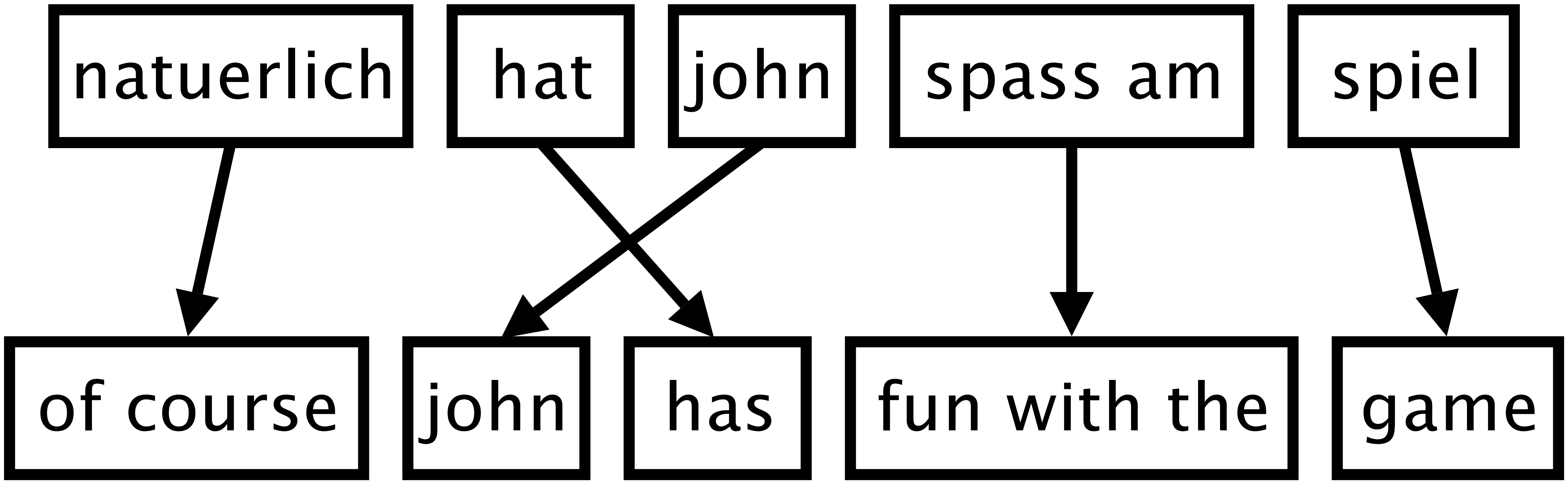
Phrase-based Models for SMT

Taro Watanabe

Why Phrases?

- Use phrases as a unit of translations
  - Directly handle many-to-many word correspondence + local reordering
  - Allow local context + non-compositional phrases
- Employed in many systems, including Google, and open-source, Moses (http://www.statmt.org/moses/)

Phrase-based Model



(An example from Chap. 5, Koehn, 2009)

- Generative story:
  - f is segmented into phrases
  - Each phrase is translated
  - Translated phrases are reordered

Phrase-based Models

$$\begin{aligned}\hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} \frac{\exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\right)}{\sum_{\mathbf{e}', \phi'} \exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}', \phi', \mathbf{f})\right)} \\ &= \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\end{aligned}$$

- Maximization of a log-linear combination of multiple feature functions h(e, Φ, f)
- Φ: phrasal partition of f and e
- w: weight of feature functions

Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

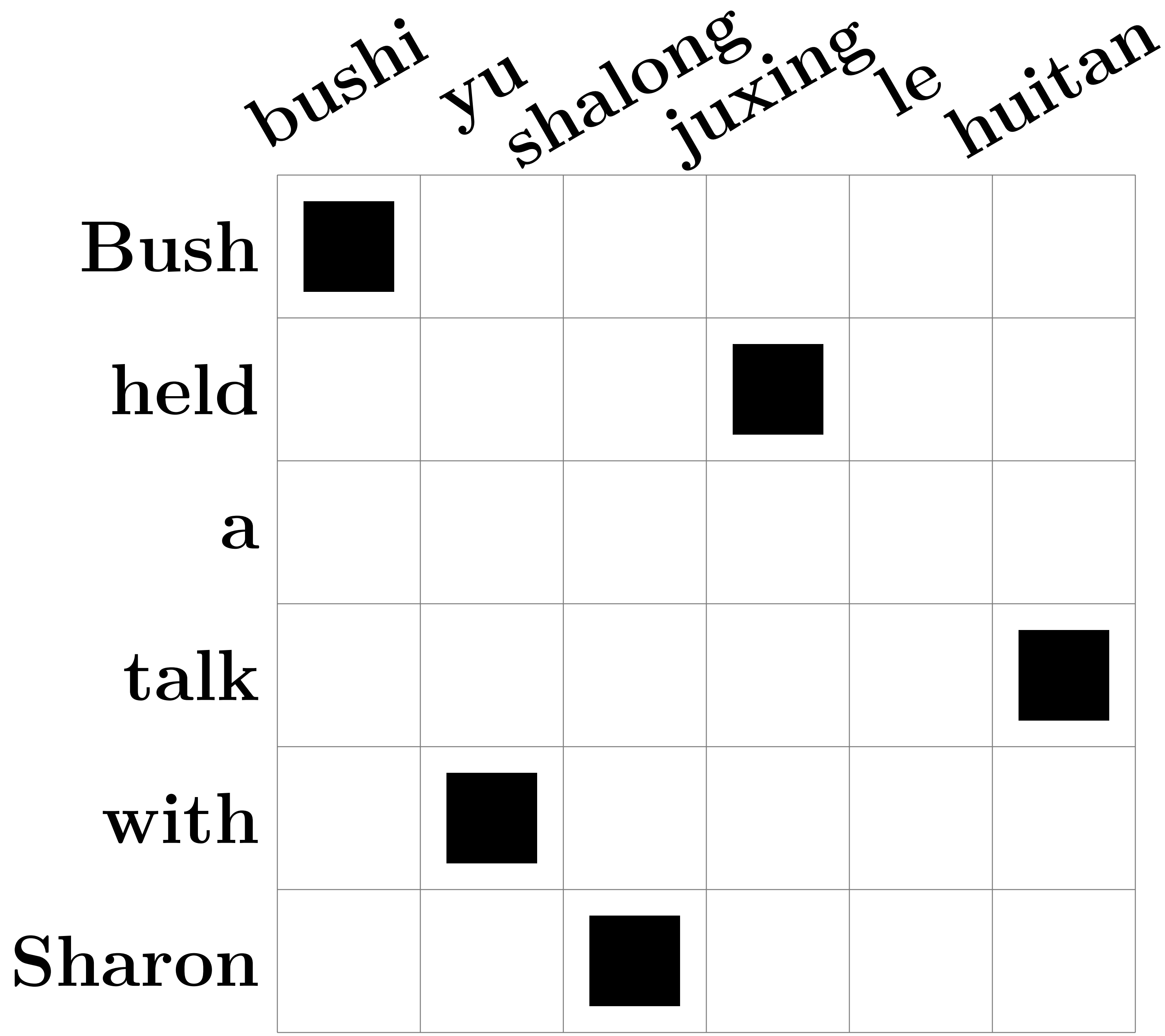
Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

Training

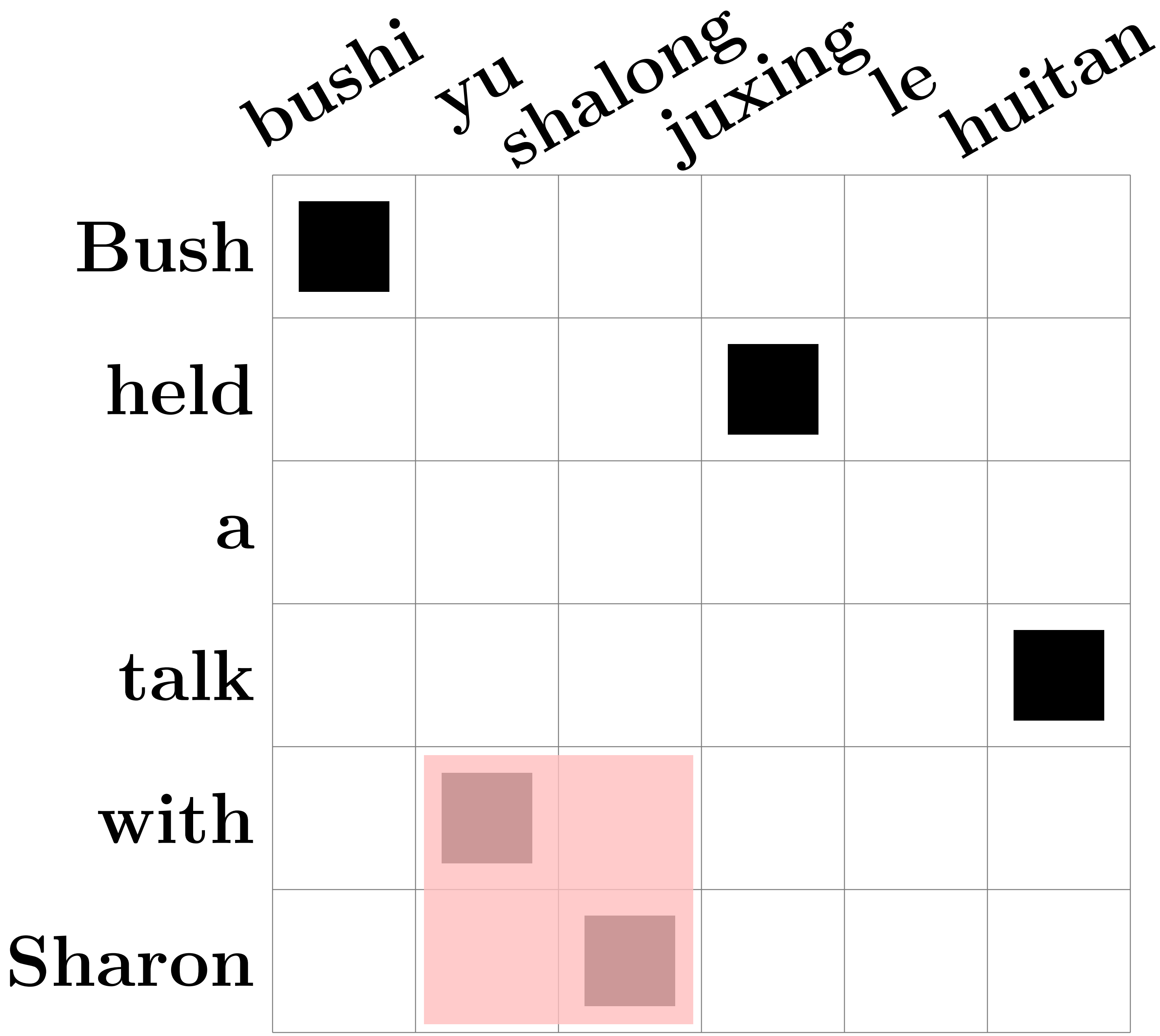
- Learn phrase pairs from $\mathcal{D} = \langle \mathcal{F}, \mathcal{E} \rangle$
- A standard heuristic approach
  - Compute word alignment
  - Extract phrase pairs
  - Score phrases

Word alignment



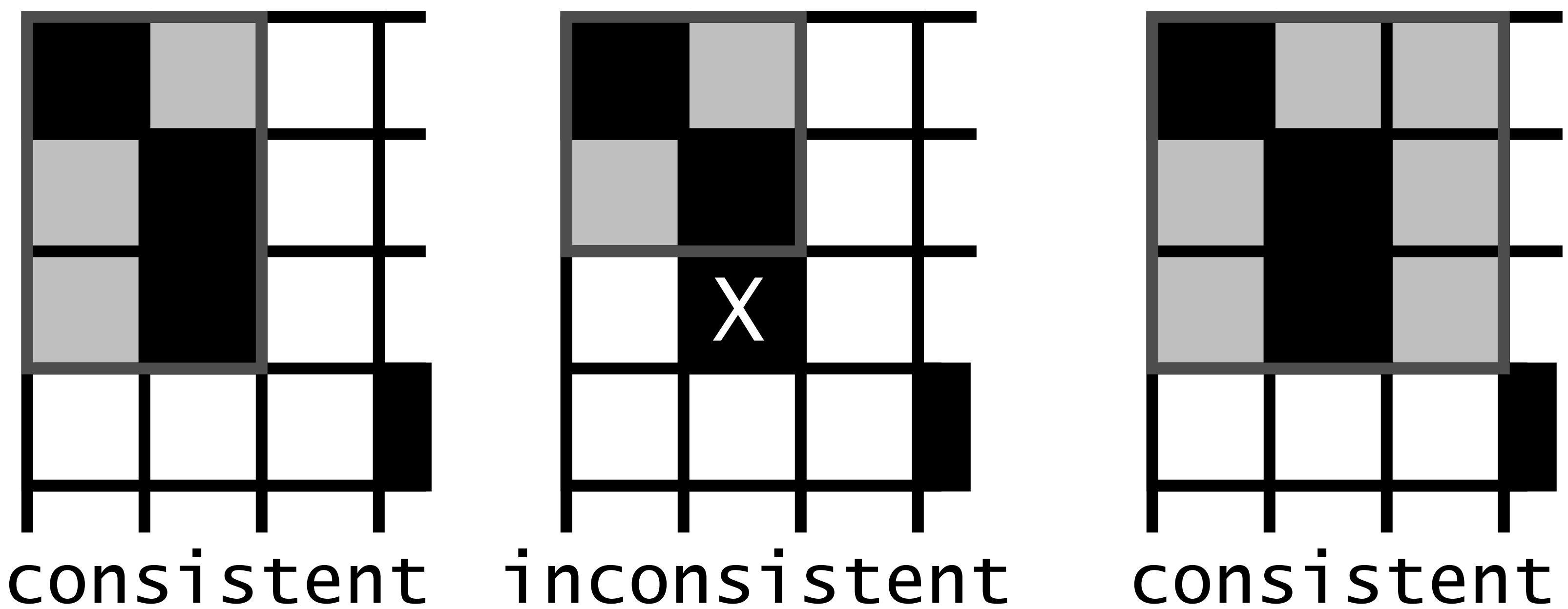
(Example from Huang and Chiang, 2007)

