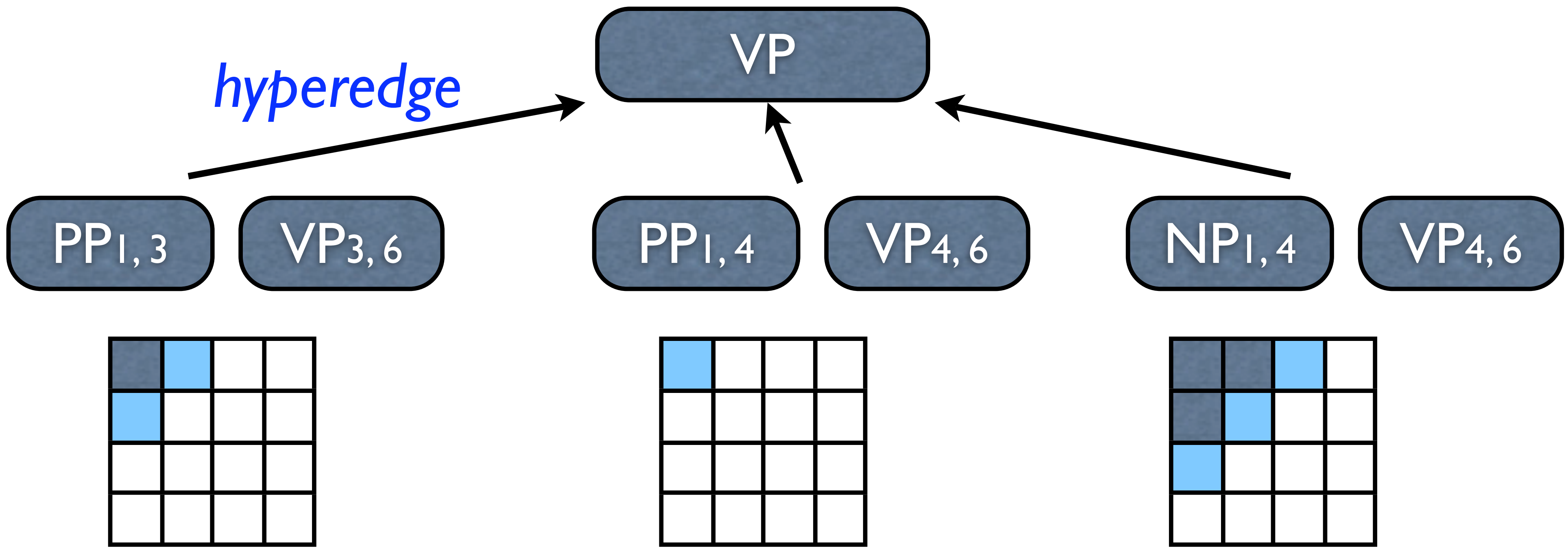
finally re-sort the buffer and return inorder:

2.4 2.5 5.1

	$(\mathrm{PP}^{\text{with} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{along} \star \text{Sharon}}_{1,3})$	$(\mathrm{PP}^{\text{with} \star \text{Shalong}}_{1,3})$
	1.0	3.0	8.0
$(\mathrm{VP}^{\text{held} \star \text{meeting}}_{3,6})$ 1.0	2.5	9.0	9.5
$(\mathrm{VP}^{\text{held} \star \text{talk}}_{3,6})$ 1.1	2.4	9.5	9.4
$(\mathrm{VP}^{\text{hold} \star \text{conference}}_{3,6})$ 3.5	5.1	17.0	12.1

Across Hyperedges

k-best parsing
(Huang and Chiang, 2005)

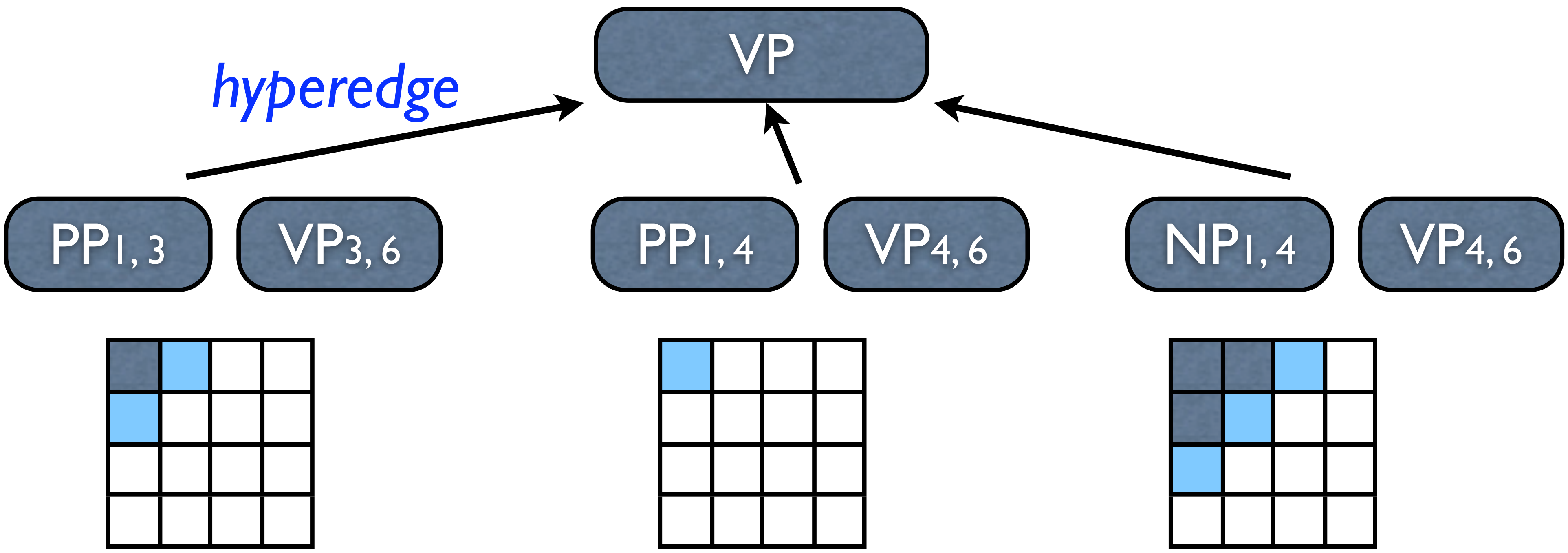


process all hyperedges simultaneously!
significant savings of computation

Across Hyperedges

k-best parsing
(Huang and Chiang, 2005)

on-the-fly rescoring at each node,
instead of only at the root node



process all hyperedges simultaneously!
significant savings of computation

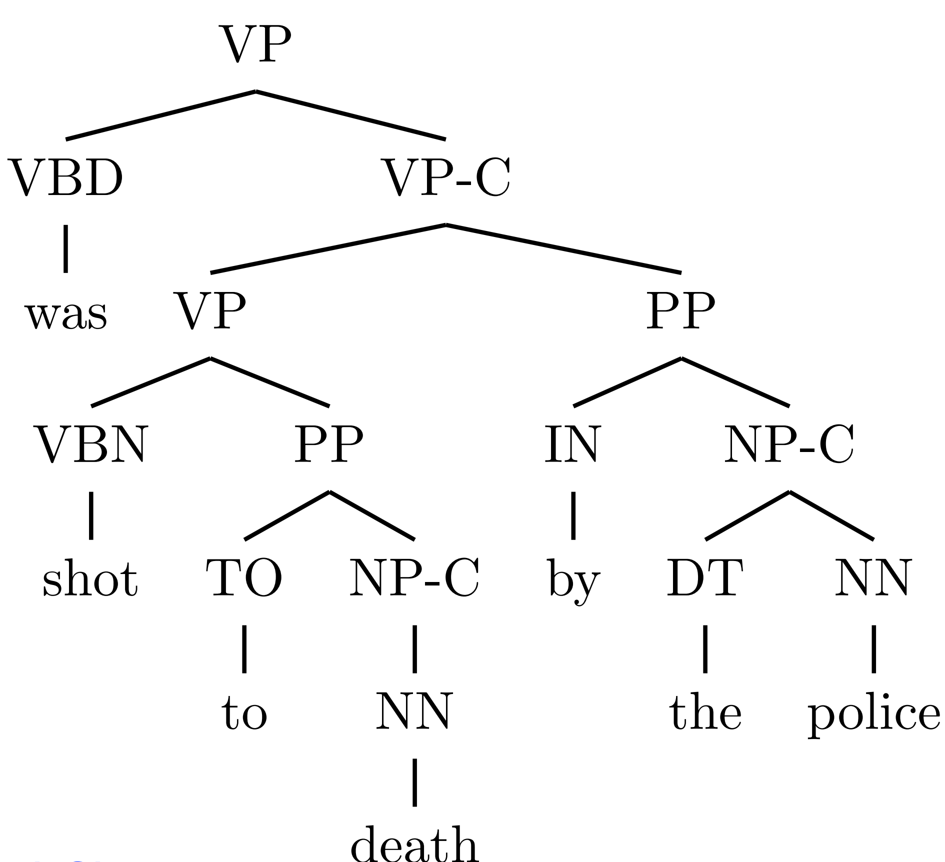
Cube Growing

- an even faster variant of cube pruning
- motivation
  - why do we have a fixed beam of size *k* at each node?
    - why don't we on-the-fly figure out the minimum *k*?
- cube growing uses
  - lazy *k*-best parsing (Huang and Chiang, 2005, Algorithm 3)
  - on-demand computation
- but harder to implement

Syntax-based Experiments

Tree-to-String System

- syntax-directed, English to Chinese (Huang, Knight, Joshi, 2006)
- first parse input, and then recursively transfer



synchronous tree-substitution grammars (STSG)

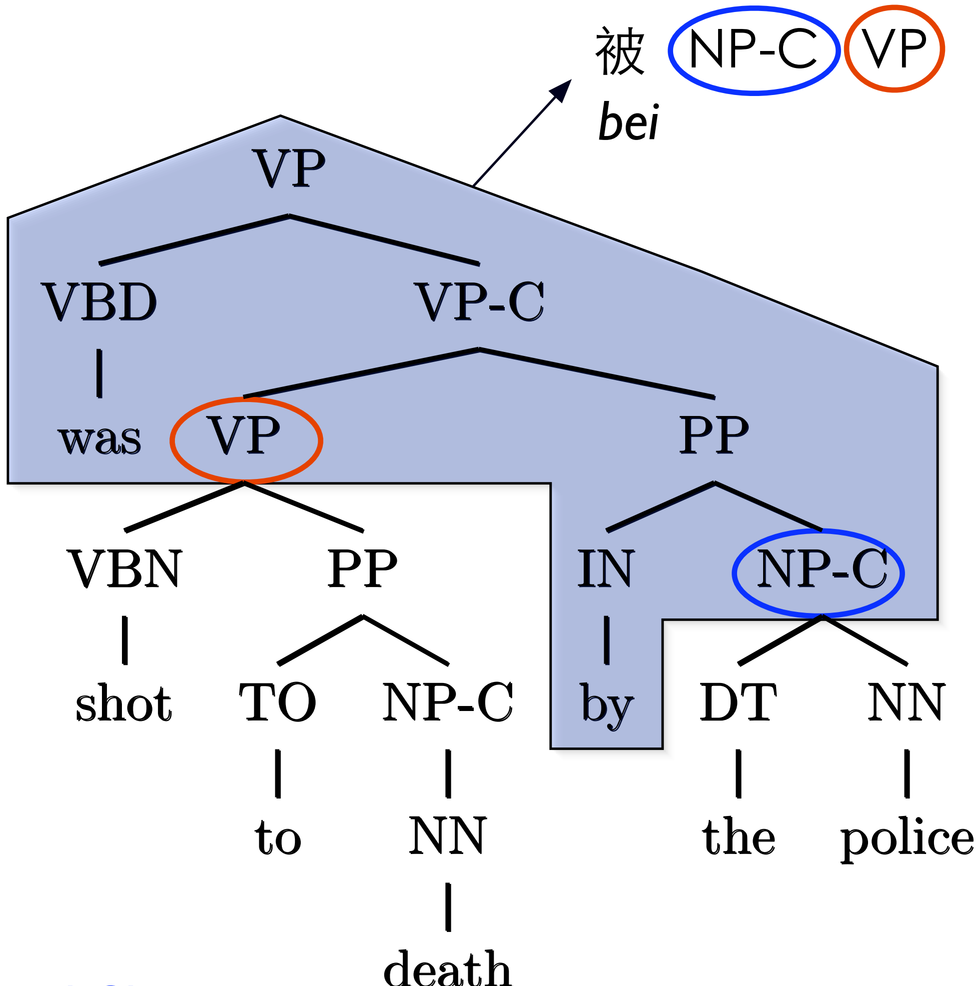
(Galley et al., 2004; Eisner, 2003)

search space
still a hypergraph

tested on 140 sentences
slightly better BLEU scores
than Pharaoh

Tree-to-String System

- syntax-directed, English to Chinese (Huang, Knight, Joshi, 2006)
- first parse input, and then recursively transfer



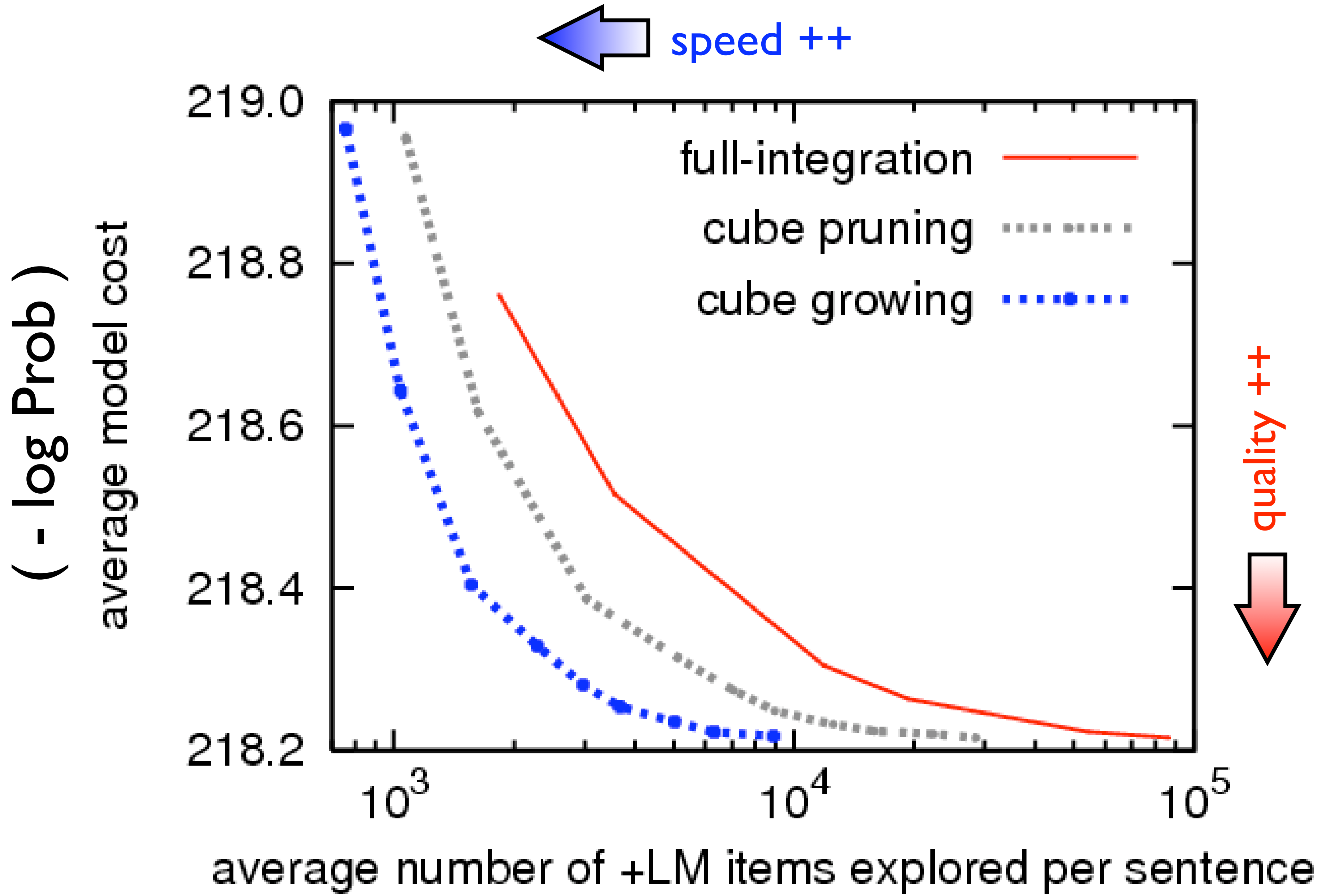
synchronous tree-substitution grammars (STSG)

(Galley et al., 2004; Eisner, 2003)