Extract Phrase Pairs



- From word alignment, extract a phrase pair consistent with word alignment

Consistent Phrases

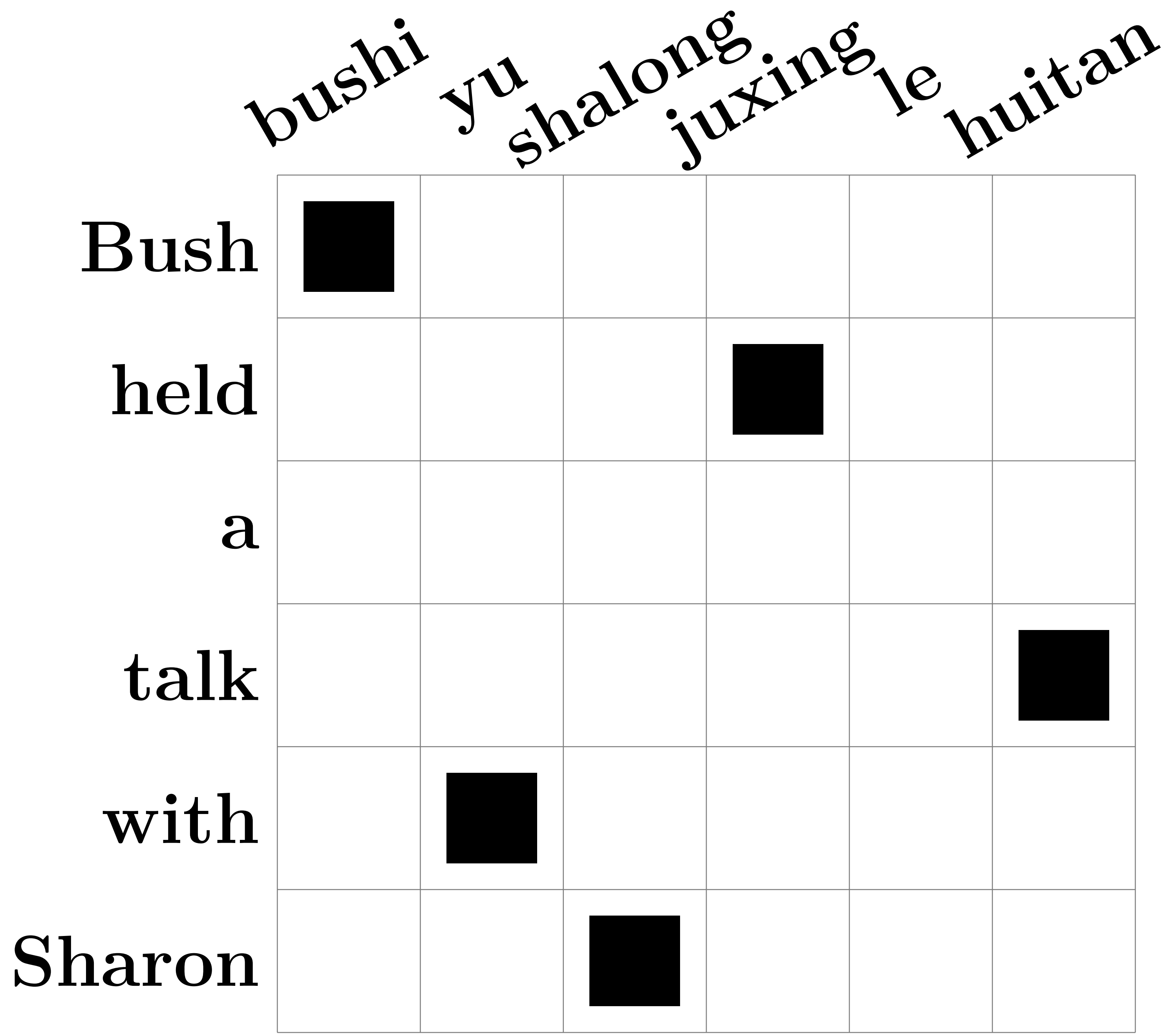


(Example from Koehn, 2009)

- a phrase pair (f, e) is consistent:

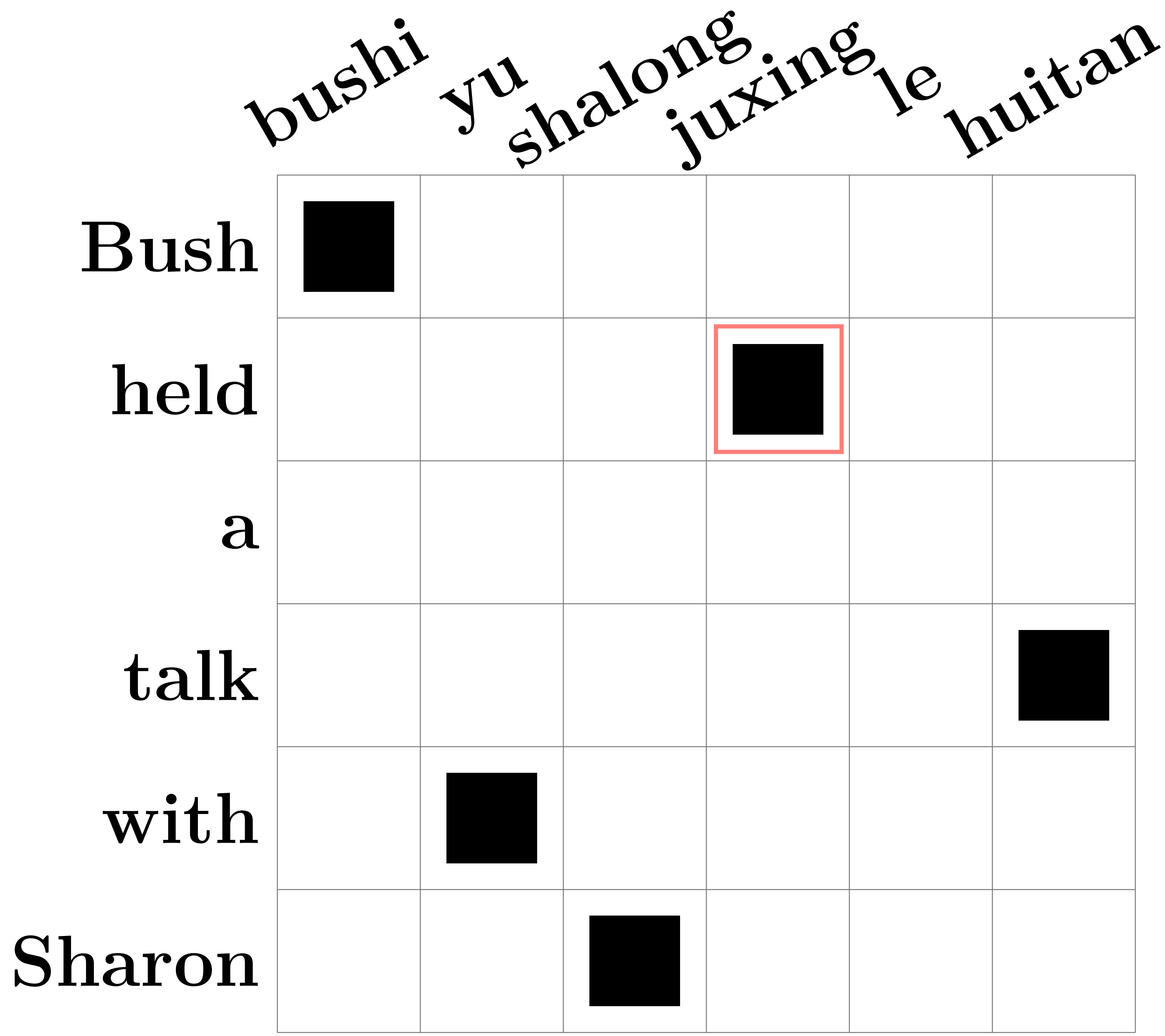
$$\forall e_i \in \bar{\mathbf{e}} : (e_i, f_j) \in \mathbf{a} \rightarrow f_j \in \bar{\mathbf{f}}$$
$$\forall f_j \in \bar{\mathbf{f}} : (e_j, f_j) \in \mathbf{a} \rightarrow e_i \in \bar{\mathbf{e}}$$
$$\exists e_i \in \bar{\mathbf{e}}, f_j \in \bar{\mathbf{f}} : (e_j, f_j) \in \mathbf{a}$$

Exhaustive Extraction



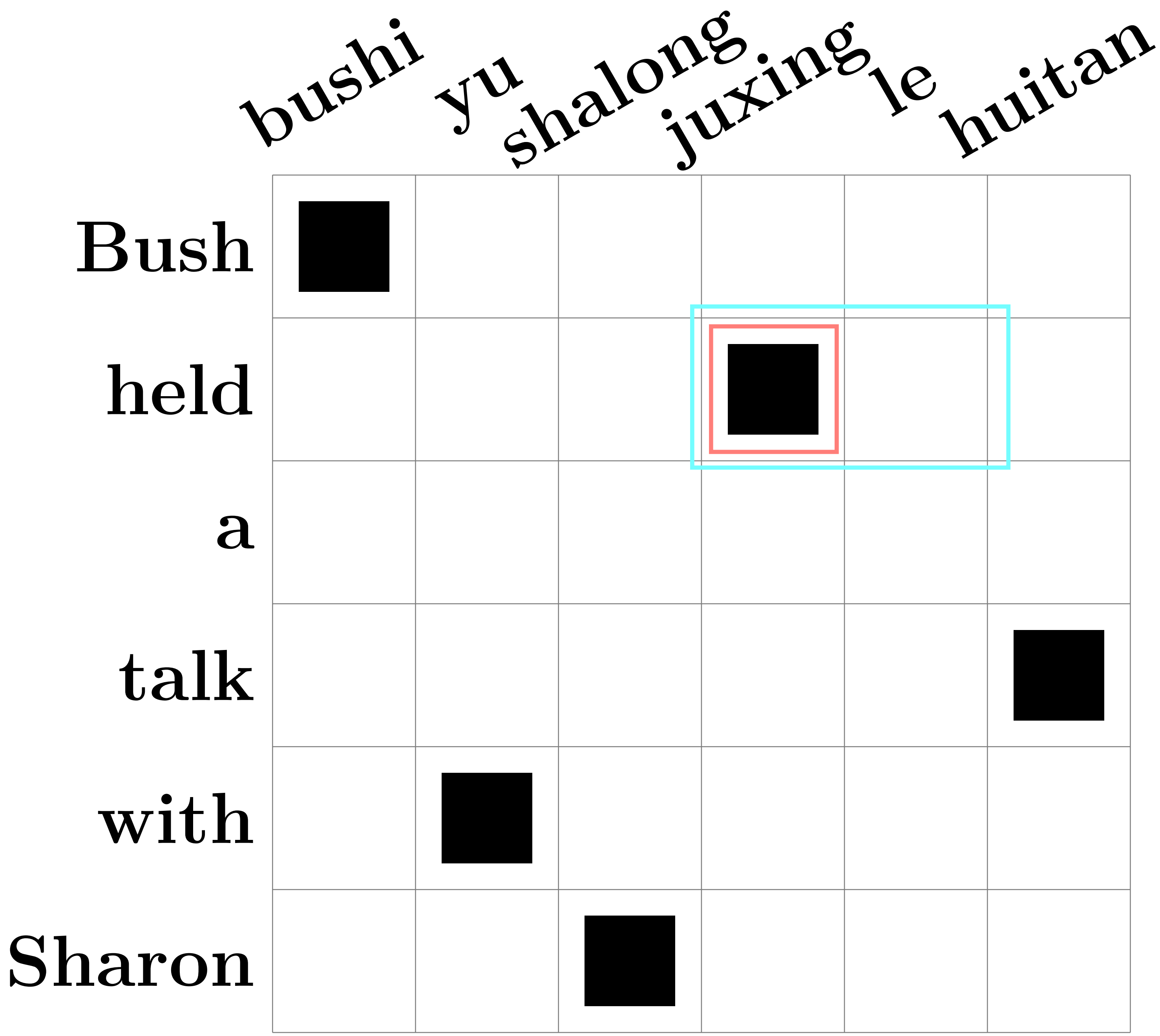
- Exhaustively extract phrases from f, e

Exhaustive Extraction



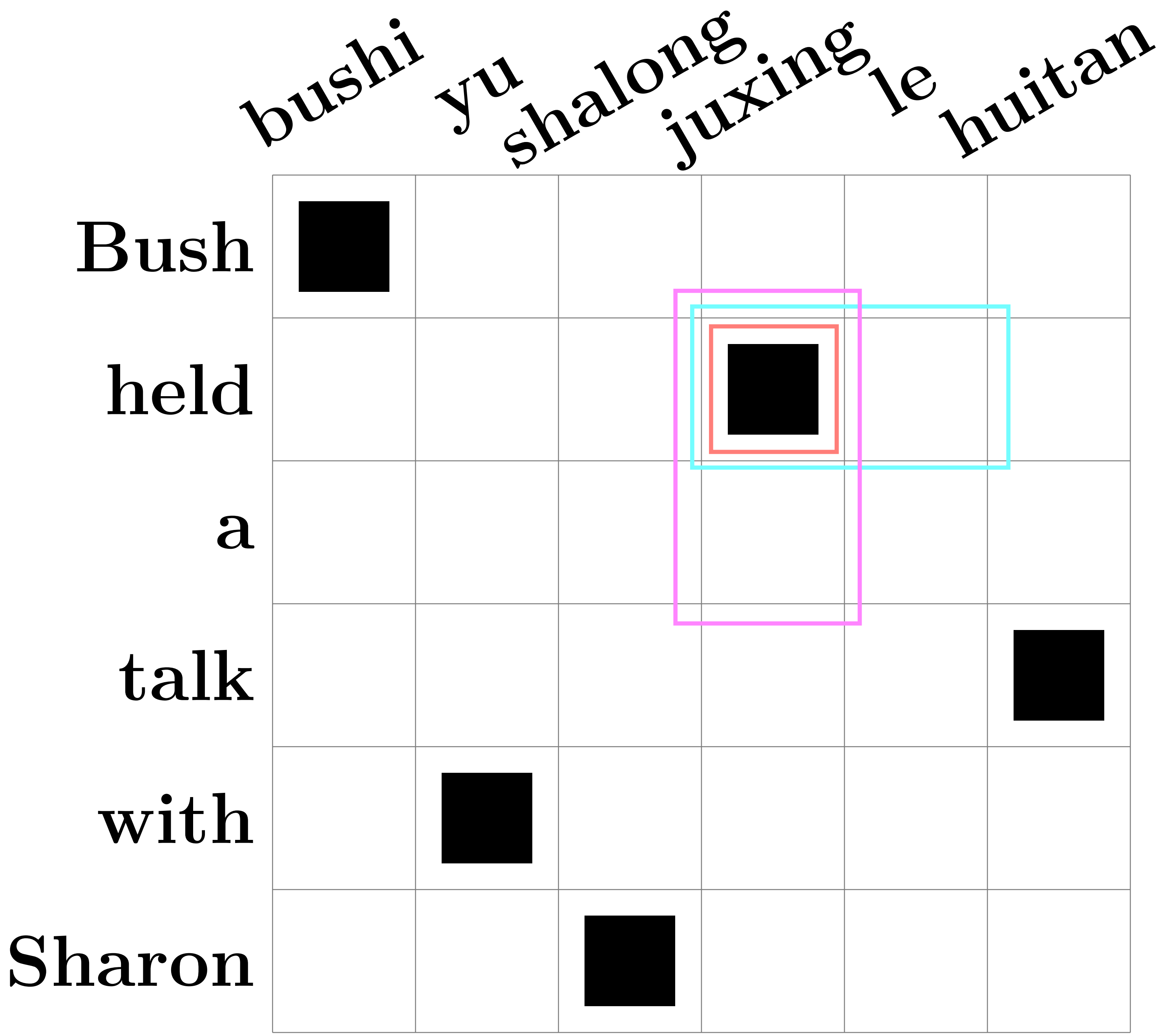
- Exhaustively extract phrases from f, e