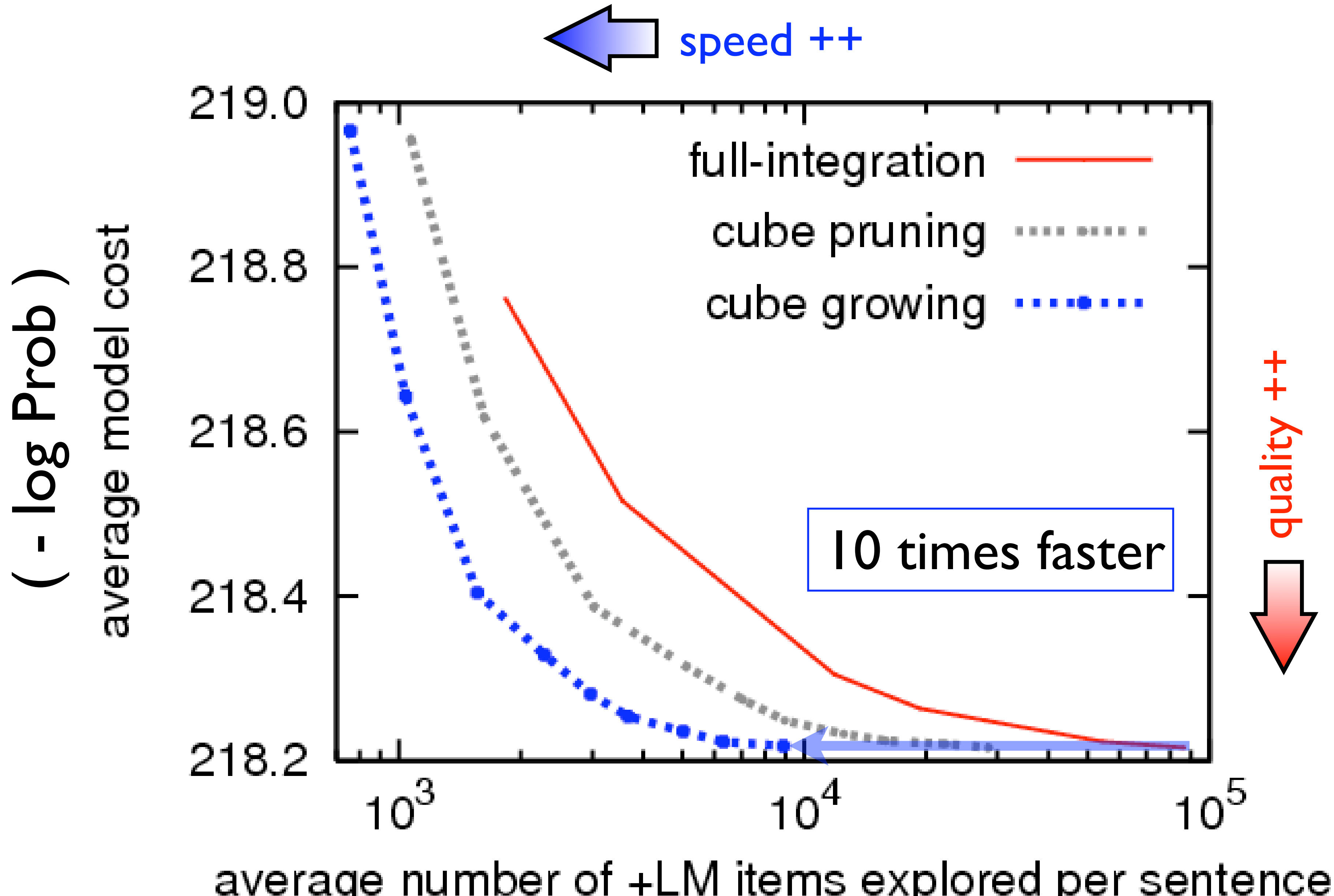
search space
still a hypergraph

tested on 140 sentences
slightly better BLEU scores than Pharaoh

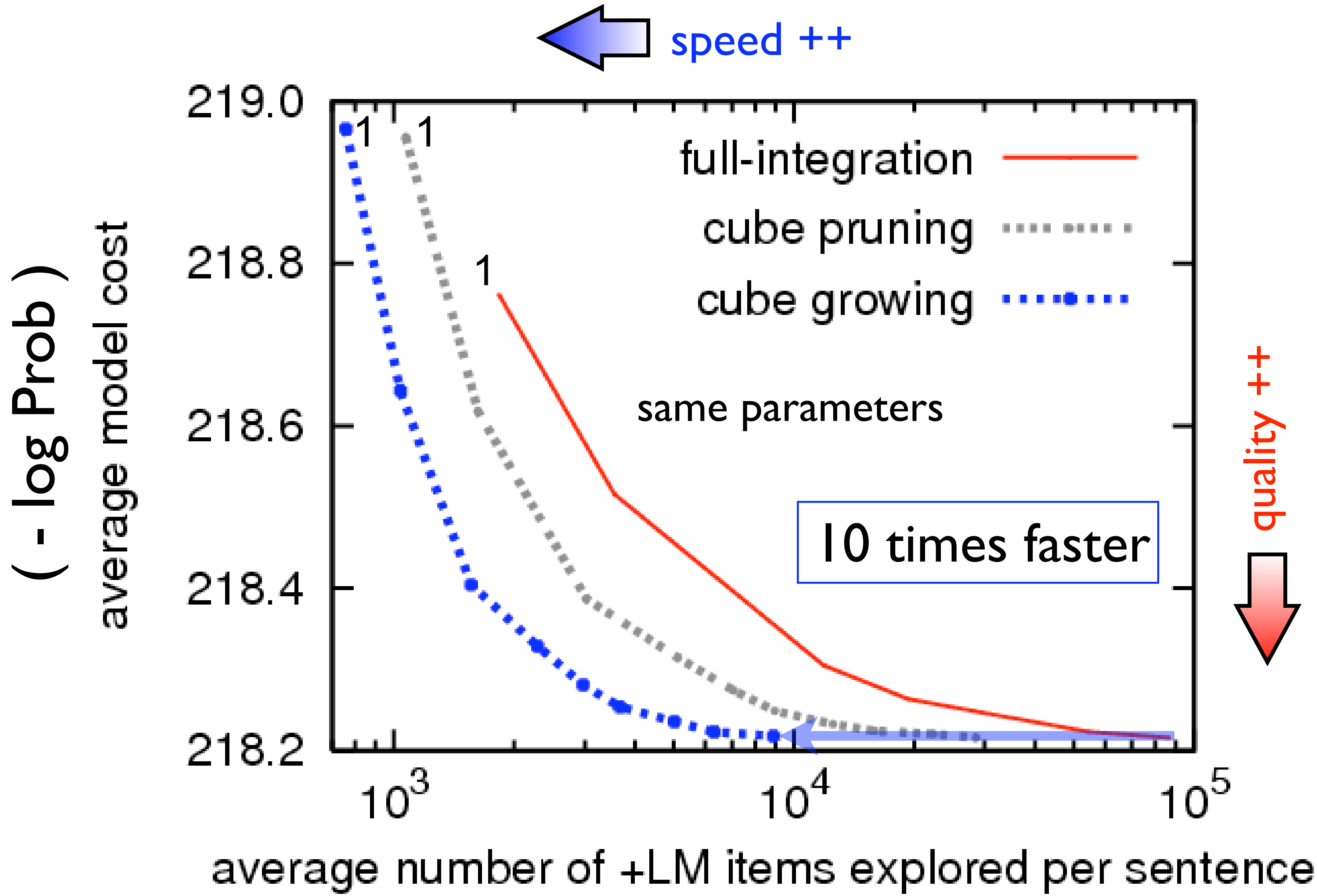
Speed vs. Search Quality



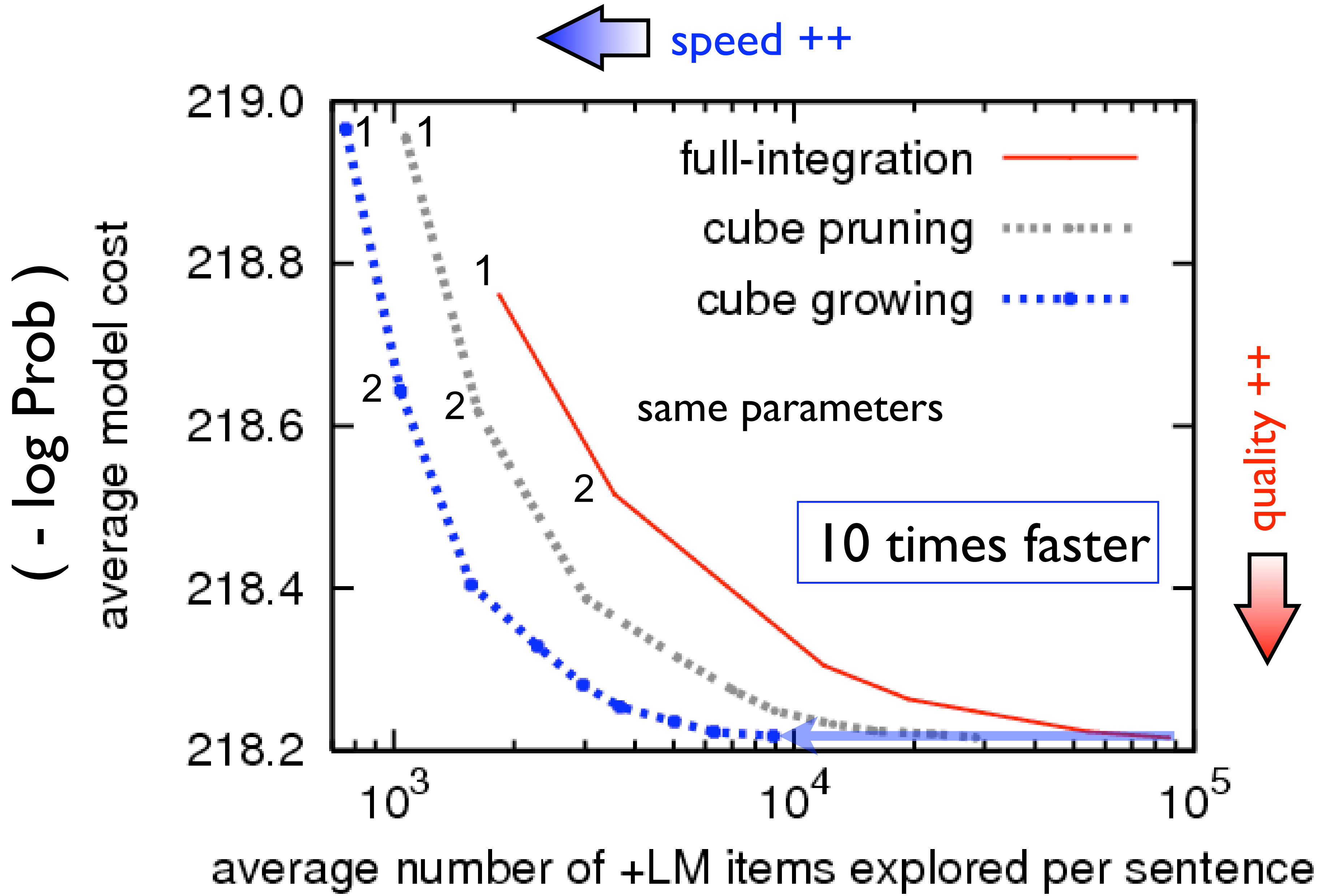
Speed vs. Search Quality



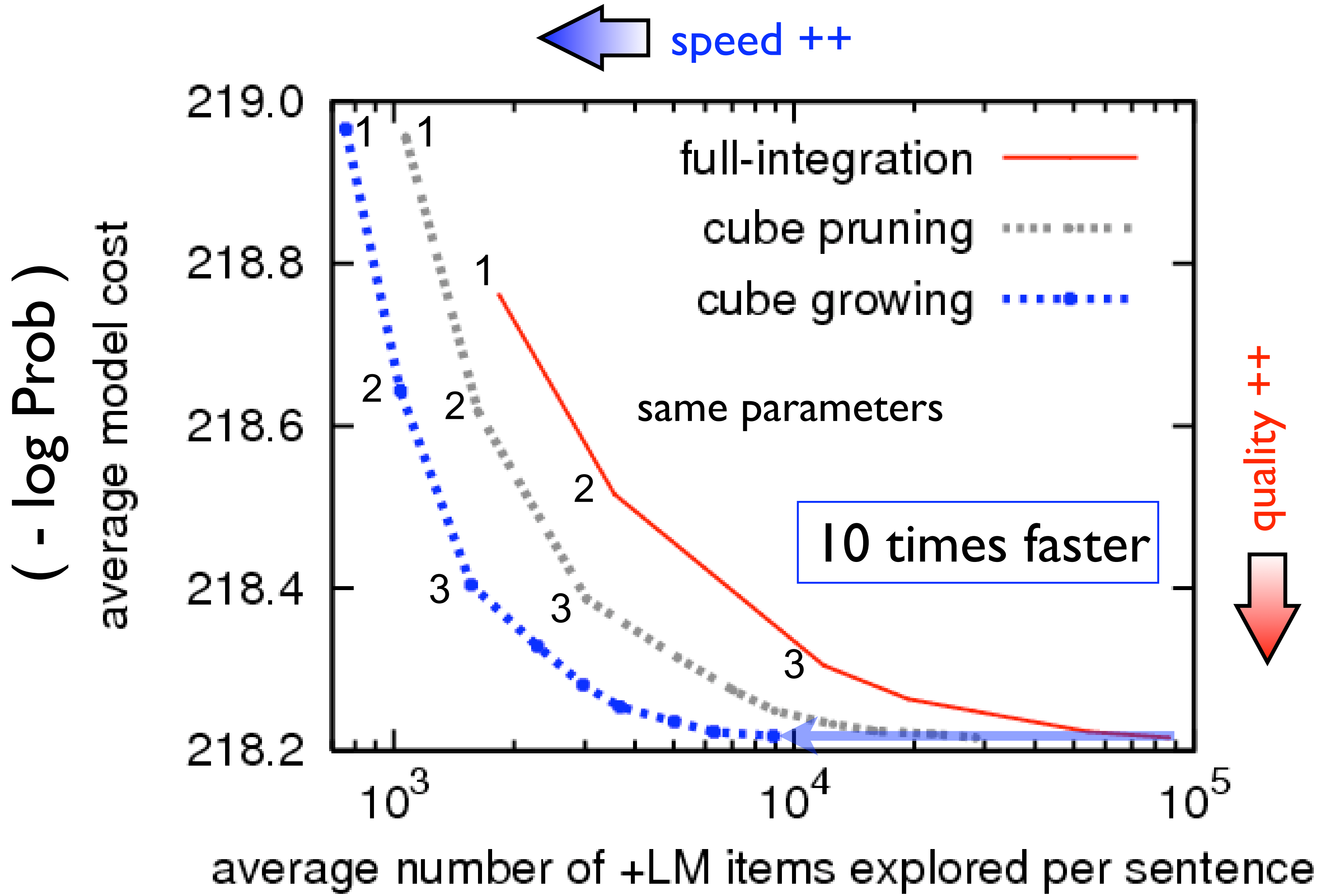
Speed vs. Search Quality



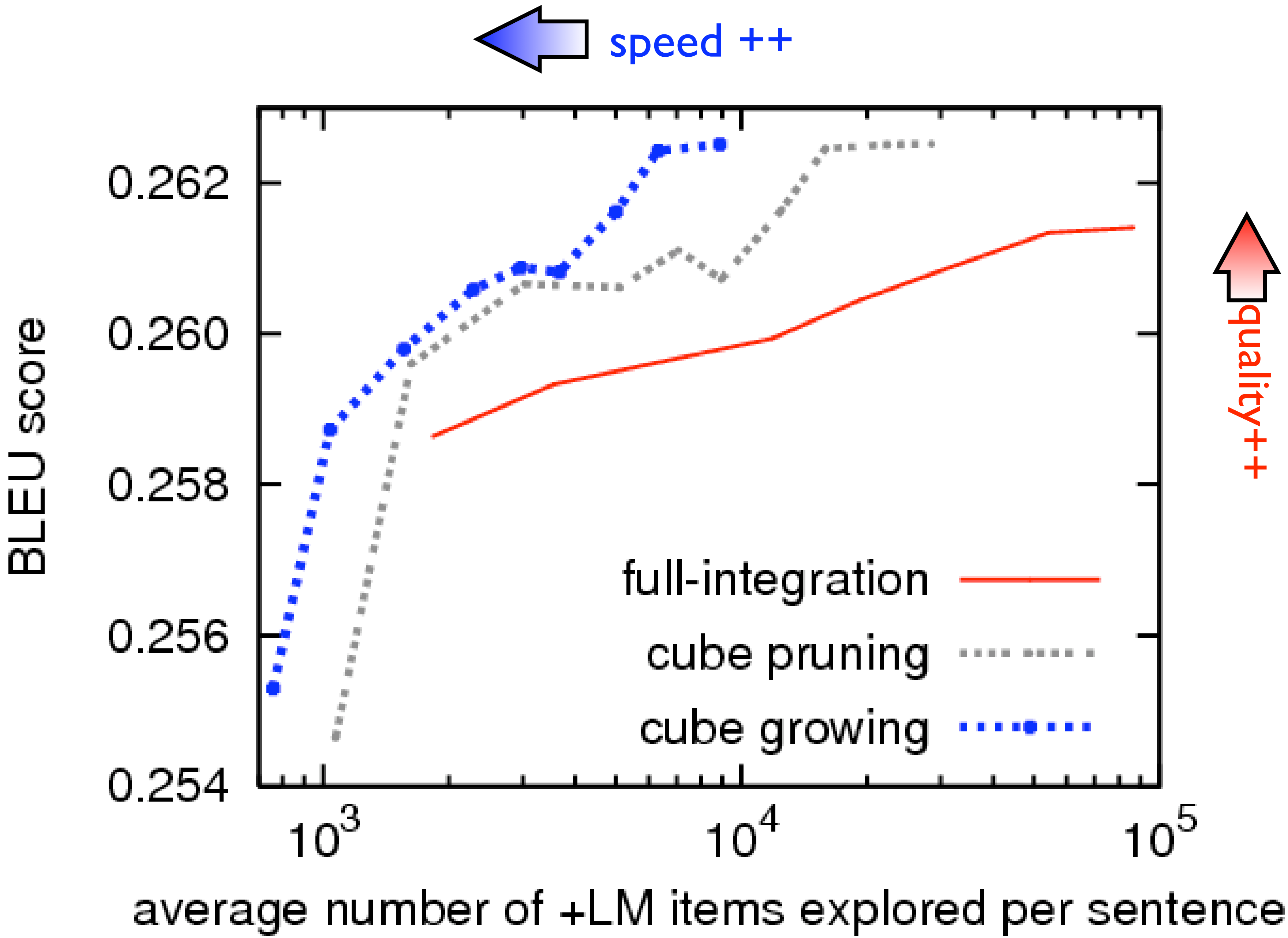
Speed vs. Search Quality



Speed vs. Search Quality



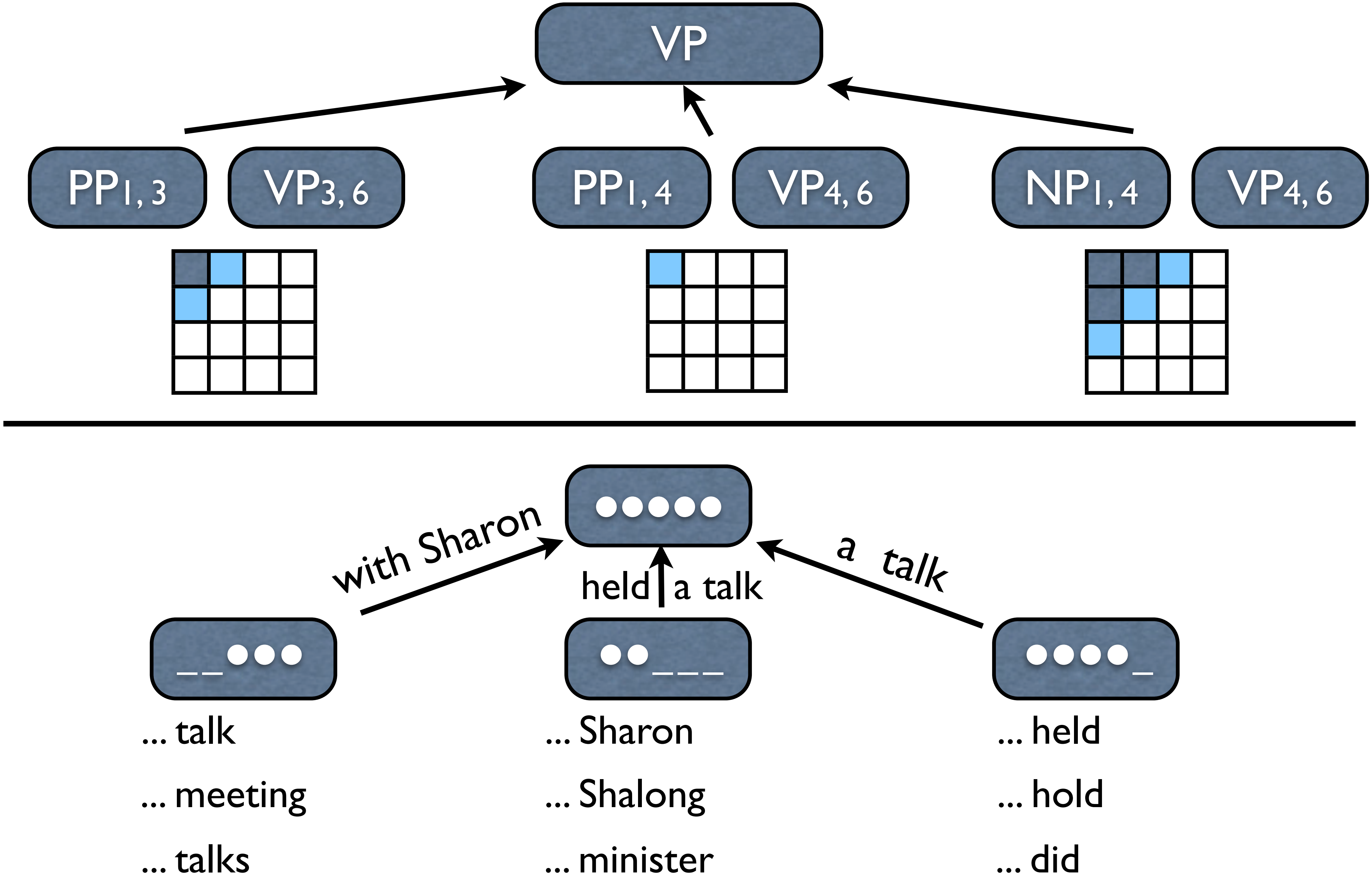
Speed vs. Translation Accuracy



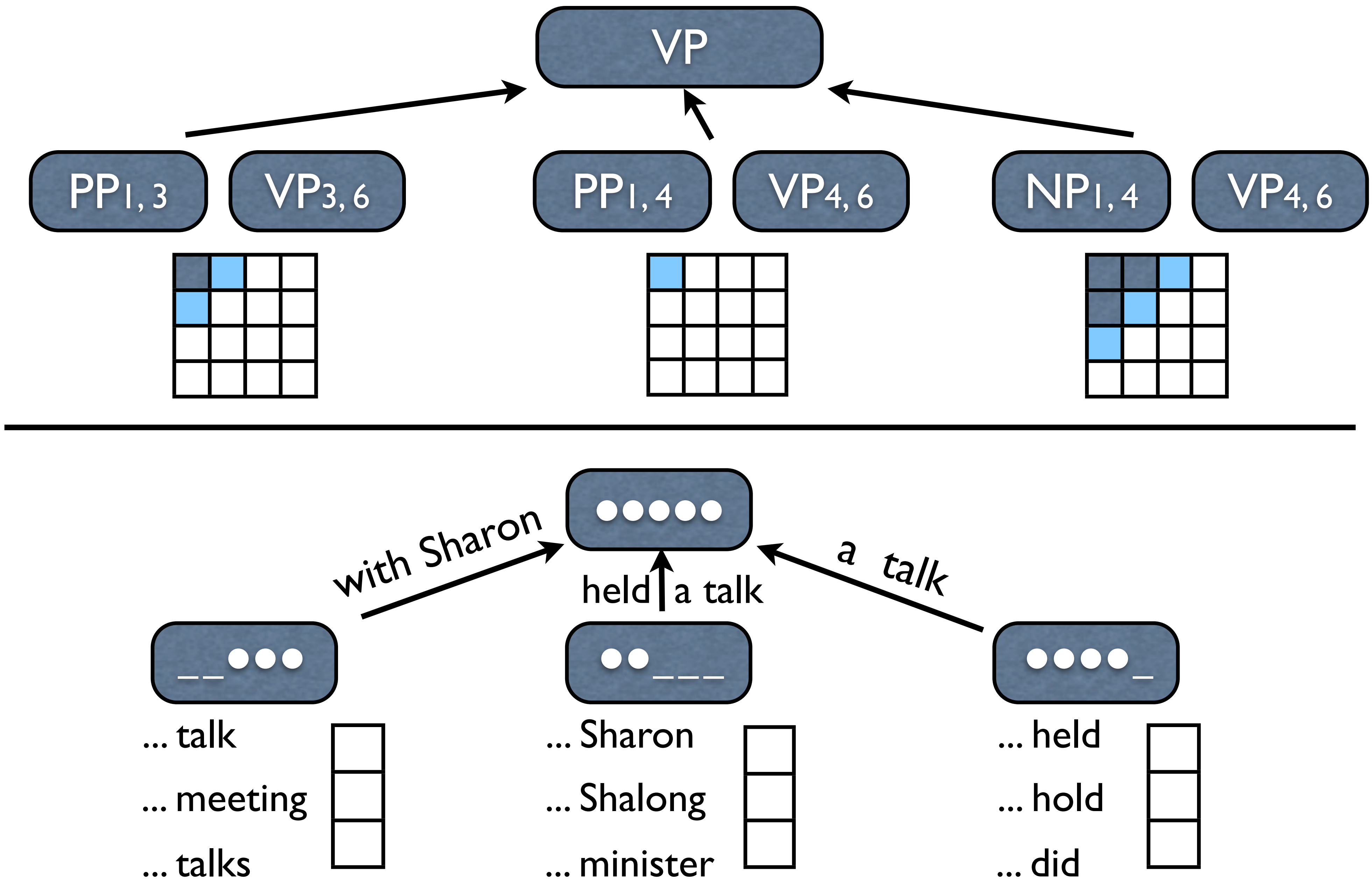
Cube-Pruning for

Phrase-based Decoding

Syntax vs. Phrase-based

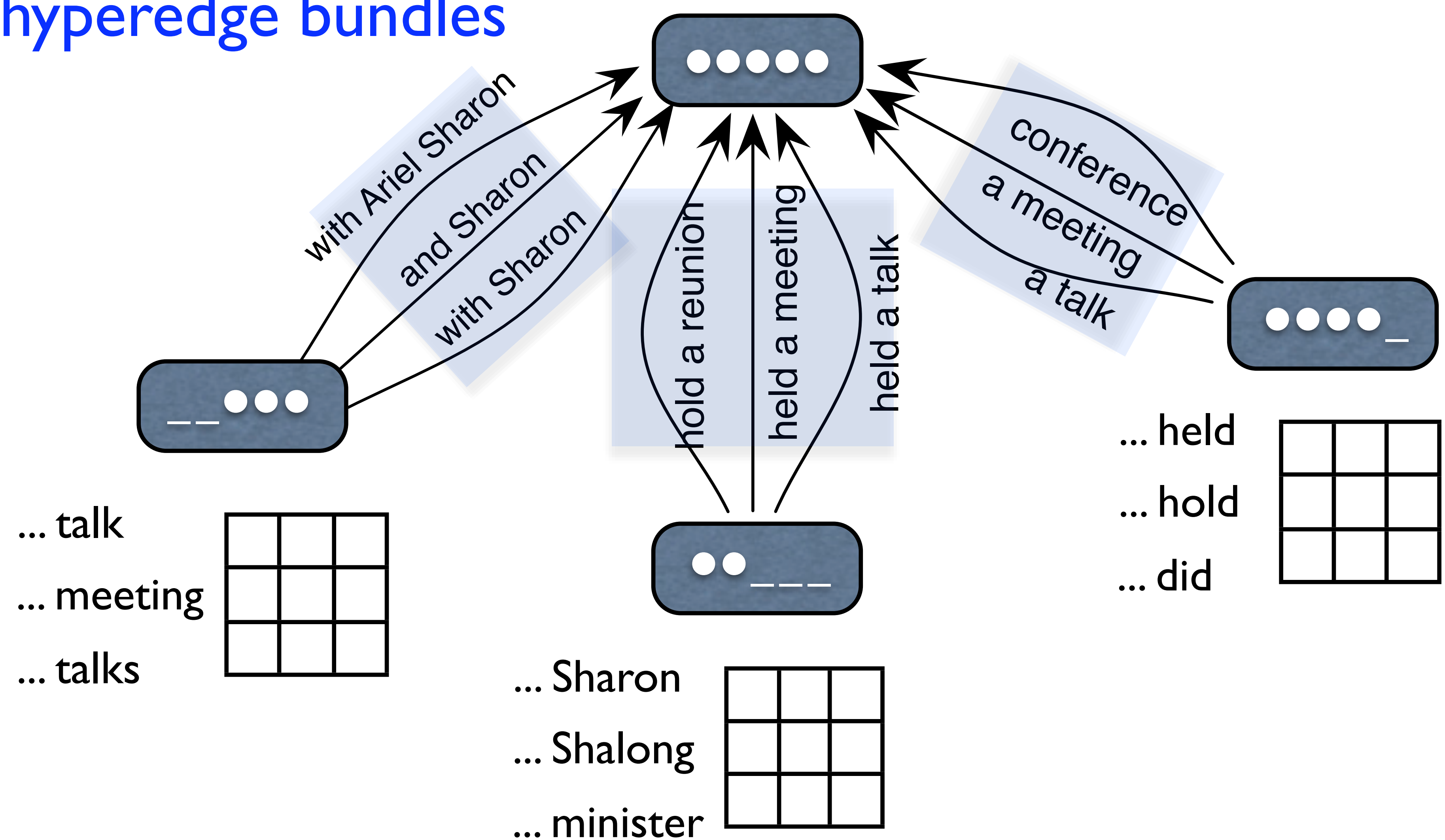


Syntax vs. Phrase-based



Alternative Phrase-Pairs

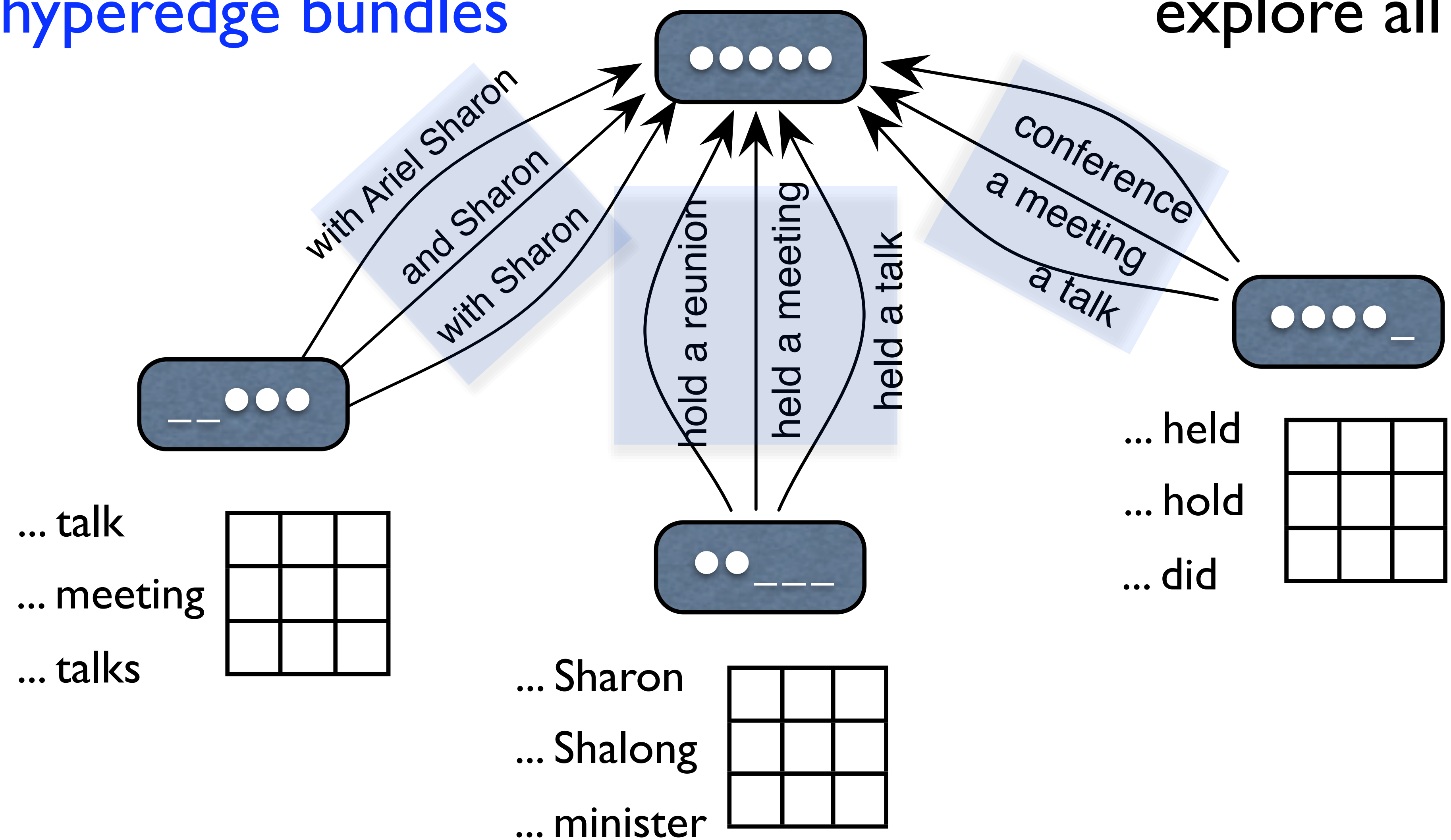