Exhaustive Extraction



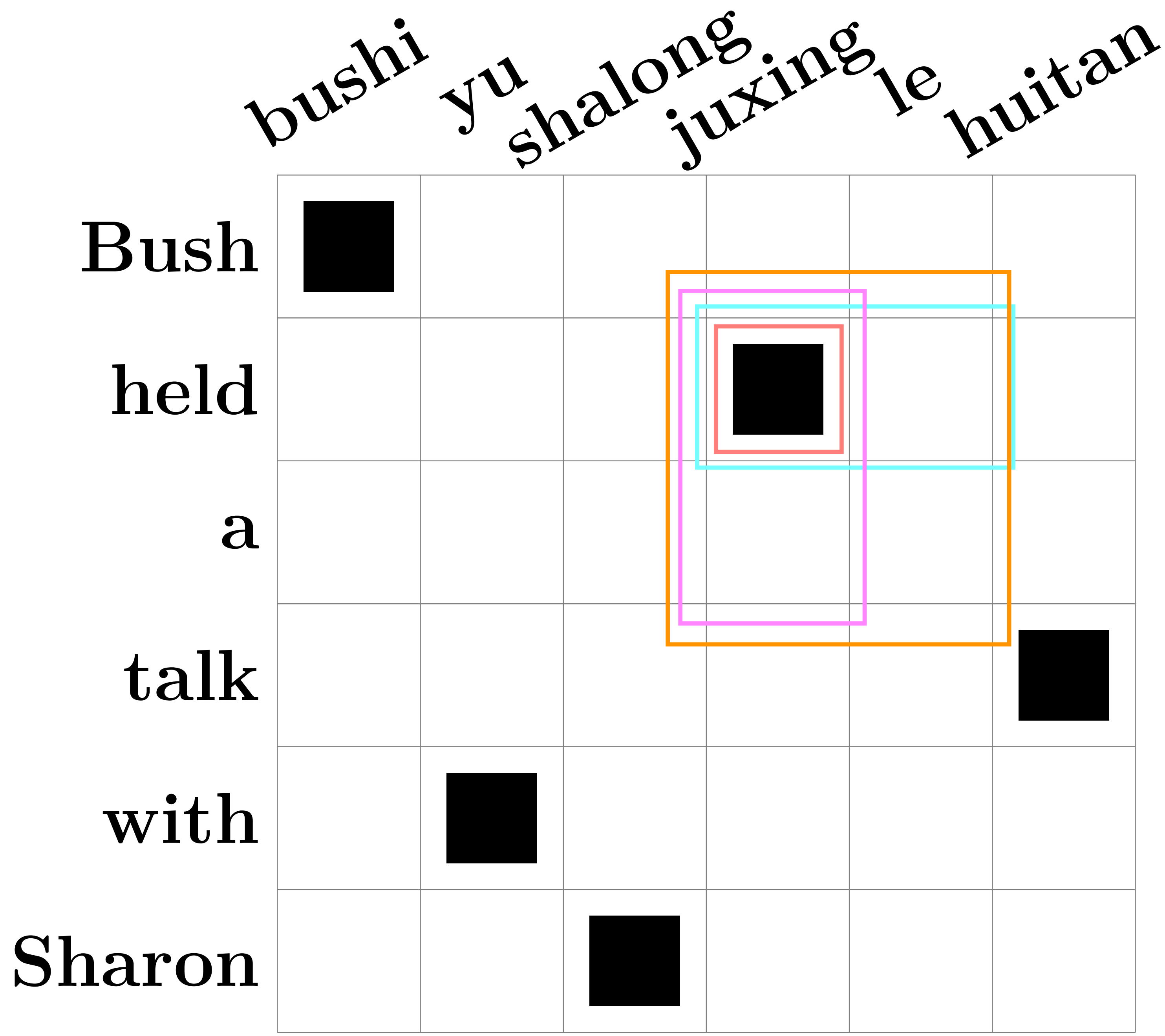
- Exhaustively extract phrases from f, e

Exhaustive Extraction



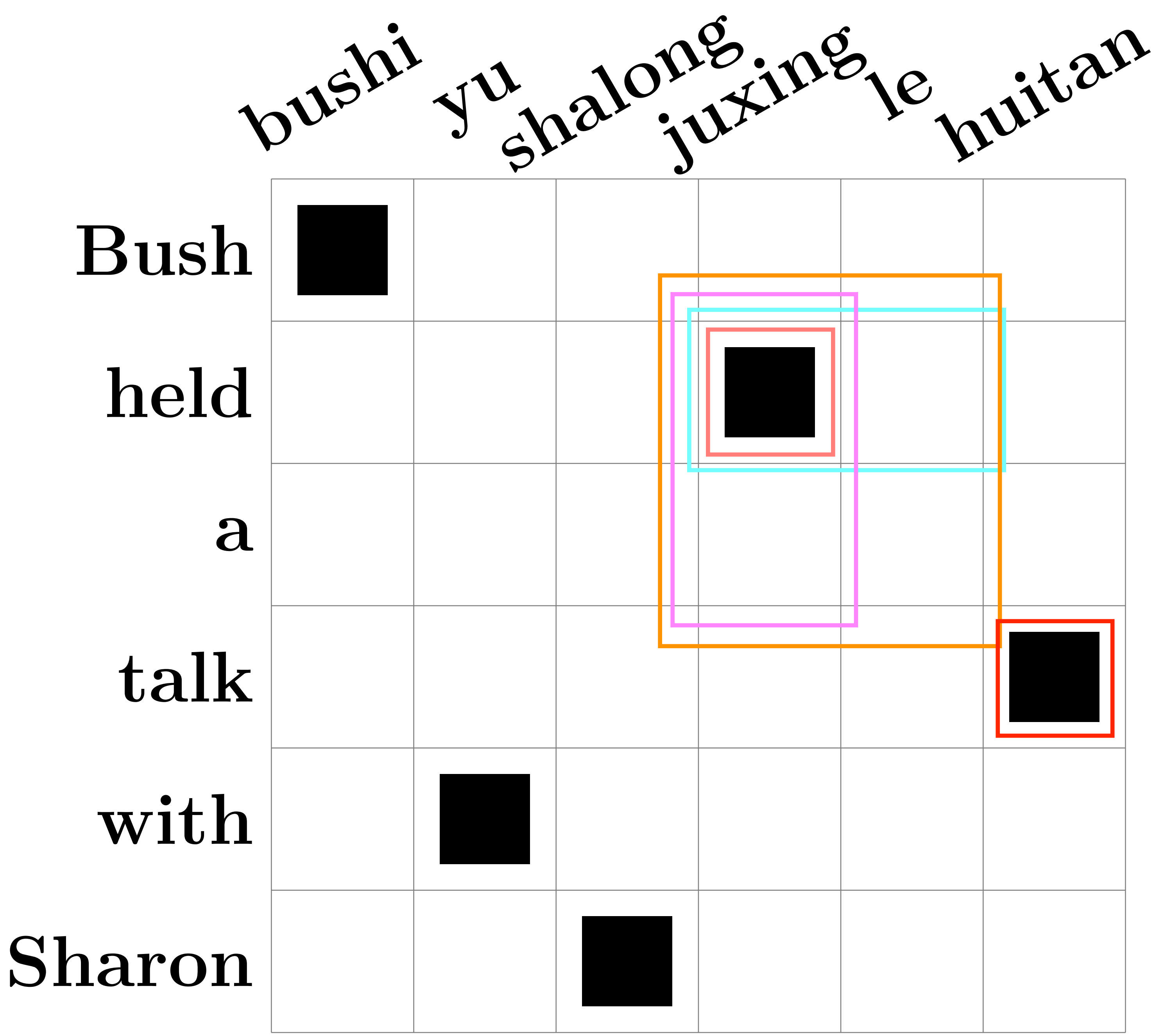
- Exhaustively extract phrases from f, e

Exhaustive Extraction



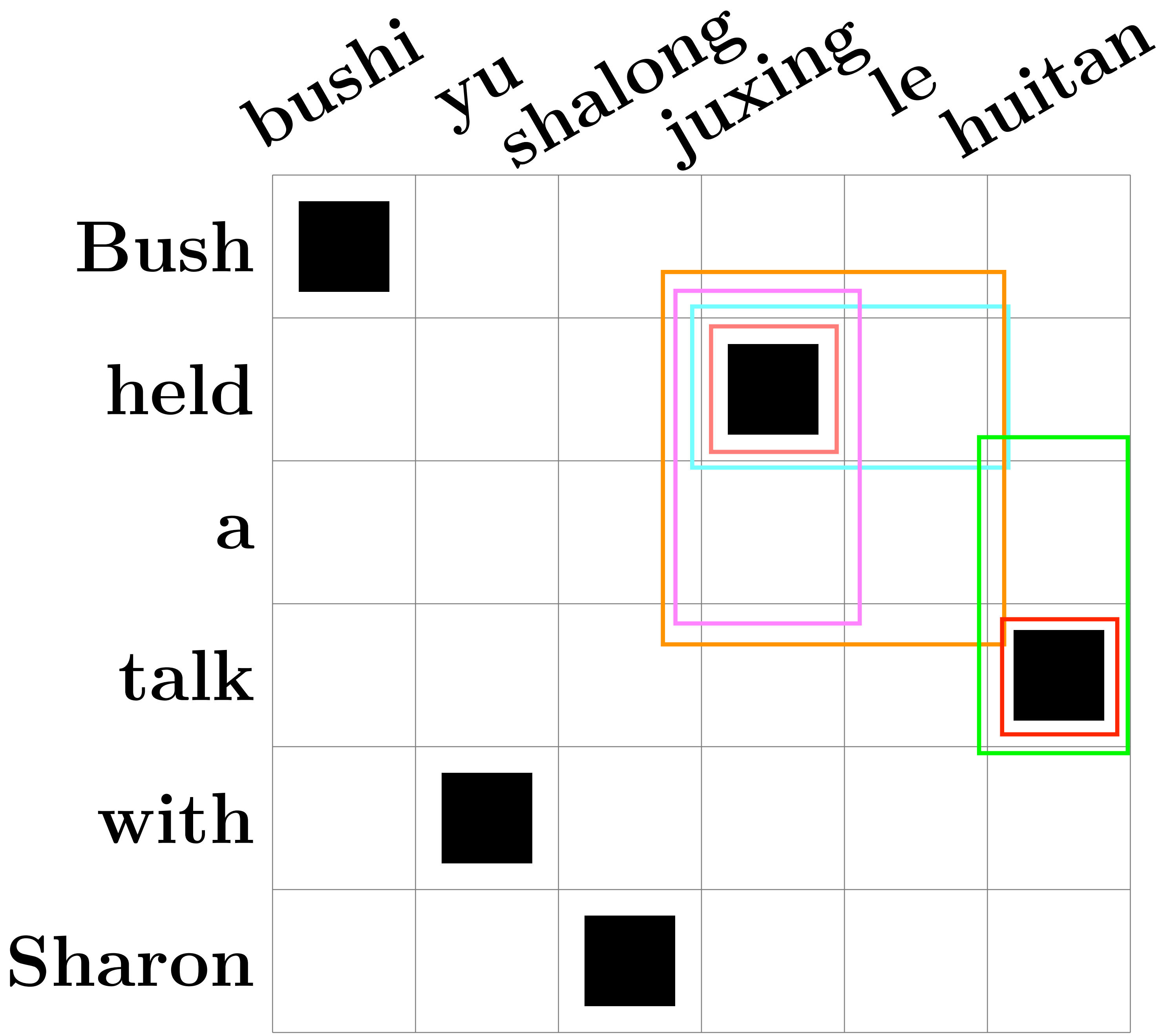
- Exhaustively extract phrases from f, e

Exhaustive Extraction



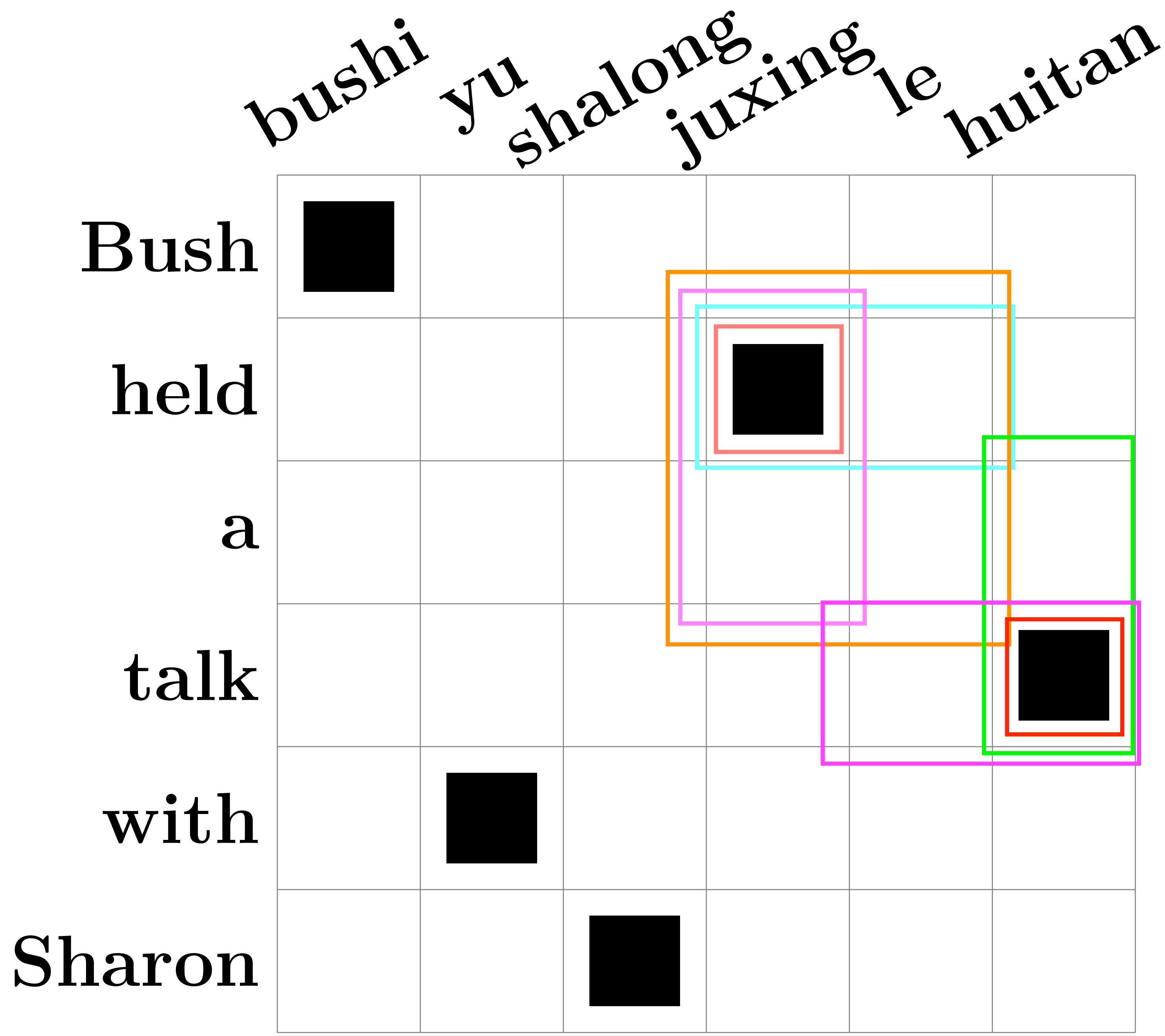
- Exhaustively extract phrases from f, e

Exhaustive Extraction



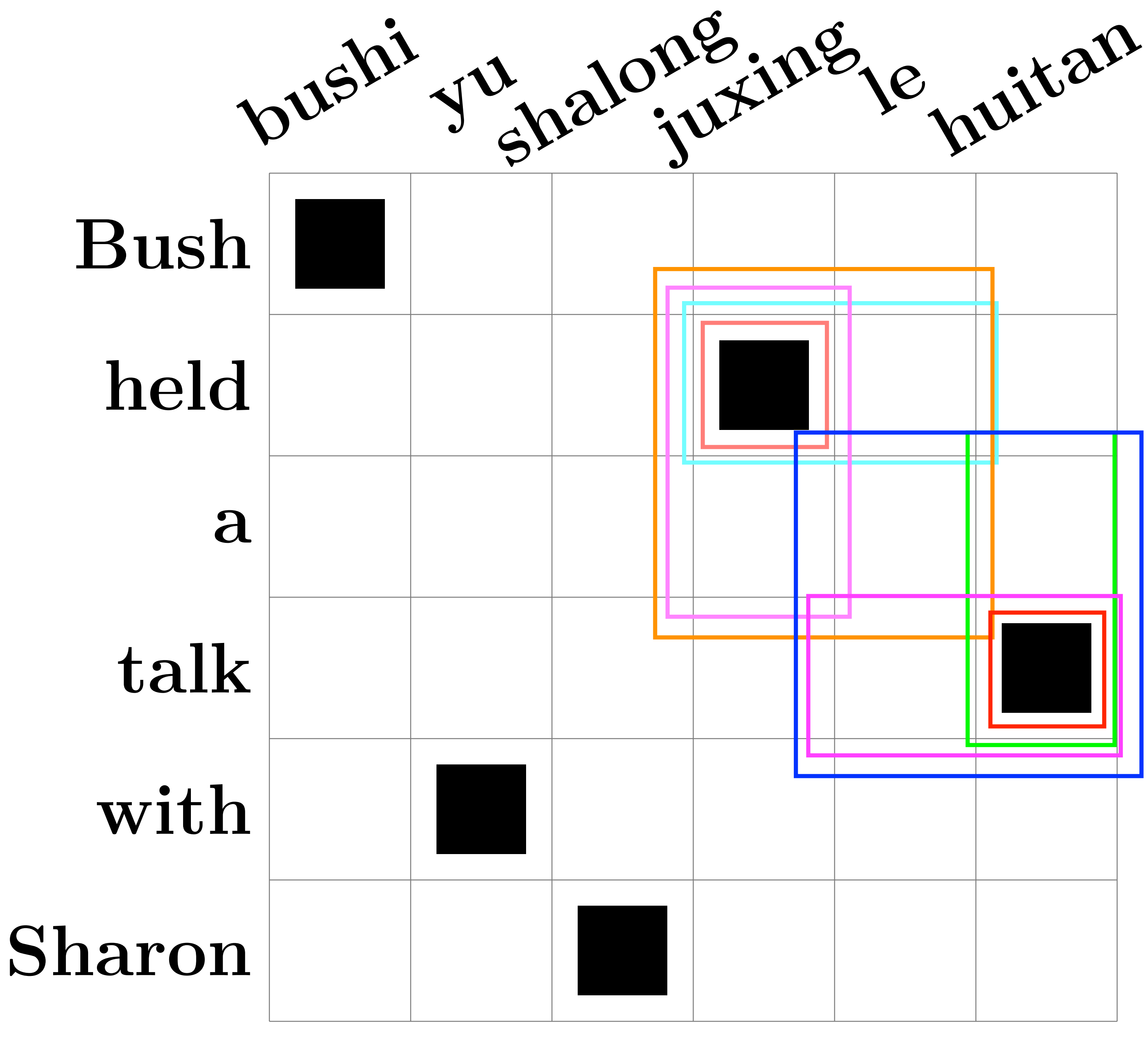
- Exhaustively extract phrases from f, e

Exhaustive Extraction



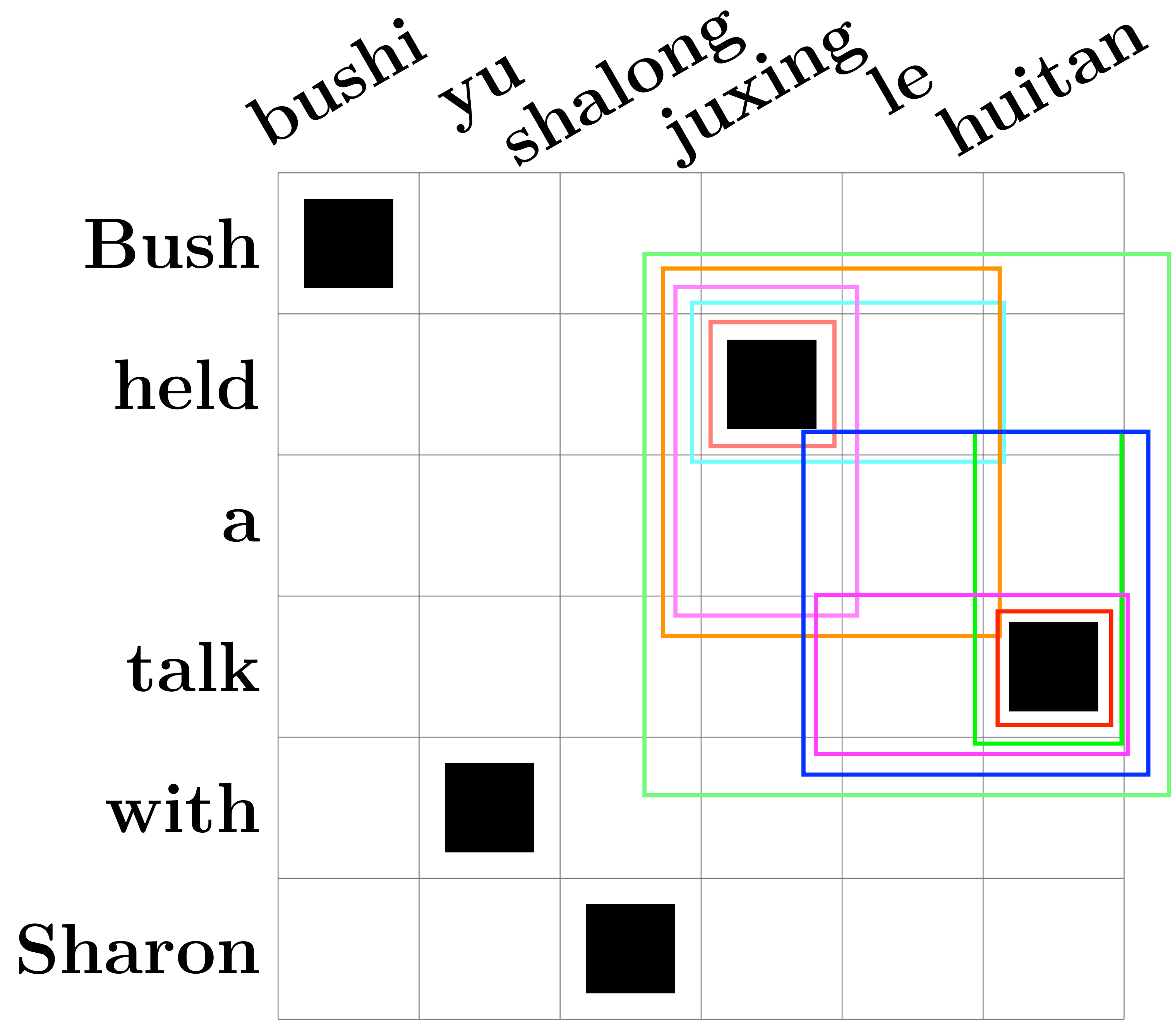
- Exhaustively extract phrases from f, e

Exhaustive Extraction



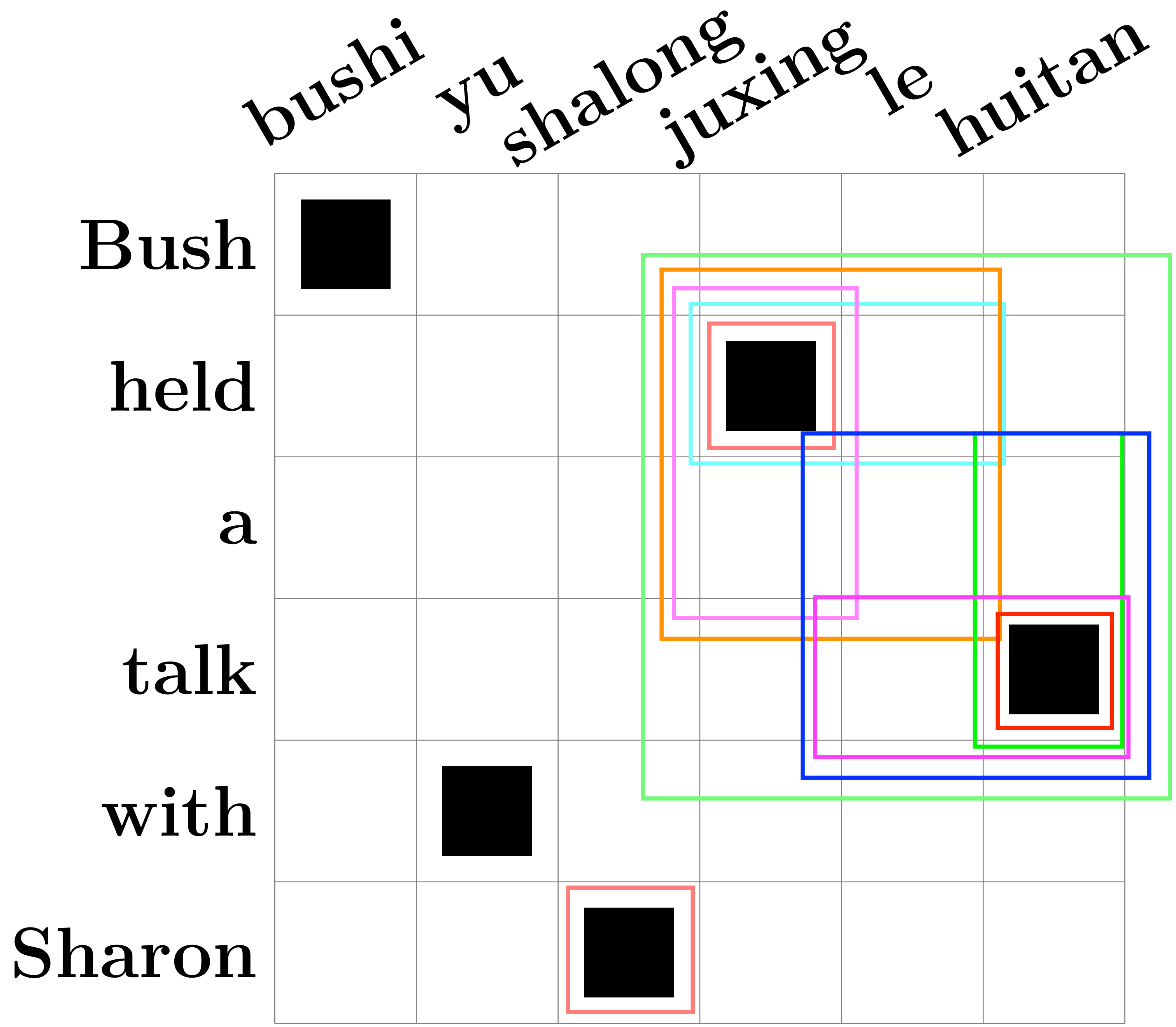
- Exhaustively extract phrases from f, e