grouping into
hyperedge bundles



Alternative Phrase-Pairs

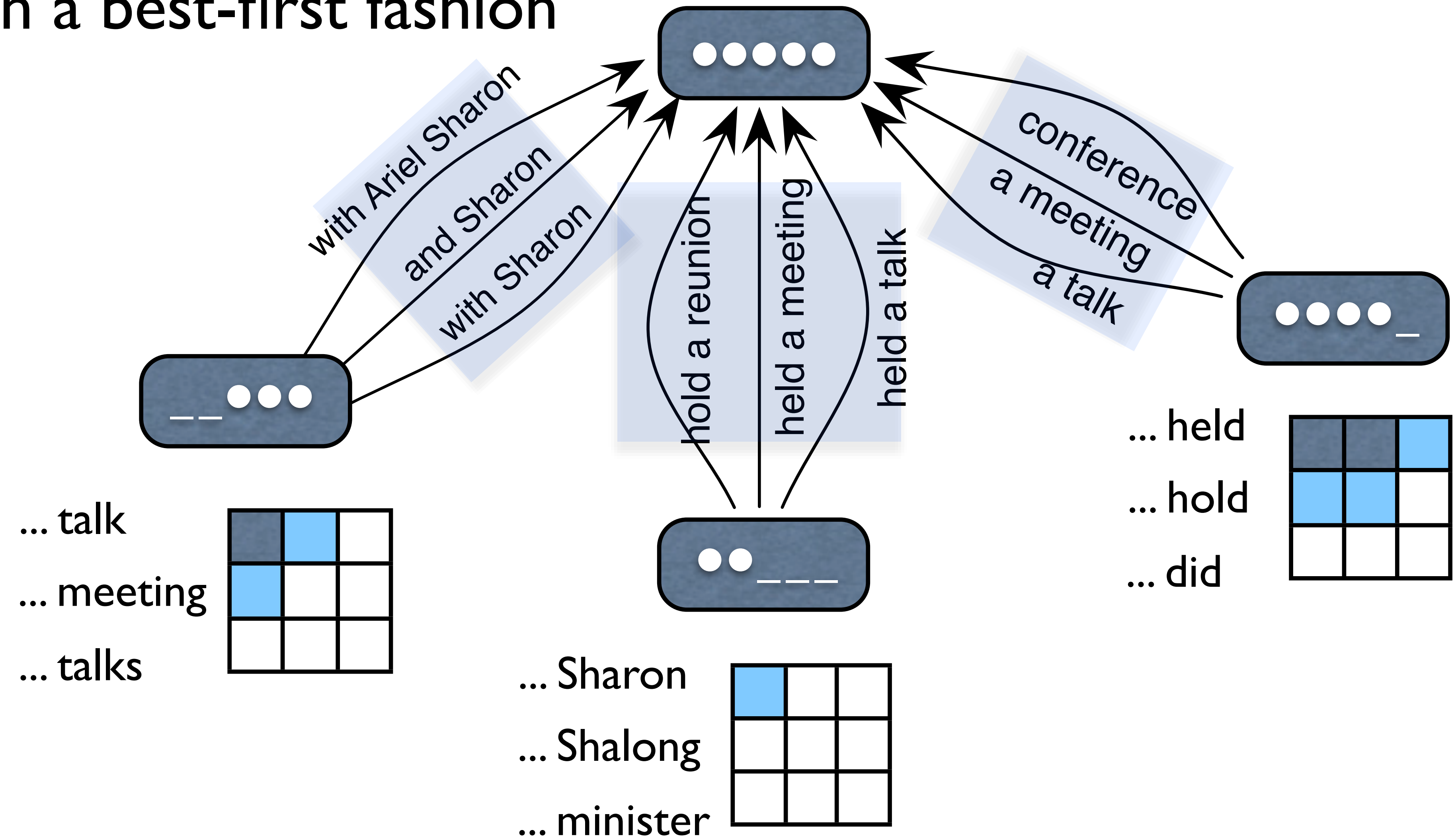
grouping into
hyperedge bundles

Pharaoh would
explore all cells



Cube Pruning

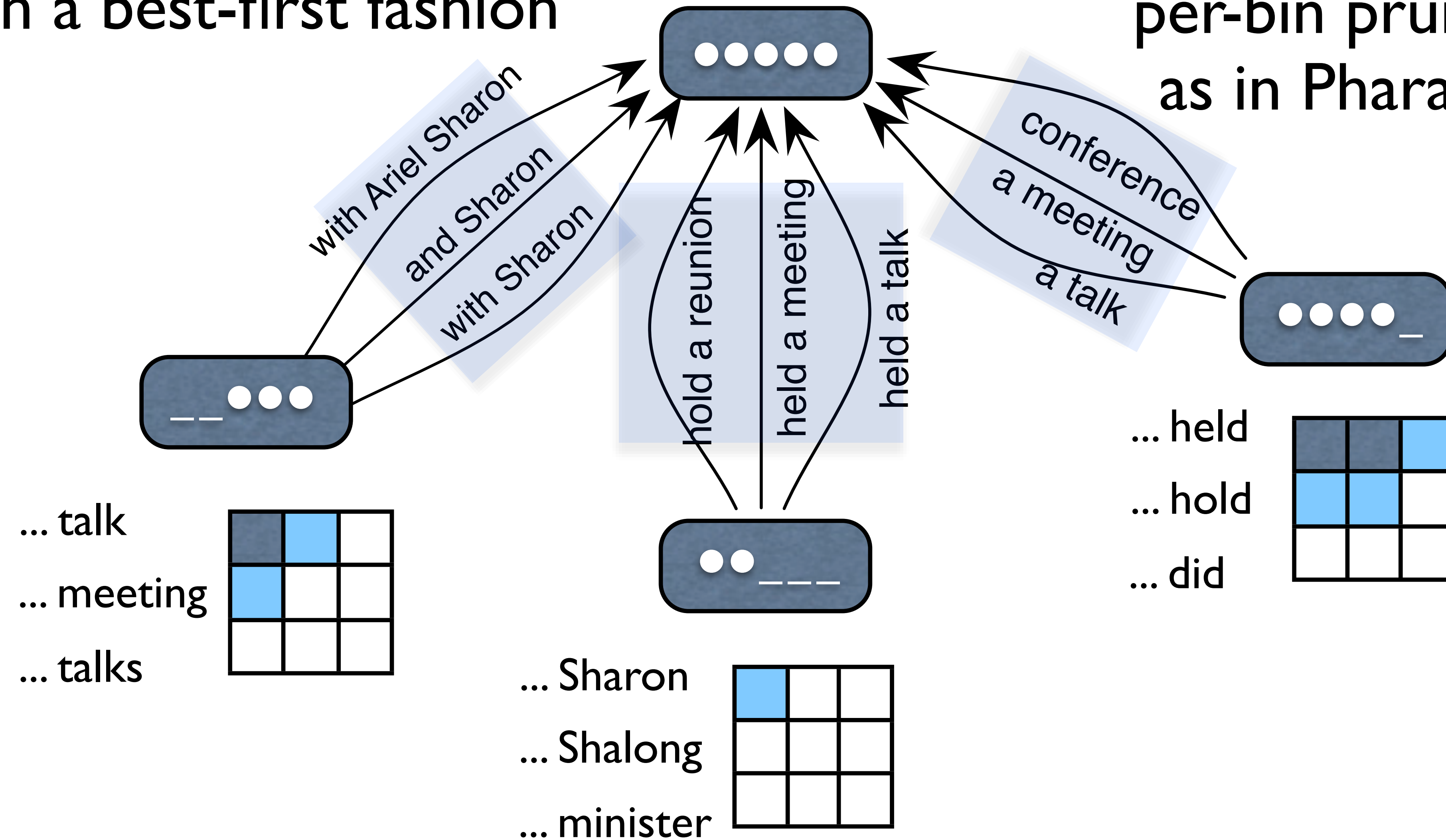
but we explore the grids
in a best-first fashion



Cube Pruning

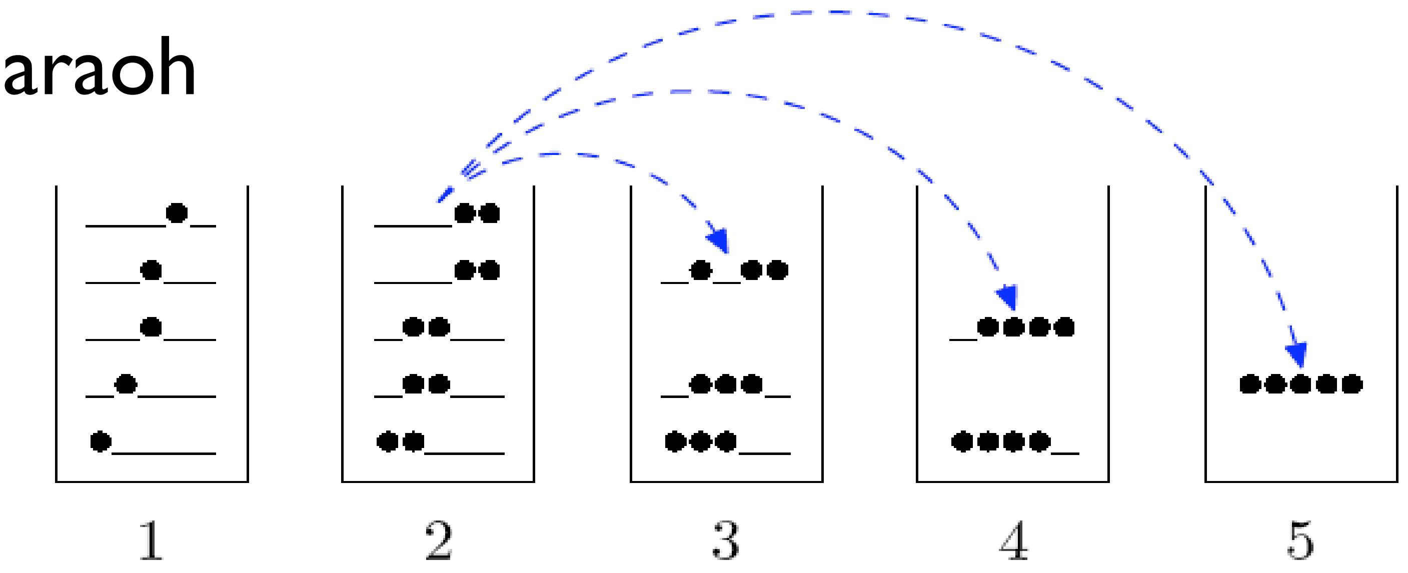
but we explore the grids in a best-first fashion

in practice we use per-bin pruning as in Pharaoh



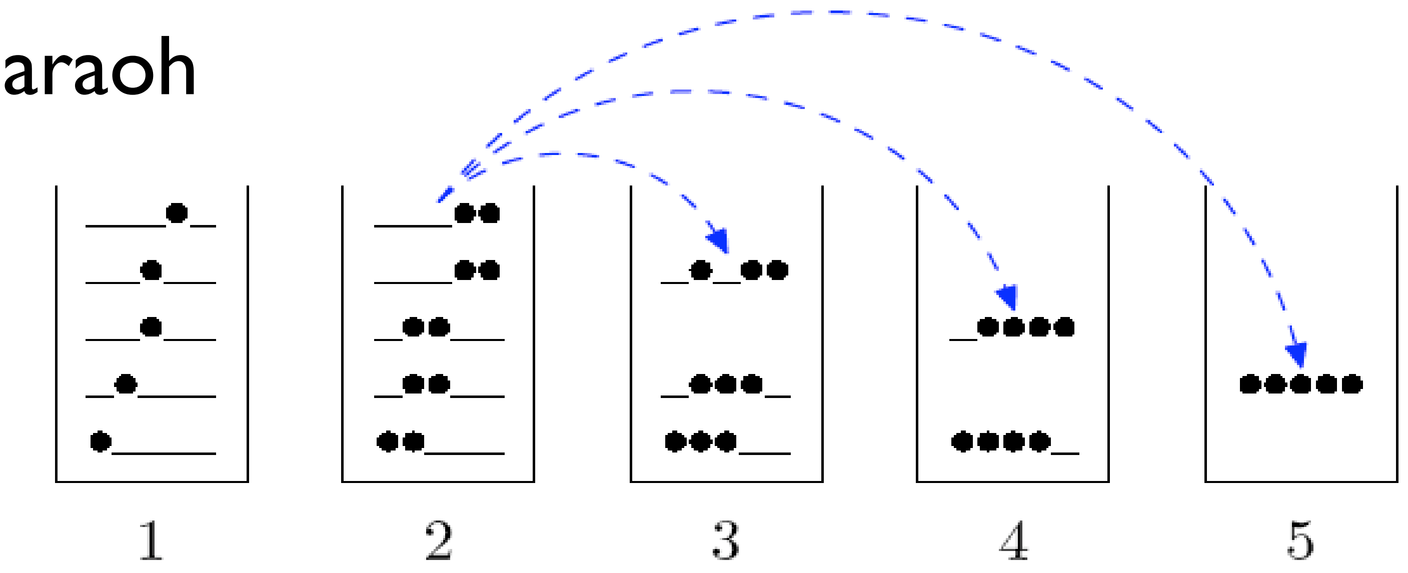
In Practice: per-bin Pruning

Pharaoh

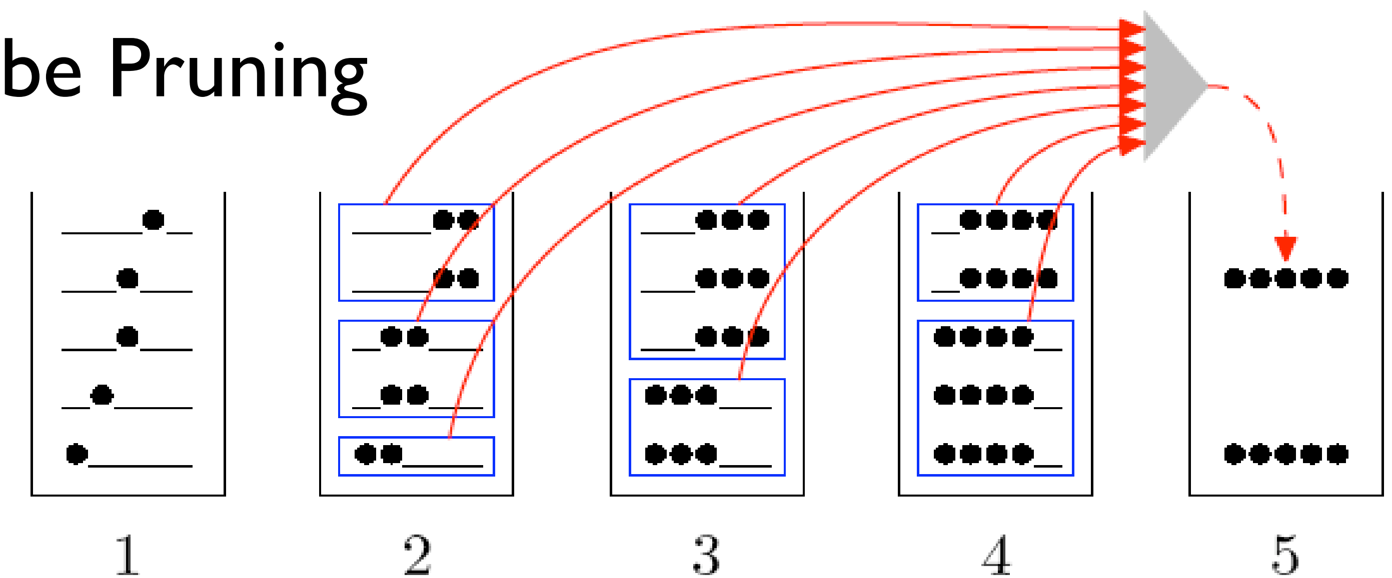


In Practice: per-bin Pruning

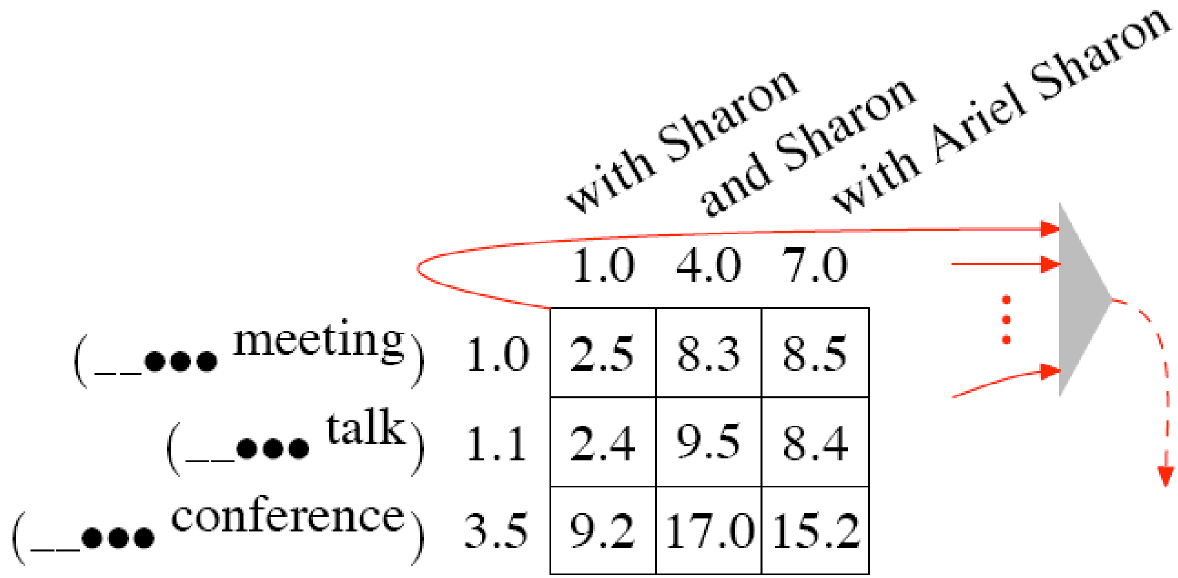
Pharaoh



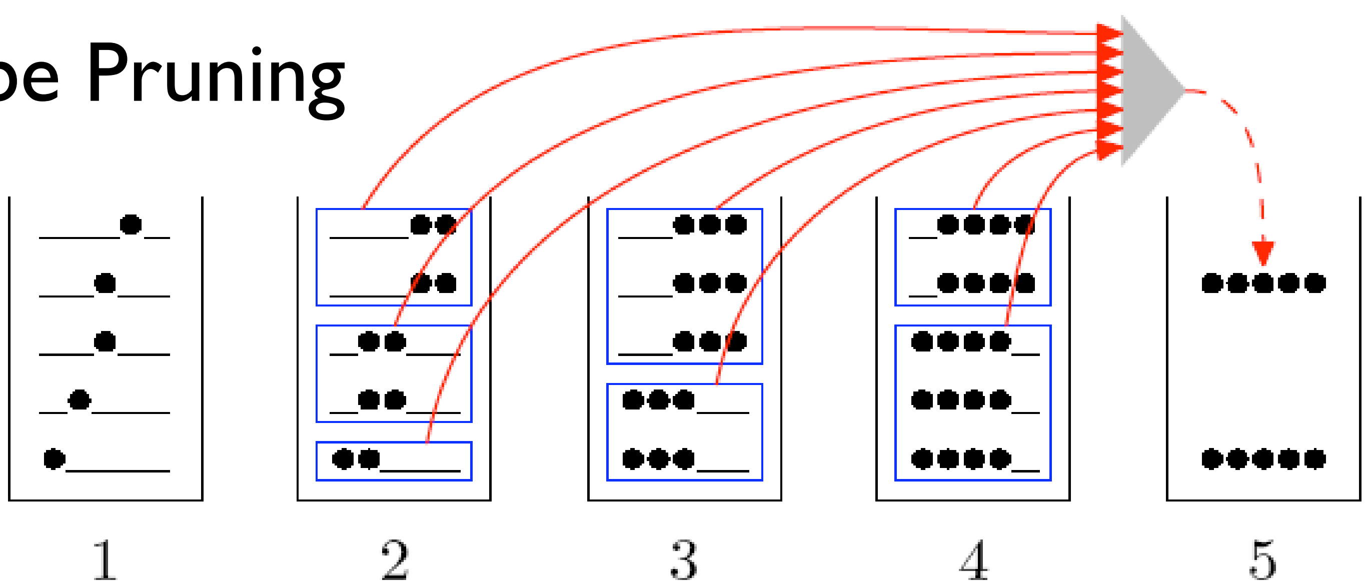
Cube Pruning



In Practice: per-bin Pruning

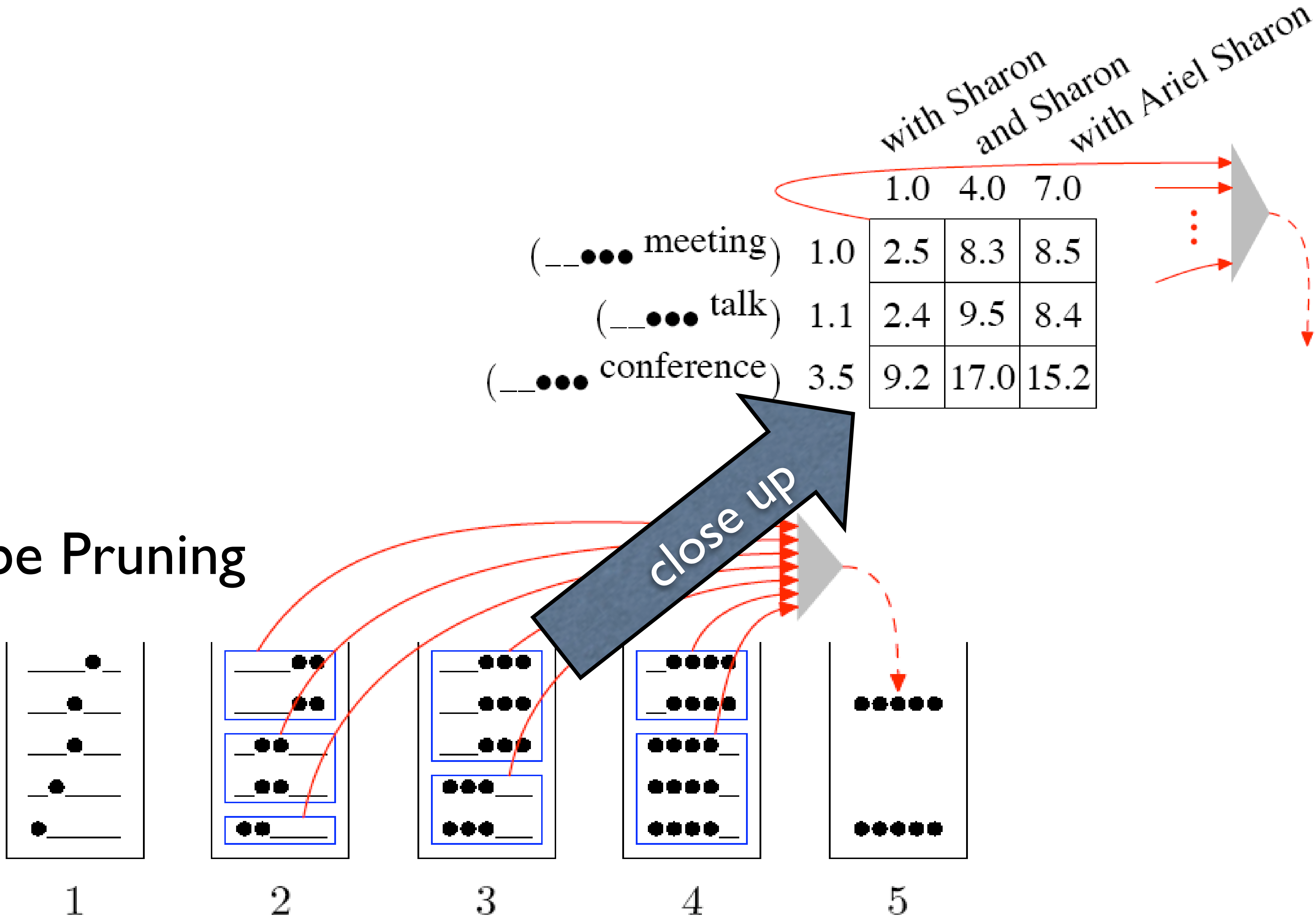


Cube Pruning

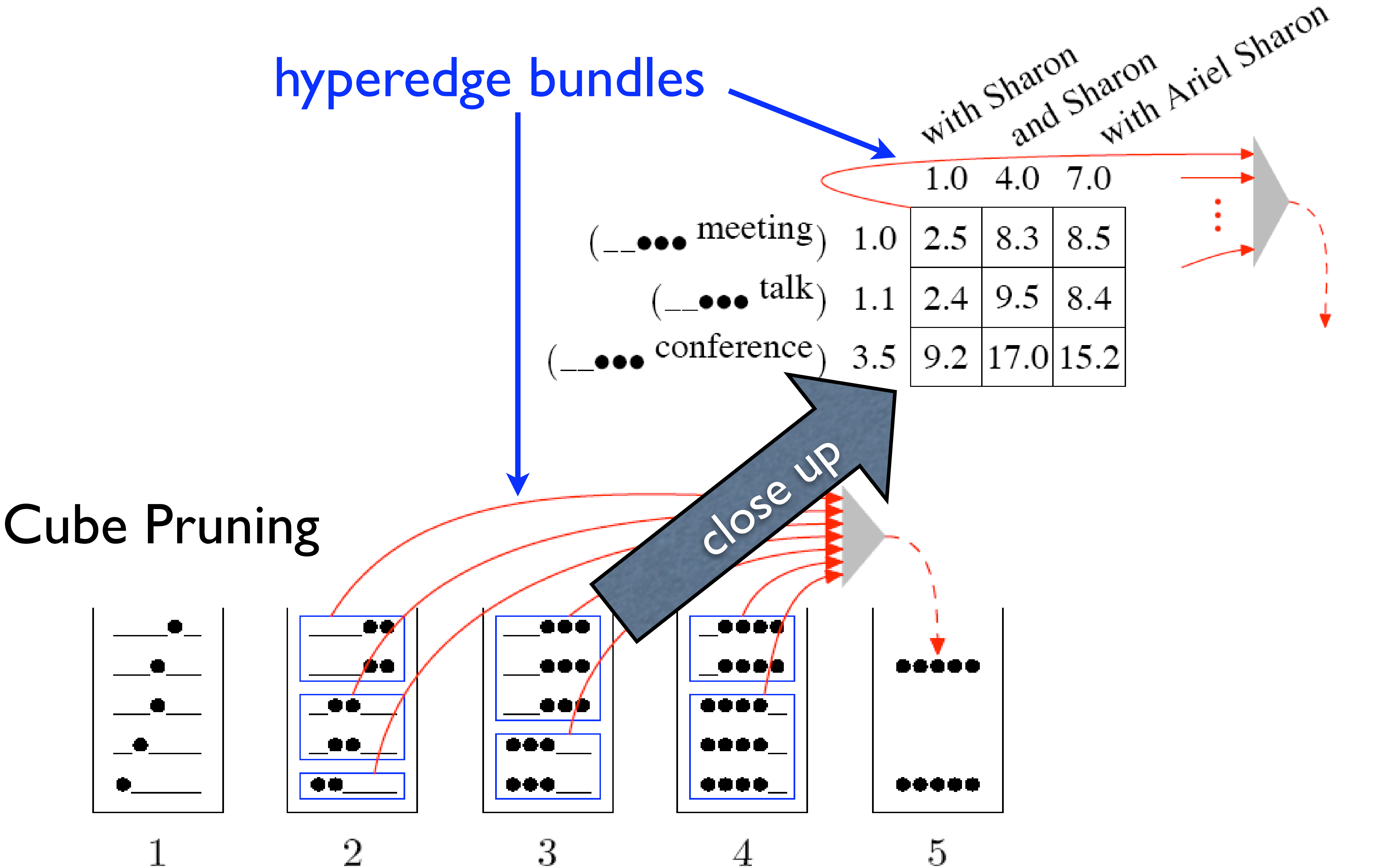


In Practice: per-bin Pruning

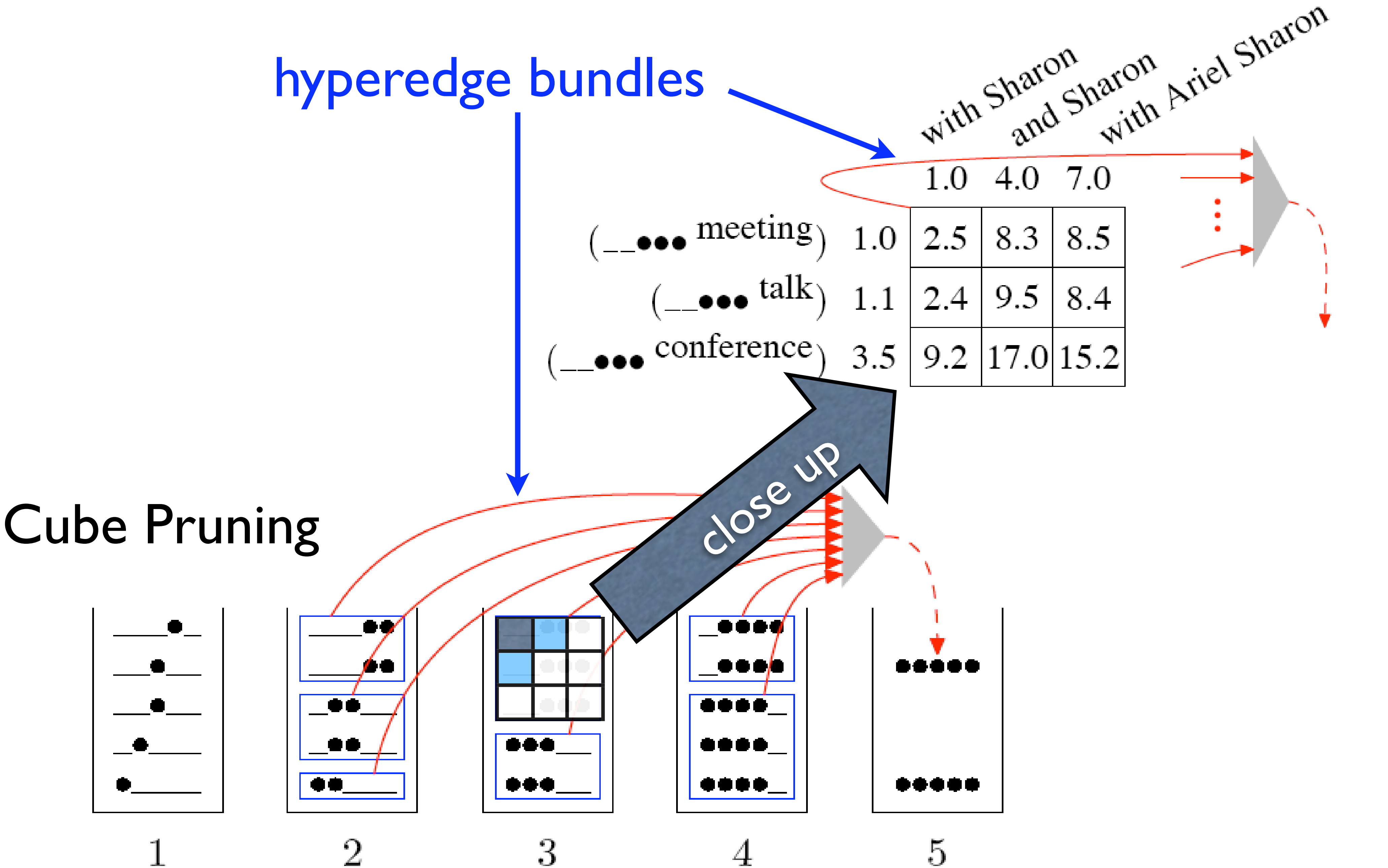
Cube Pruning



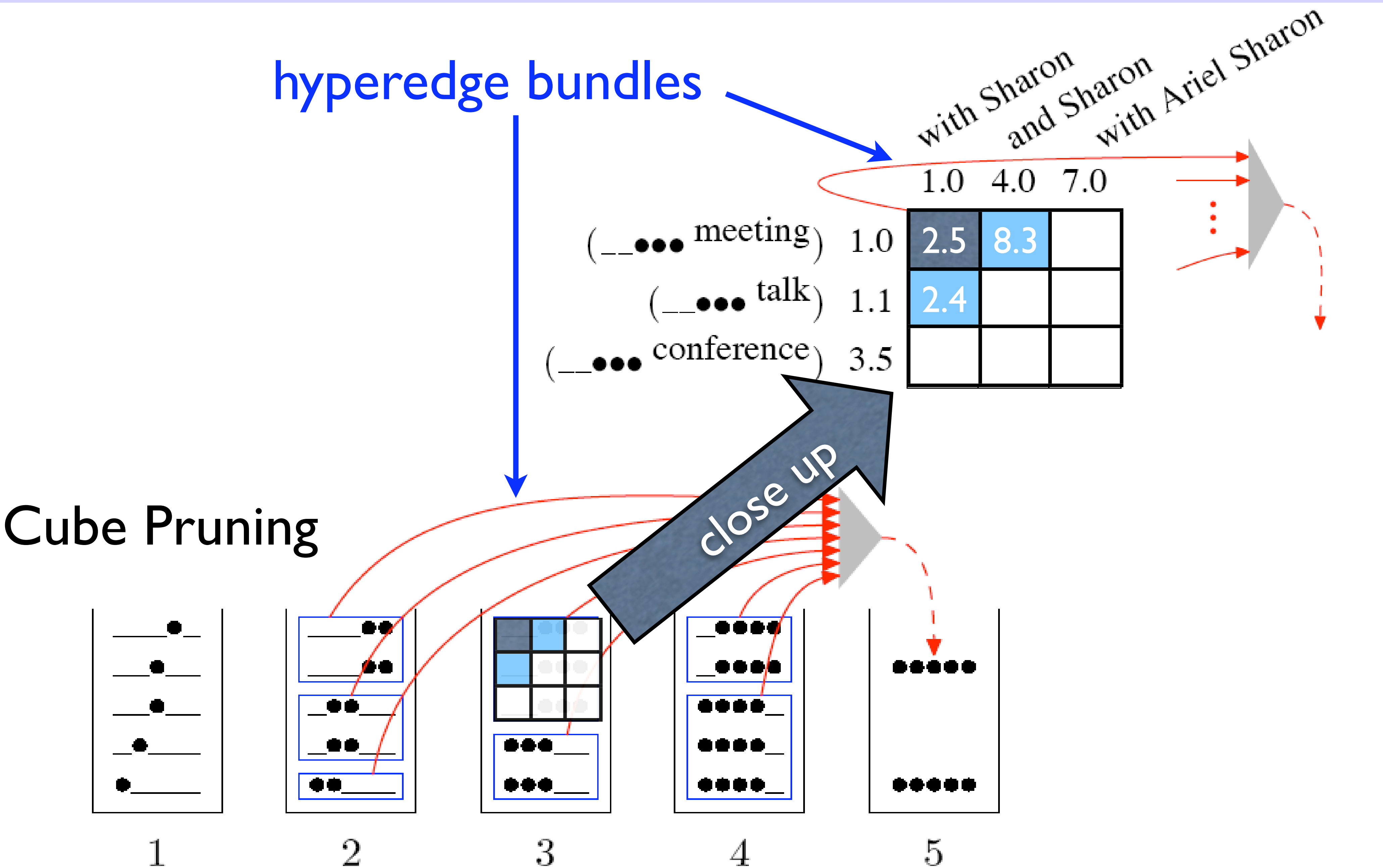