Exhaustive Extraction



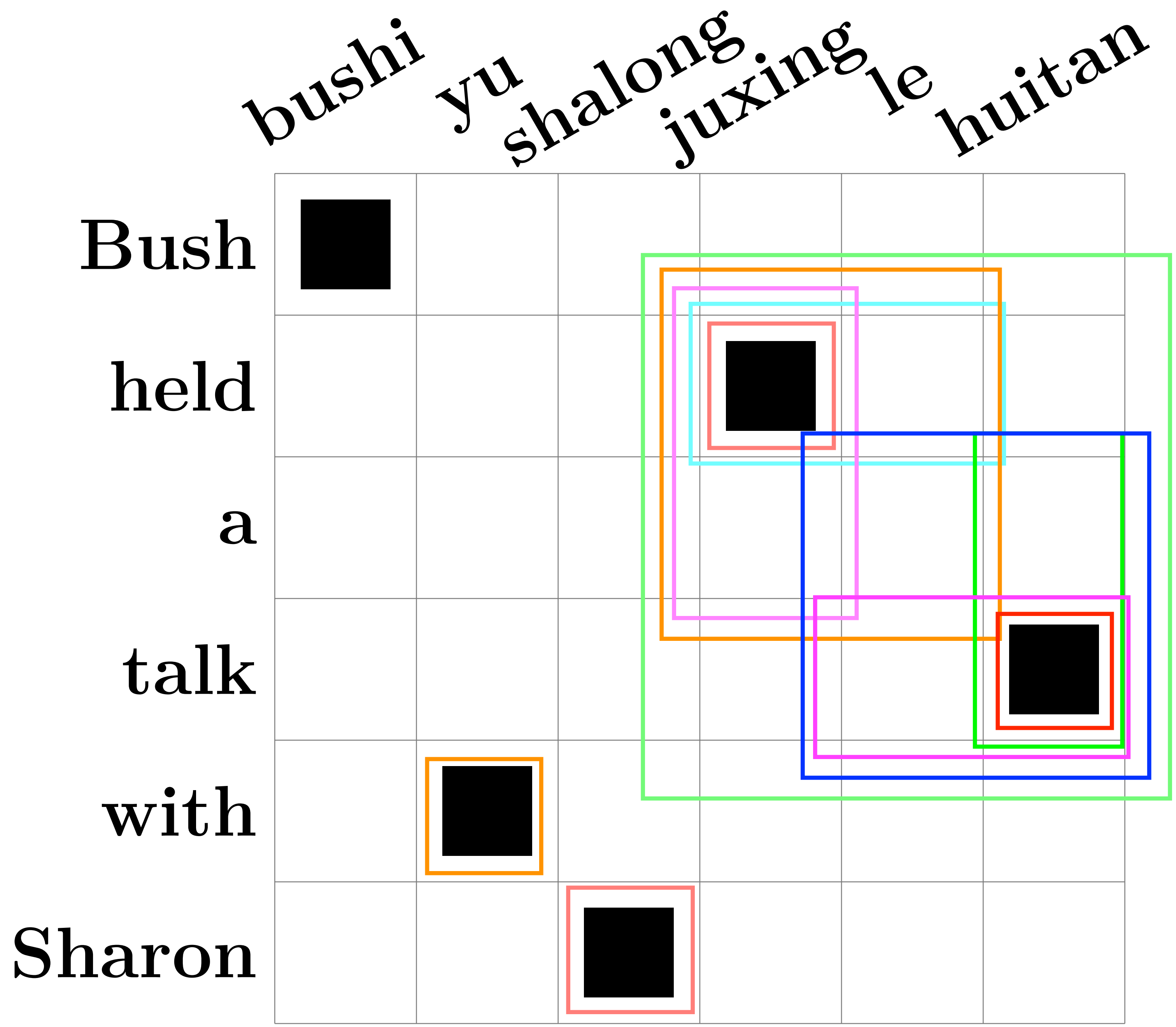
- Exhaustively extract phrases from f, e

Exhaustive Extraction



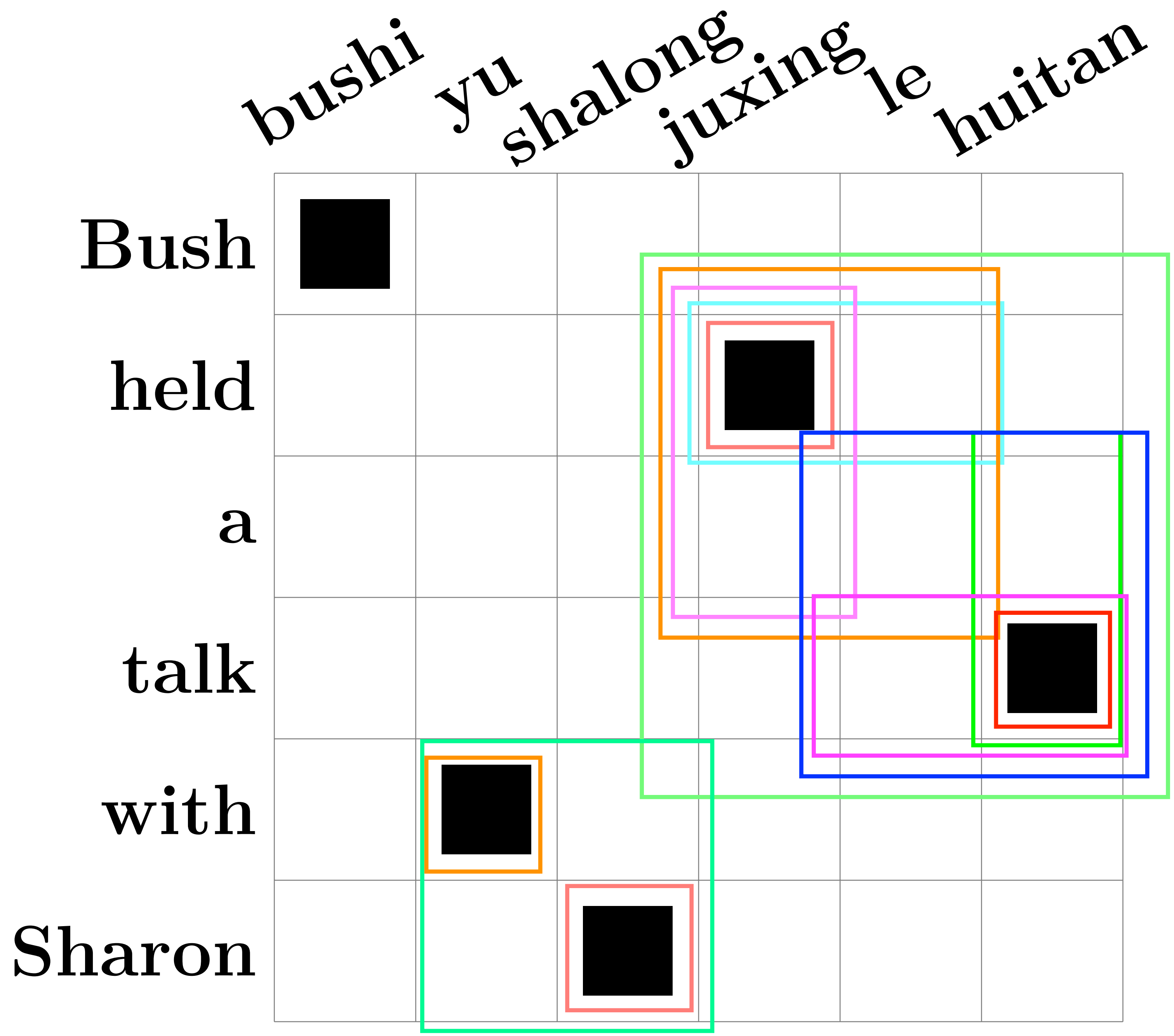
- Exhaustively extract phrases from f, e

Exhaustive Extraction



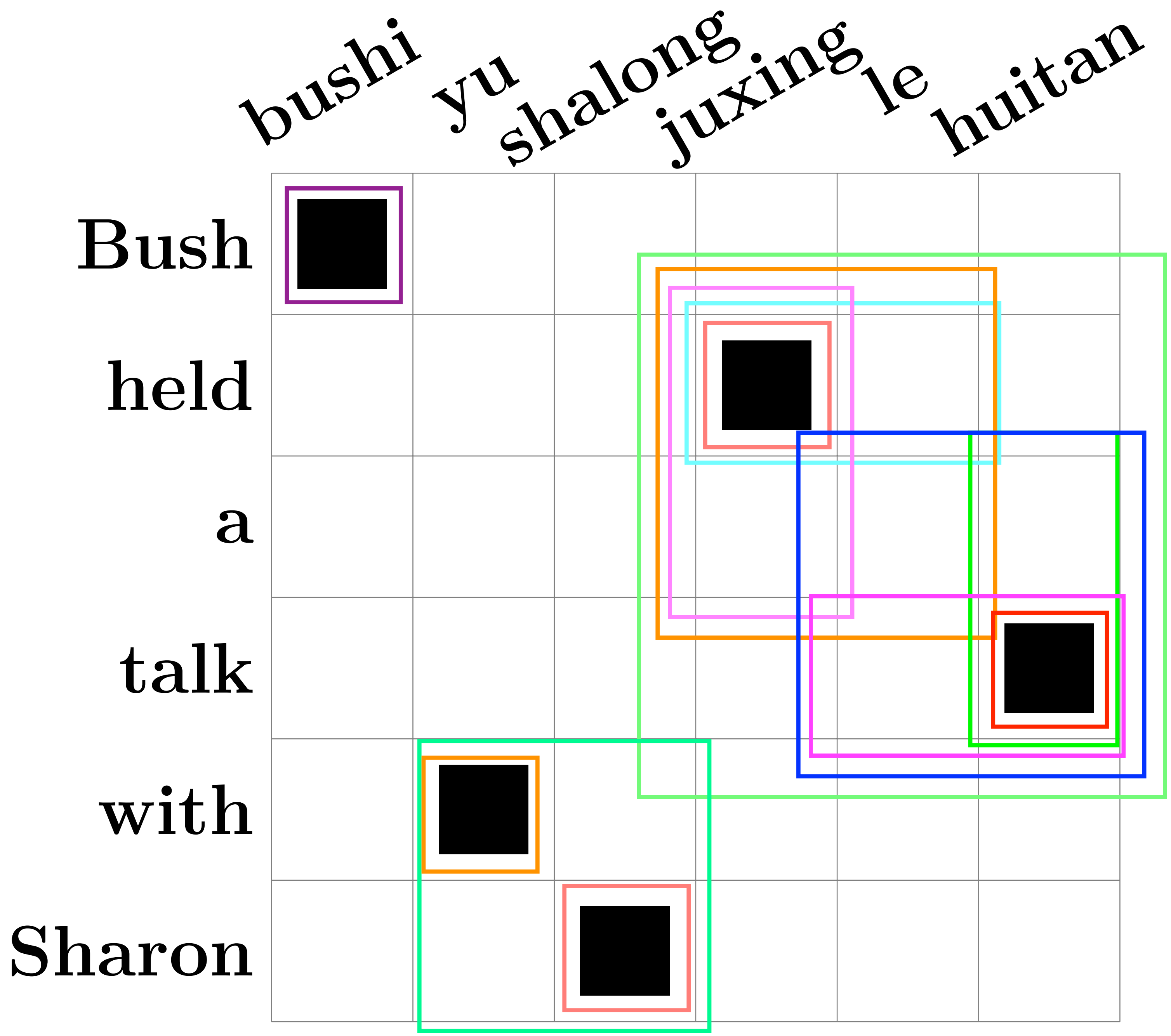
- Exhaustively extract phrases from f, e

Exhaustive Extraction



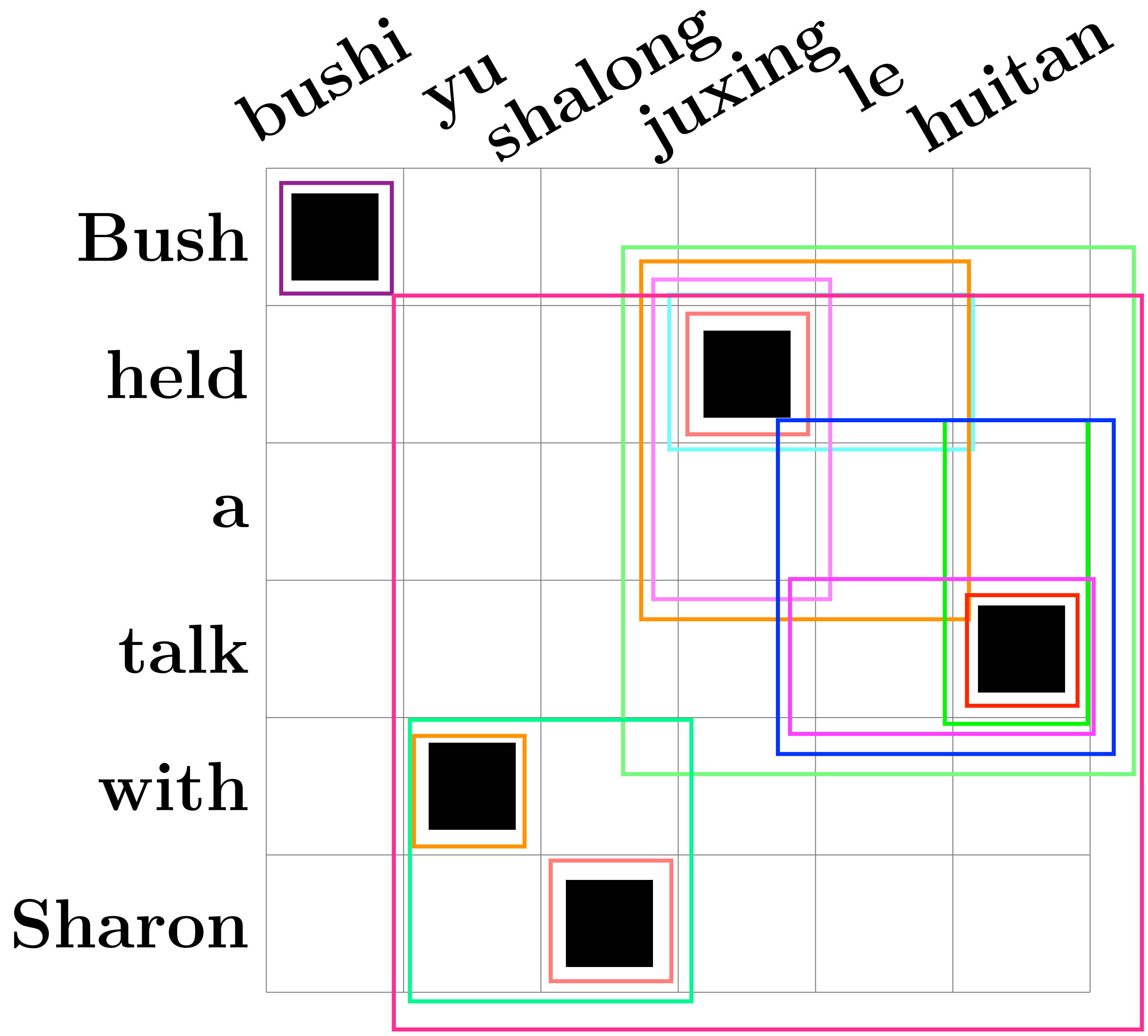
- Exhaustively extract phrases from f, e

Exhaustive Extraction



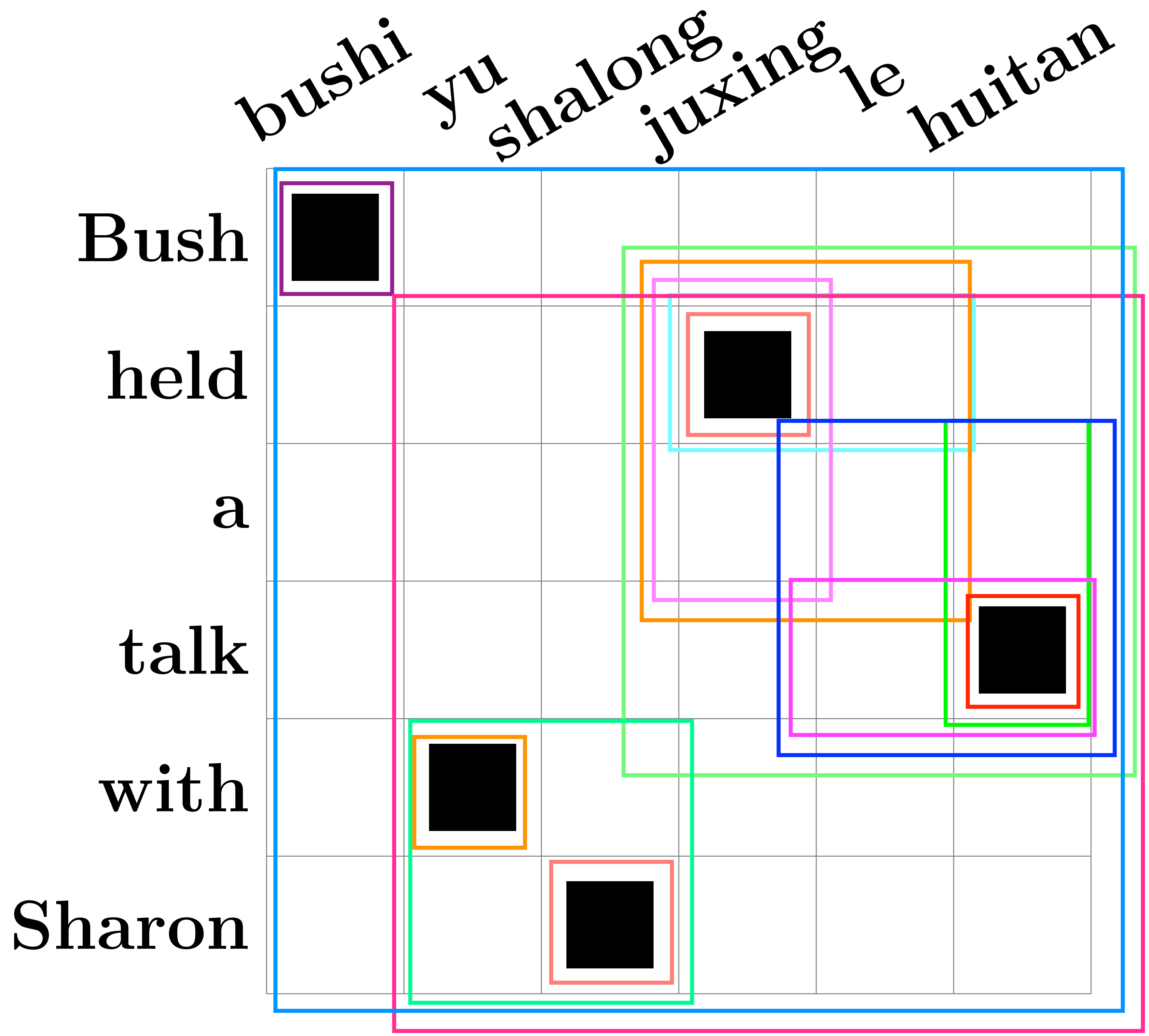
- Exhaustively extract phrases from f, e

Exhaustive Extraction



- Exhaustively extract phrases from f, e

Exhaustive Extraction



- Exhaustively extract phrases from f, e

Features from Phrases

$$
\begin{aligned}
\log p_\phi(\bar{\mathbf{f}}|\bar{\mathbf{e}}) &= \log \frac{\mathrm{count}(\bar{\mathbf{e}}, \bar{\mathbf{f}})}{\sum_{\bar{\mathbf{f}}'} \mathrm{count}(\bar{\mathbf{e}}, \bar{\mathbf{f}}')} \\
\log p_\phi(\bar{\mathbf{e}}|\bar{\mathbf{f}}) &= \log \frac{\mathrm{count}(\bar{\mathbf{e}}, \bar{\mathbf{f}})}{\sum_{\bar{\mathbf{e}}'} \mathrm{count}(\bar{\mathbf{e}}', \bar{\mathbf{f}})}
\end{aligned}
$$

- Collect all the phrase pairs from the data
- Maximum likelihood estimates by relative frequencies
- Employ scores in two directions

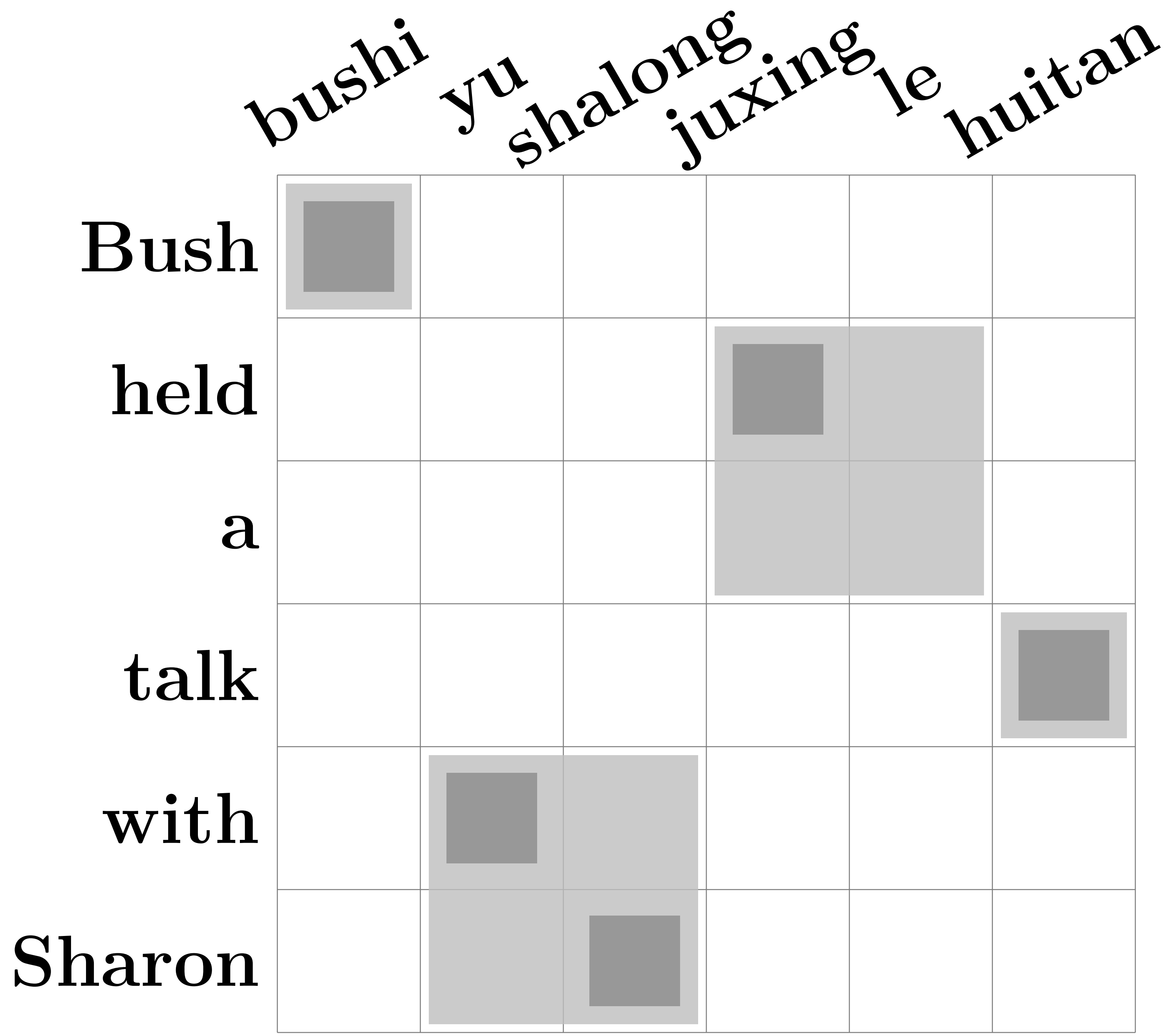
Features from Alignment

$$\log p_{lex}(\bar{\mathbf{f}}|\bar{\mathbf{e}},\bar{\mathbf{a}}) = \log \prod_{i}^{|\bar{\mathbf{e}}|} \frac{1}{|\{j|(i,j)\in\bar{\mathbf{a}}\}|} \sum_{\forall(i,j)\in\bar{\mathbf{a}}} t(e_i|f_j)$$

$$\log p_{lex}(\bar{\mathbf{e}}|\bar{\mathbf{f}},\bar{\mathbf{a}}) = \log \prod_{j}^{|\bar{\mathbf{f}}|} \frac{1}{|\{i|(j,i)\in\bar{\mathbf{a}}\}|} \sum_{\forall(j,i)\in\bar{\mathbf{a}}} t(f_j|e_i)$$

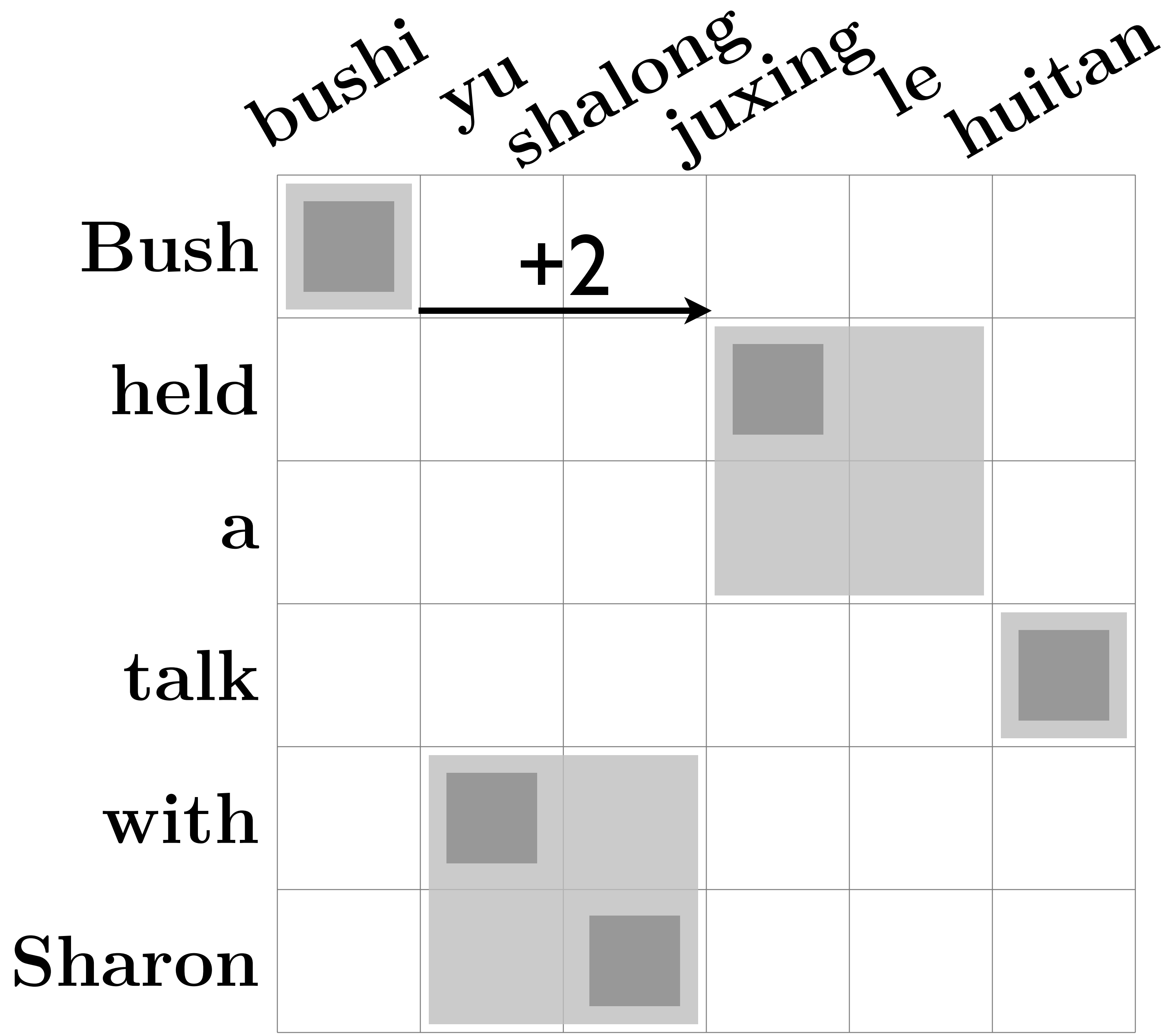
- Lexical weighing which scores by word translation probabilities
- Idea: counts for rare phrase pairs are unreliable
  - Smoothing effect by decomposing into word pairs