In Practice: per-bin Pruning



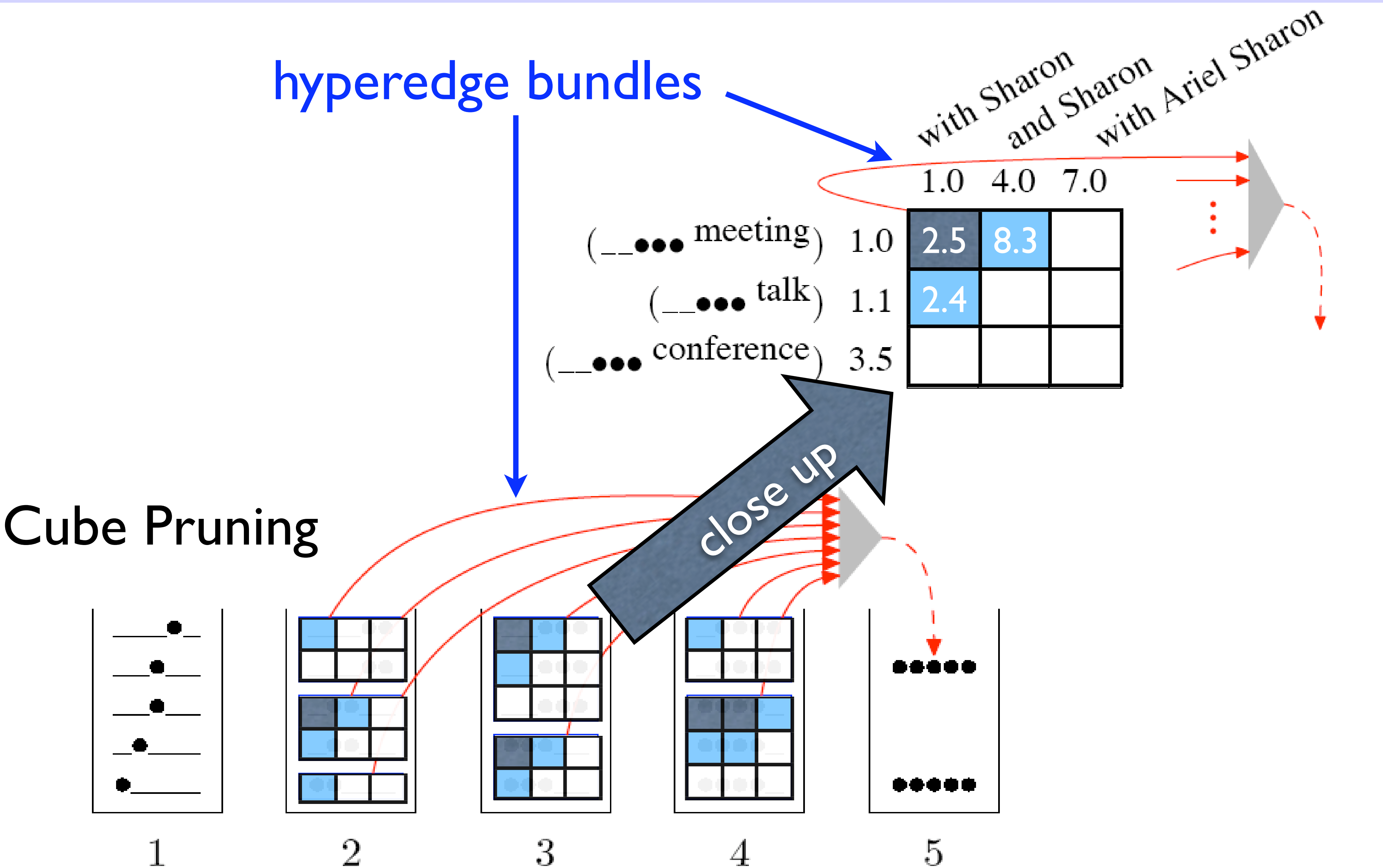
In Practice: per-bin Pruning



In Practice: per-bin Pruning

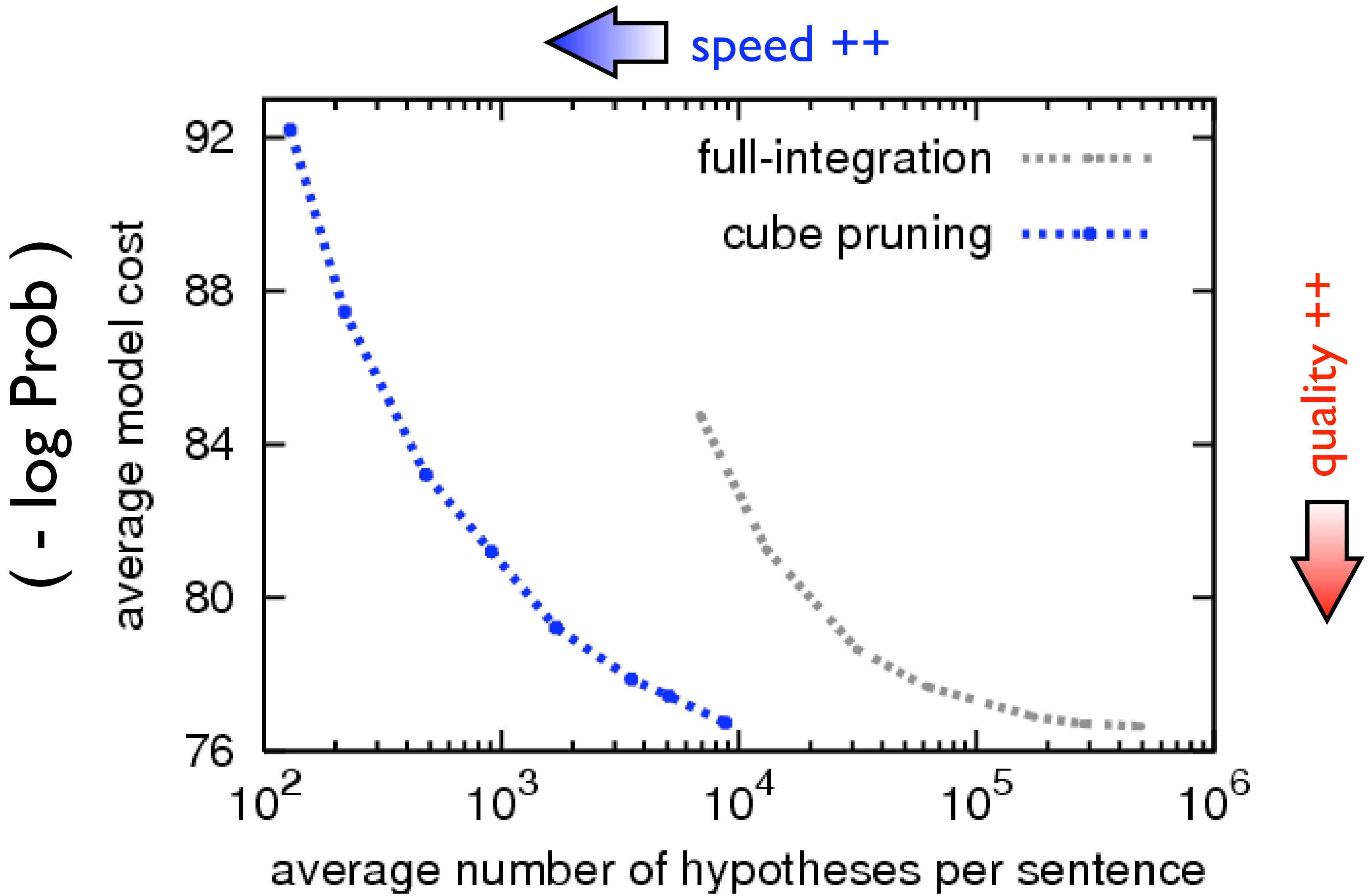


In Practice: per-bin Pruning



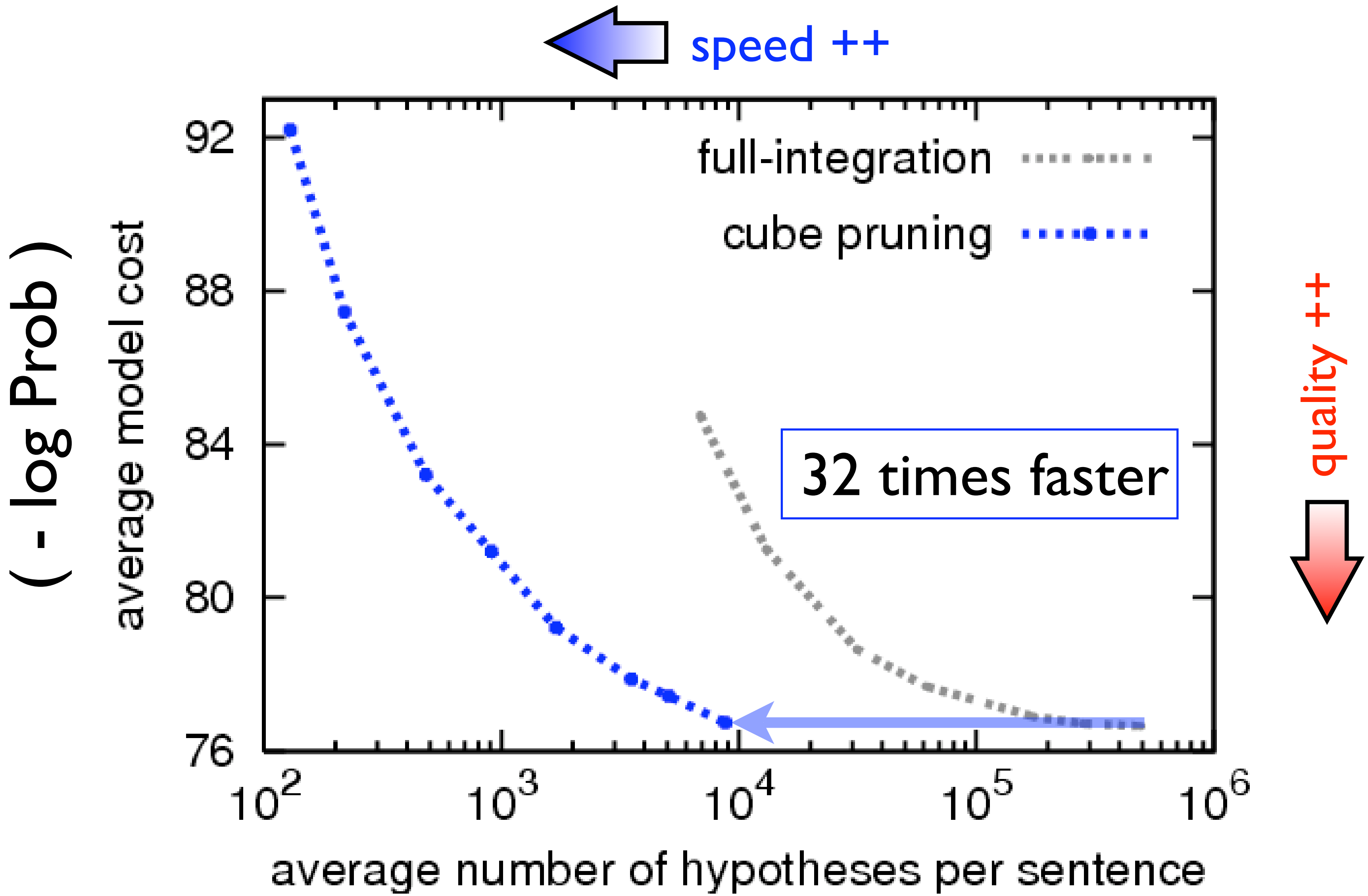
Speed vs. Search Quality

tested on our faithful clone of Pharaoh



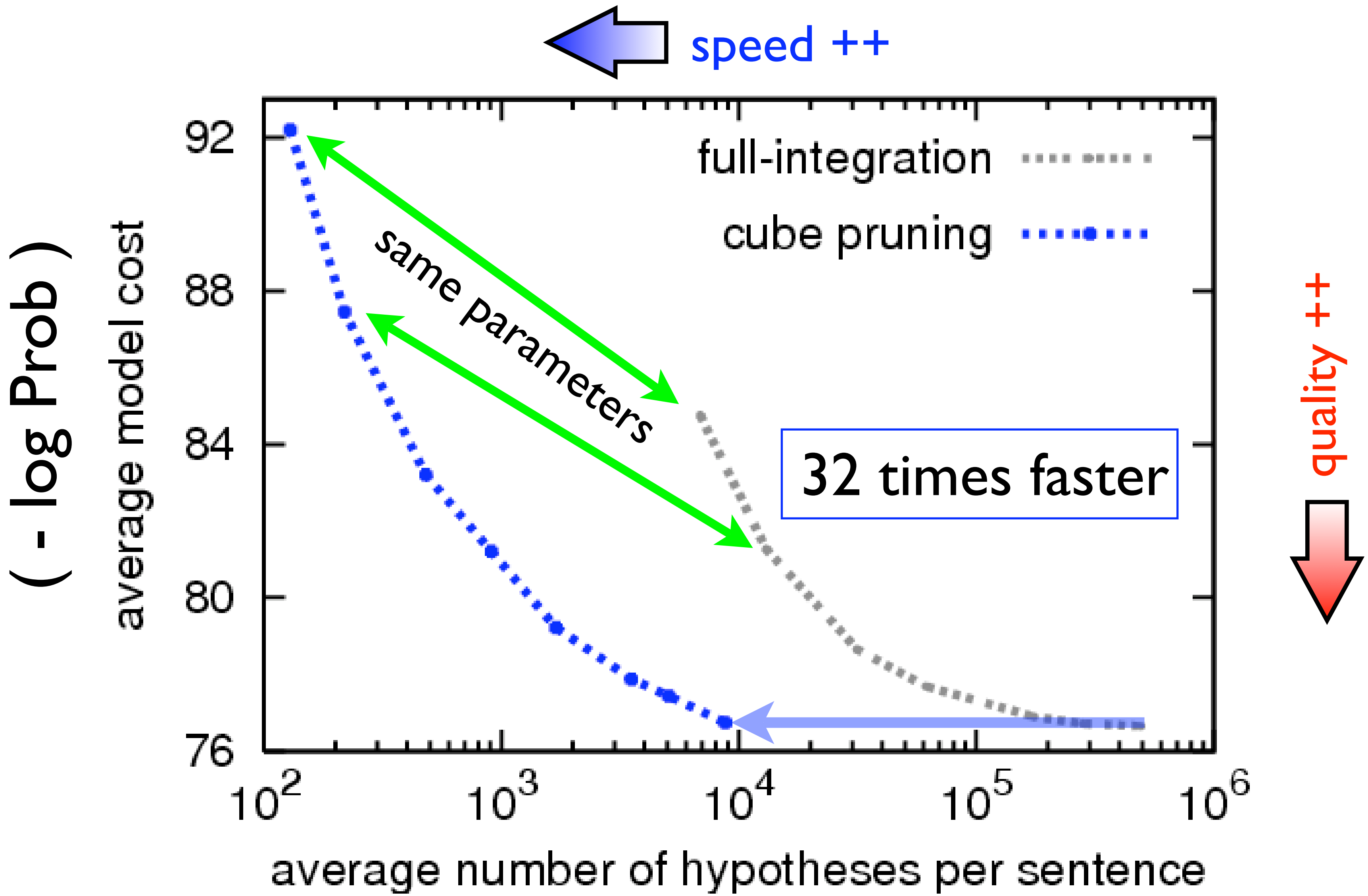
Speed vs. Search Quality

tested on our faithful clone of Pharaoh

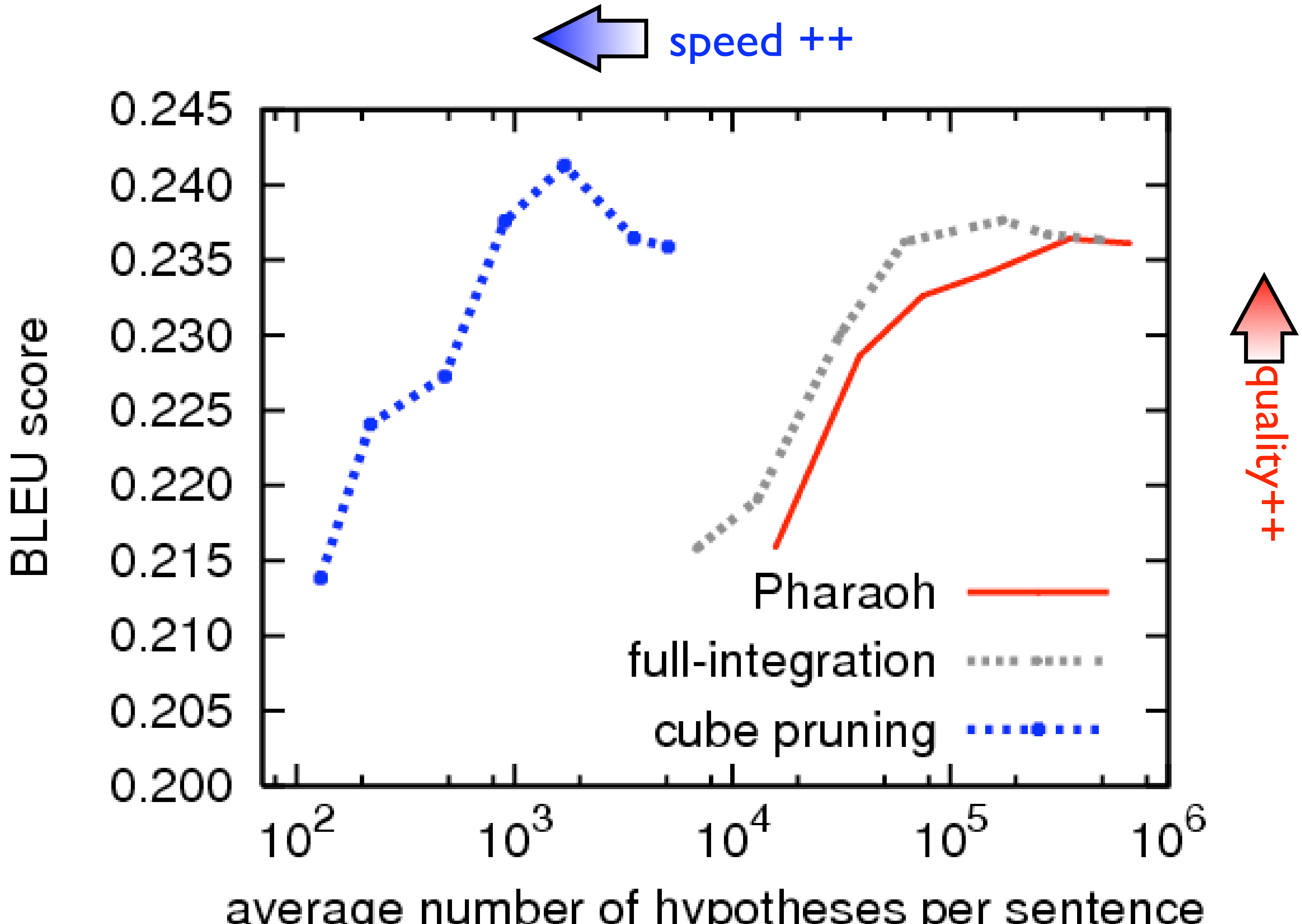


Speed vs. Search Quality

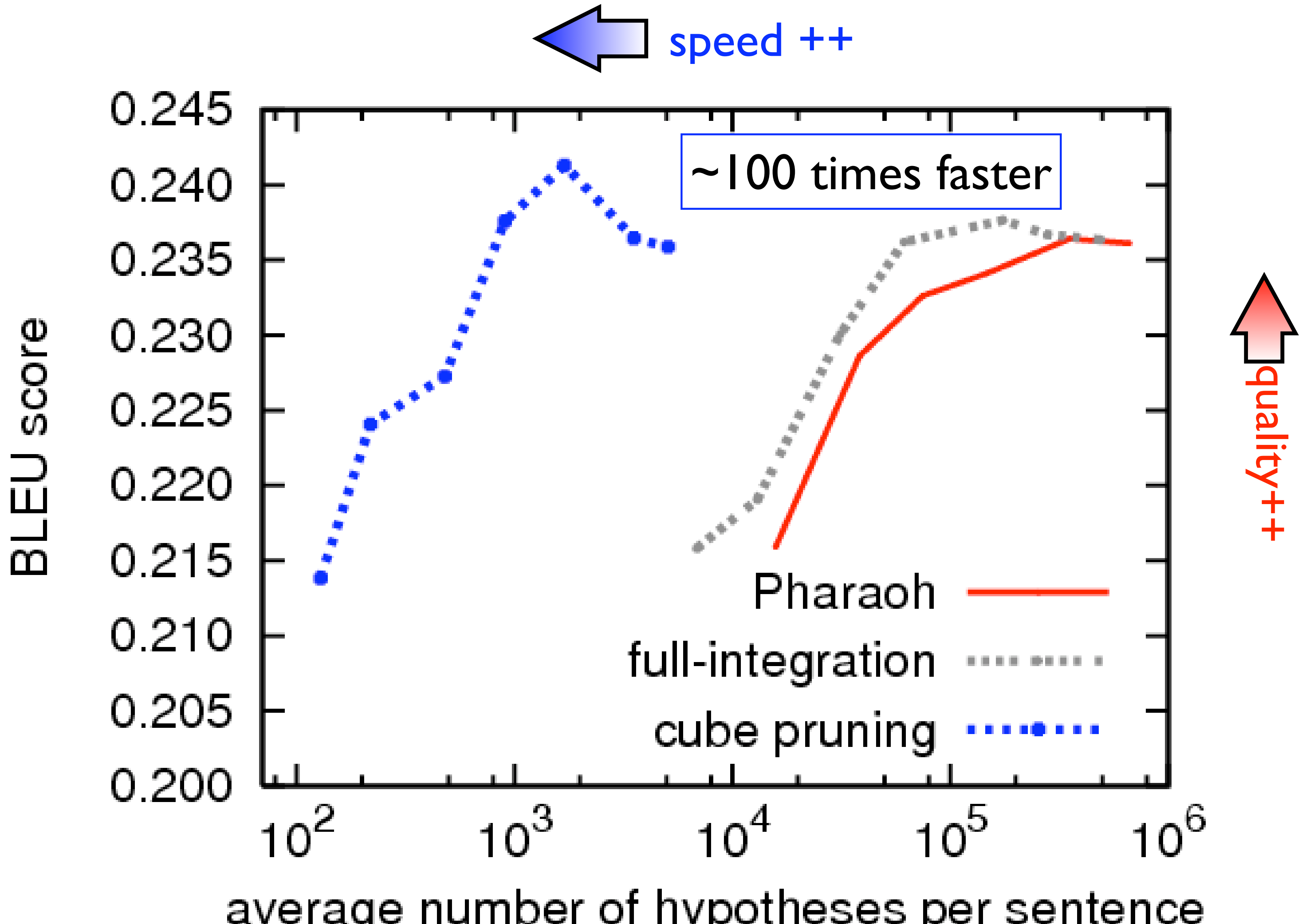
tested on our faithful clone of Pharaoh



Speed vs. Translation Accuracy



Speed vs. Translation Accuracy



Conclusions

- forest-rescoring: cube pruning and cube growing
  - on-the-fly rescoring using *k*-best parsing
  - applicable to both phrase- and syntax-based systems
  - significant speed-up against conventional beam search
- general technique for reducing search spaces
  - effectiveness depends on scale of non-monotonicity
- future work
  - forest-reranking: parsing with non-local features

Thanks!

try out Cubit

a cube pruning decoder for phrase-based translation

www.cis.upenn.edu/~lhuang3/cubit/