Features for Distortion



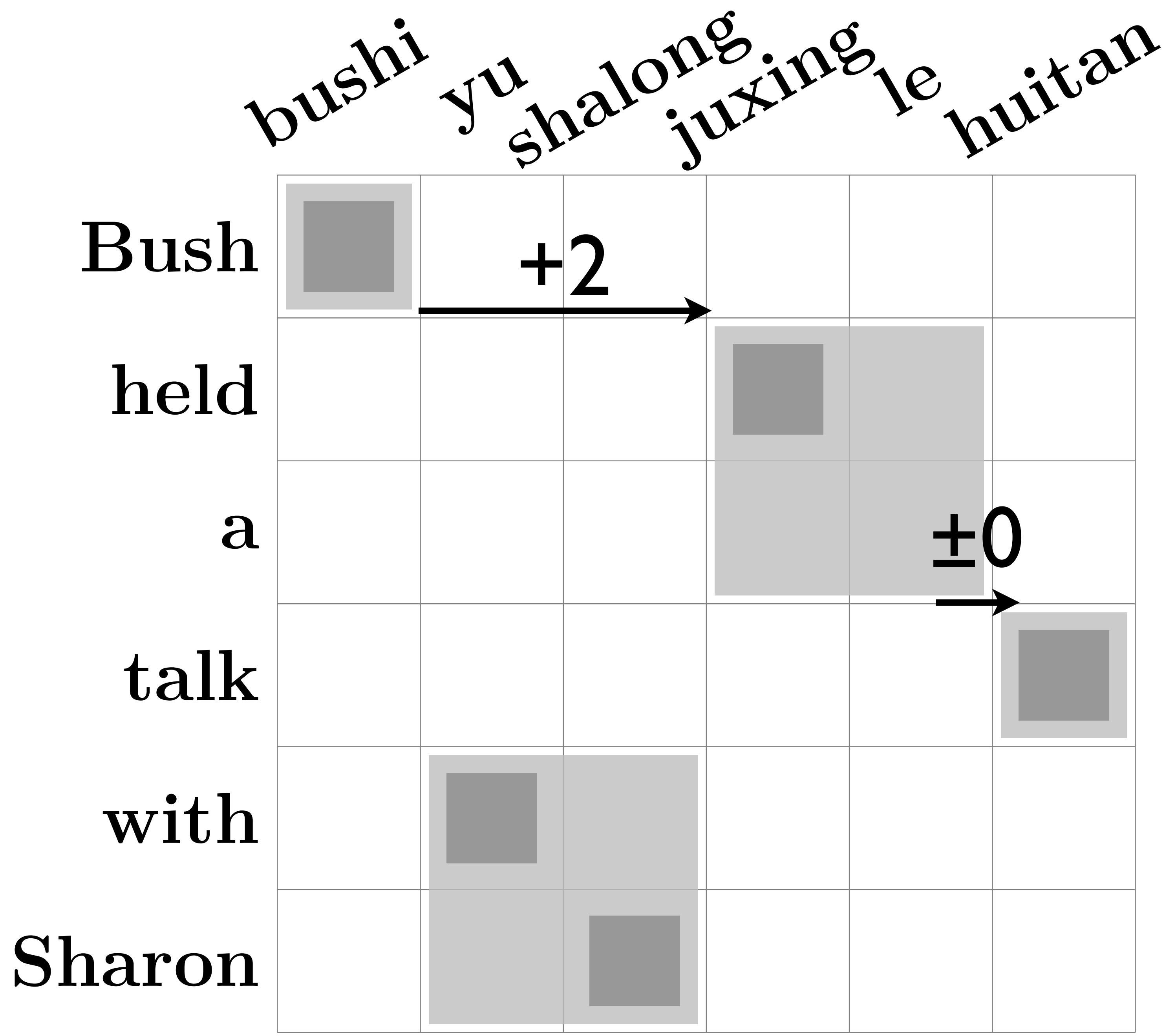
- Distance-based distortion modeling

Features for Distortion



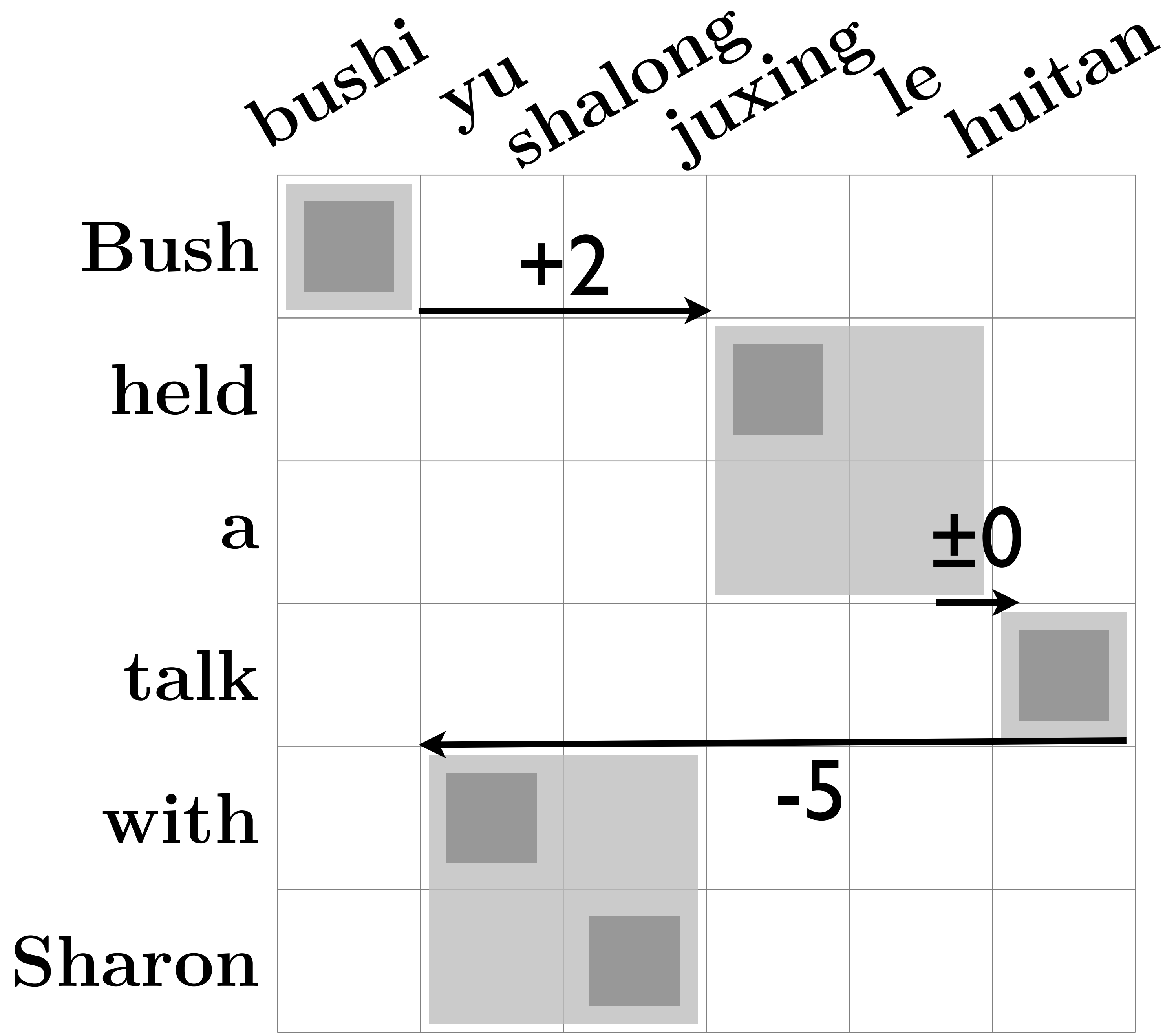
- Distance-based distortion modeling

Features for Distortion



- Distance-based distortion modeling

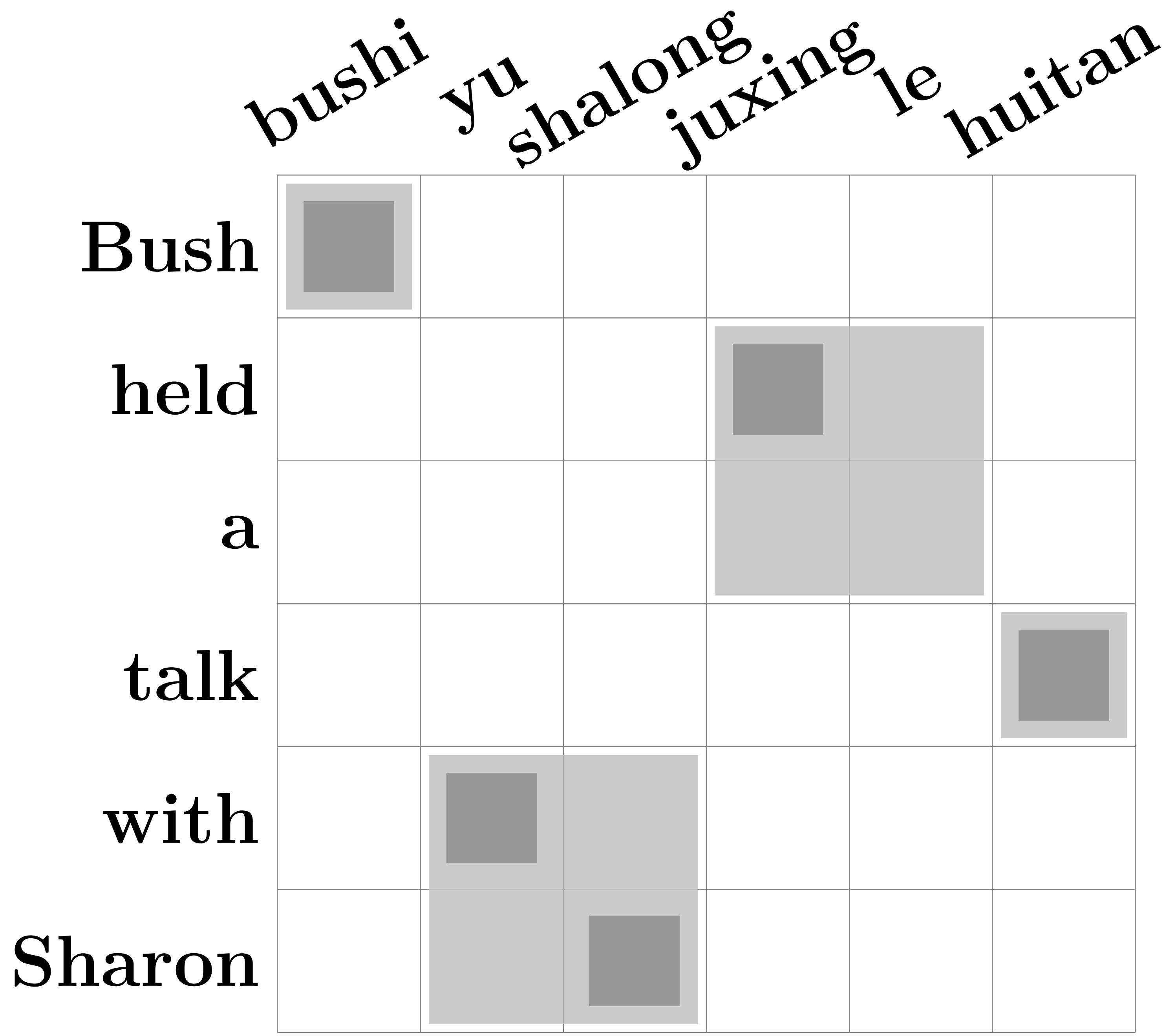
Features for Distortion



- Distance-based distortion modeling

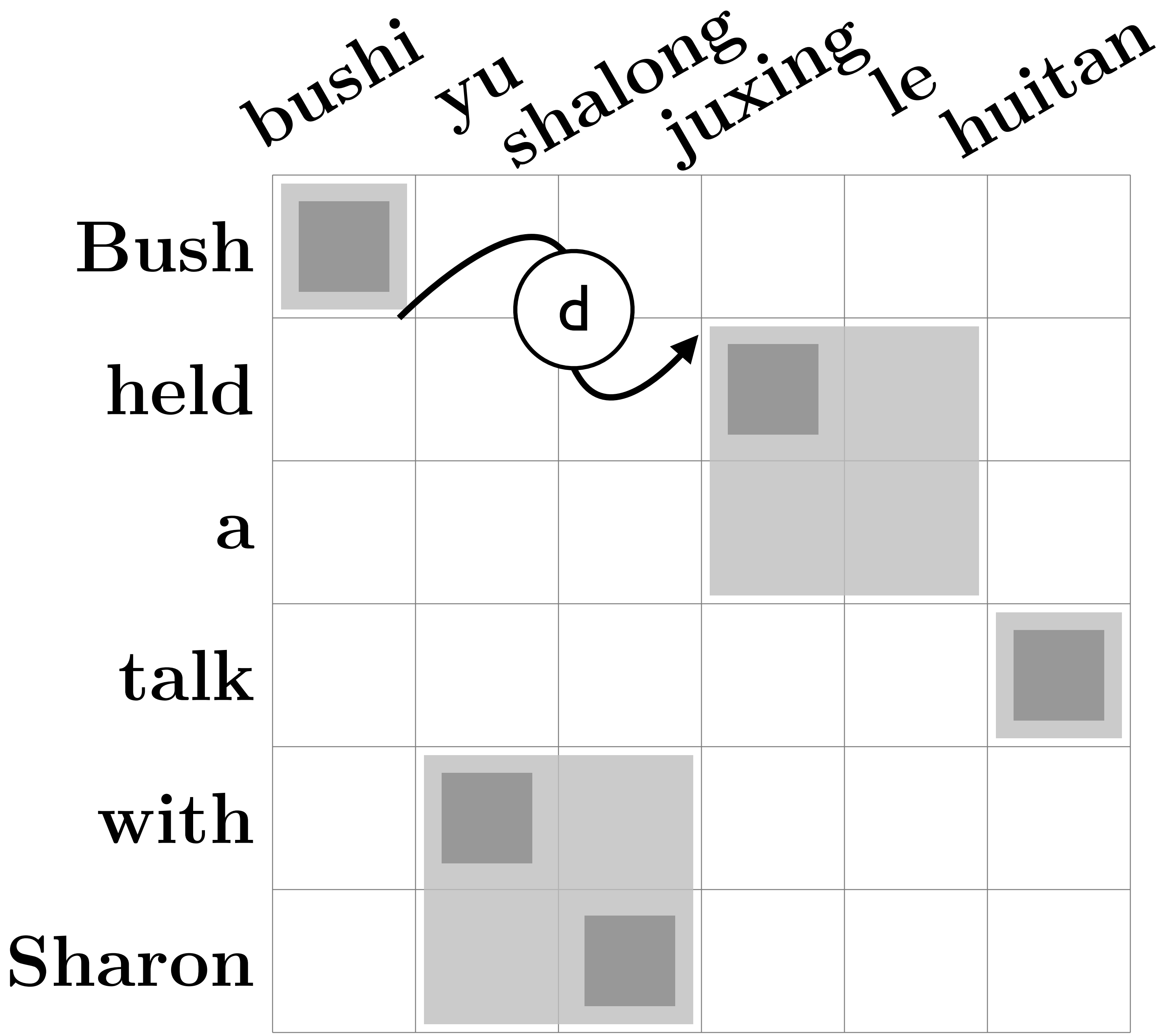
$$d(\mathbf{f}, \phi, \mathbf{e}) = |+2| + |0| + |-5| = 7$$

Features for Reordering



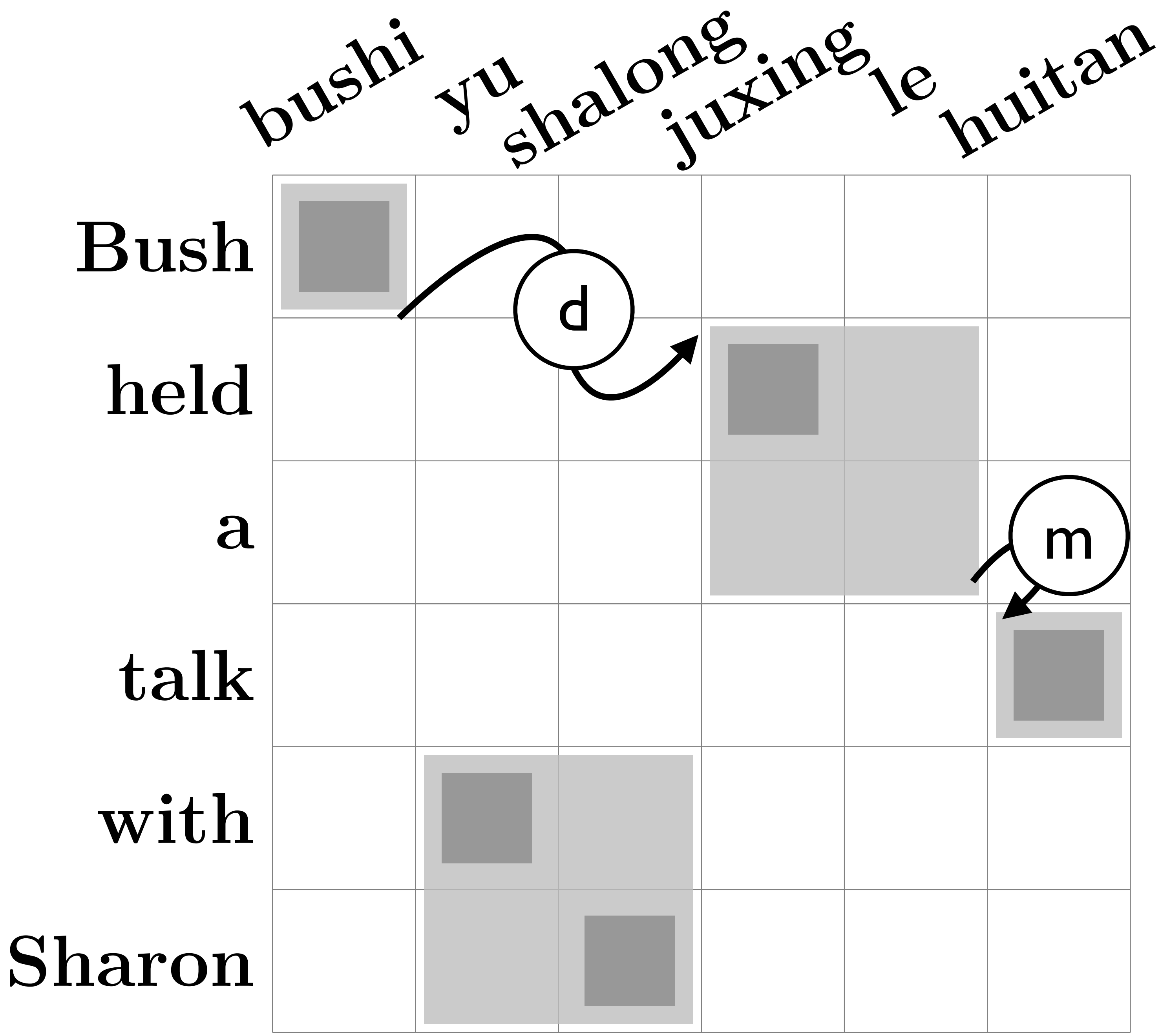
- Fine grained reordering features: $\log p_o(o \in \{m, s, d\} \,|\bar{\mathbf{f}}, \bar{\mathbf{e}})$
- Either monotone, swap, discontinuous

Features for Reordering



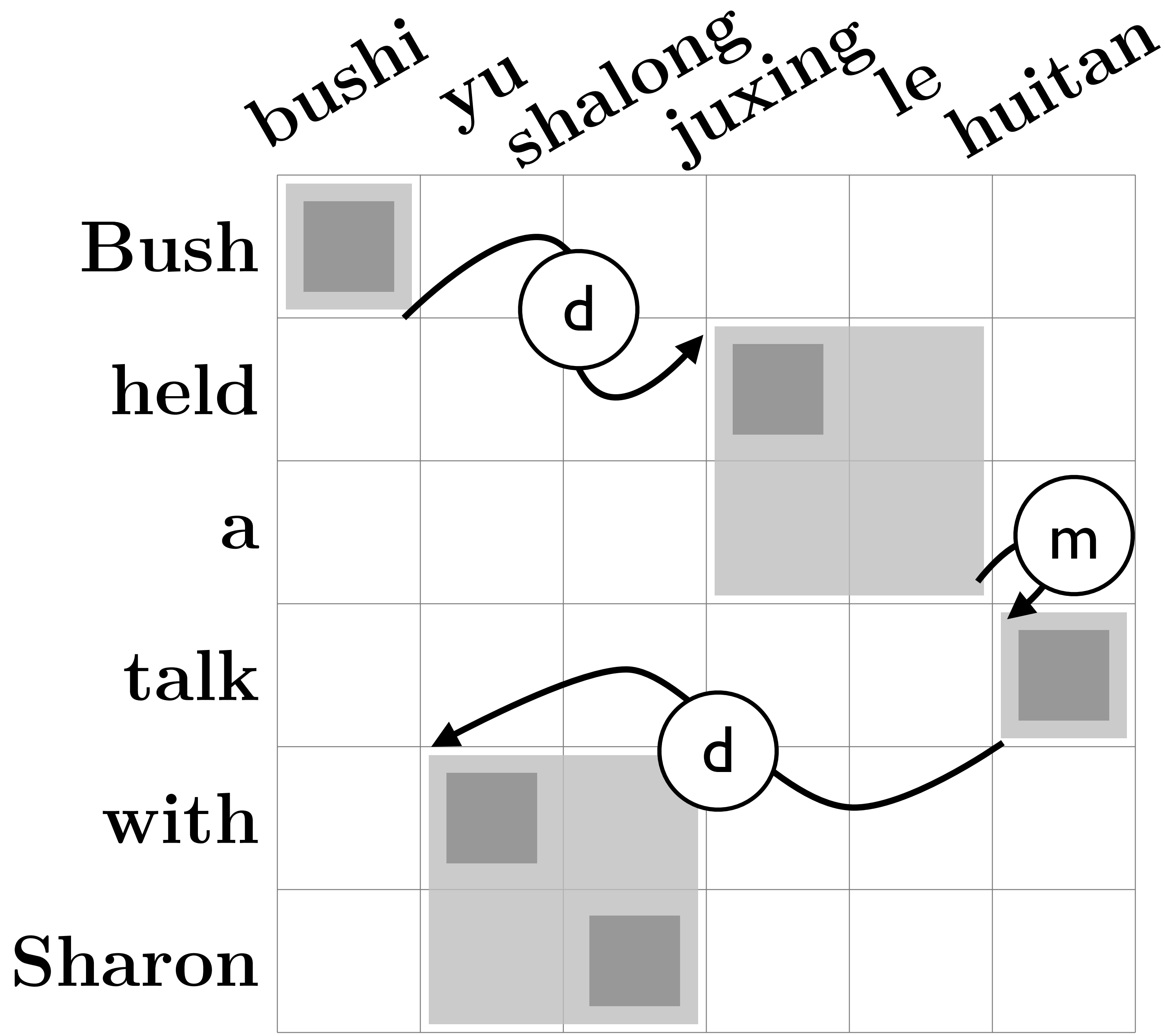
- Fine grained reordering features: $\log p_o(o \in \{m, s, d\} \,|\bar{\mathbf{f}}, \bar{\mathbf{e}})$
- Either monotone, swap, discontinuous

Features for Reordering



- Fine grained reordering features: $\log p_o(o \in \{m, s, d\} \,|\bar{\mathbf{f}}, \bar{\mathbf{e}})$
- Either monotone, swap, discontinuous

Features for Reordering



- Fine grained reordering features: $\log p_o(o \in \{m, s, d\} \,|\bar{\mathbf{f}}, \bar{\mathbf{e}})$
- Either monotone, swap, discontinuous

Other Features

- log of ngram language model(s)
- word count: bias for ngram language model(s)
- phrase count: shorter or longer phrases

Direct Training

- Instead of word alignment + extraction pipeline, directly learn phrase-pairs (Marcu and Wong, 2002)
- Bayesian approach + blocked Gibbs sampling to learn parameters (Blunsom et al., 2009)
    - Initialize derivations of D
    - For each pair f, e, sample new derivation
        - Update statistics
- Exhaustively memorize longer phrases (Neubig et al., 2011)

Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

Decoding

$$\begin{aligned}\hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} \frac{\exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\right)}{\sum_{\mathbf{e}', \phi'} \exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}', \phi', \mathbf{f})\right)} \\ &= \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\end{aligned}$$

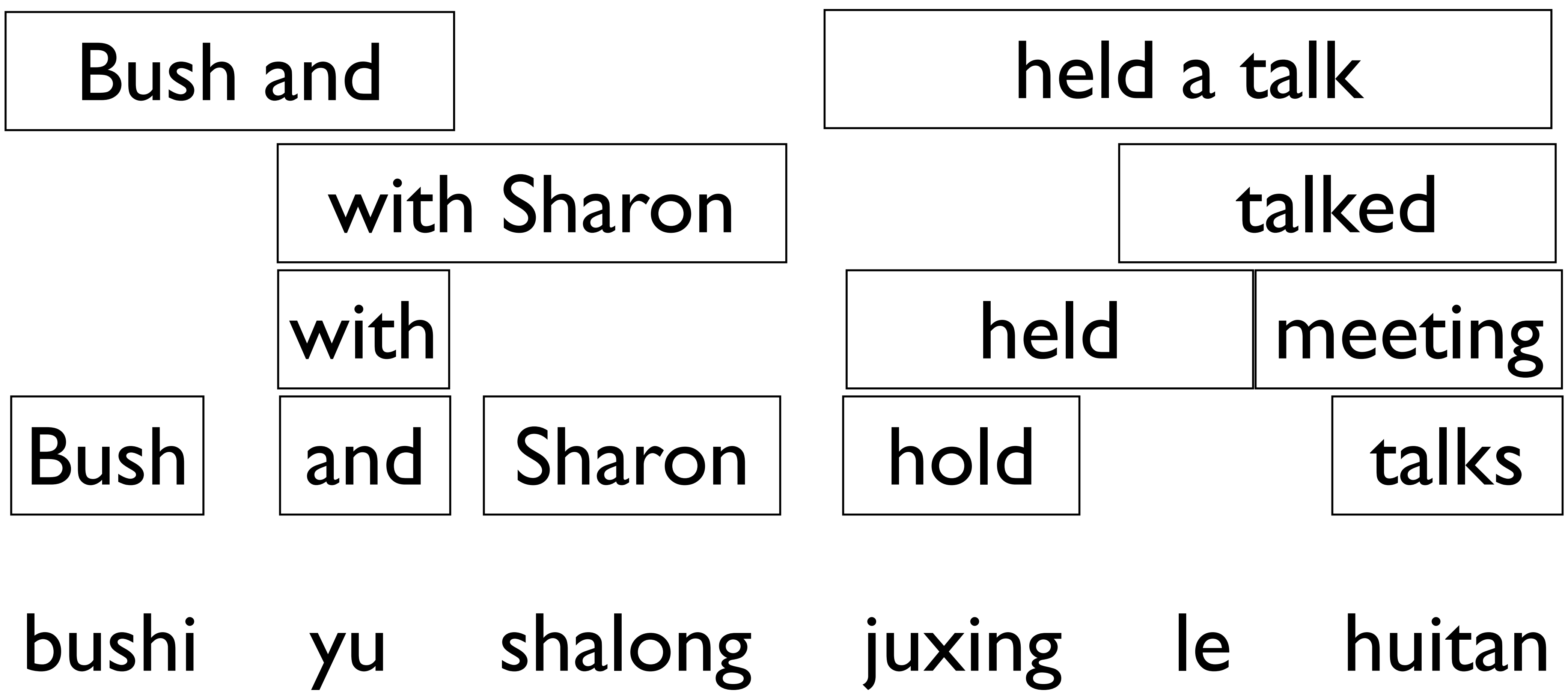
- Given an input sentence f and phrasal model h and w, seek e with the highest score
- Potential errors:
  - Search error: we cannot find the best scored hypothesis
  - Translation error: highest scored hypothesis is bad

Enumerate Phrase Pairs

bushi yu shalong juxing le huitan

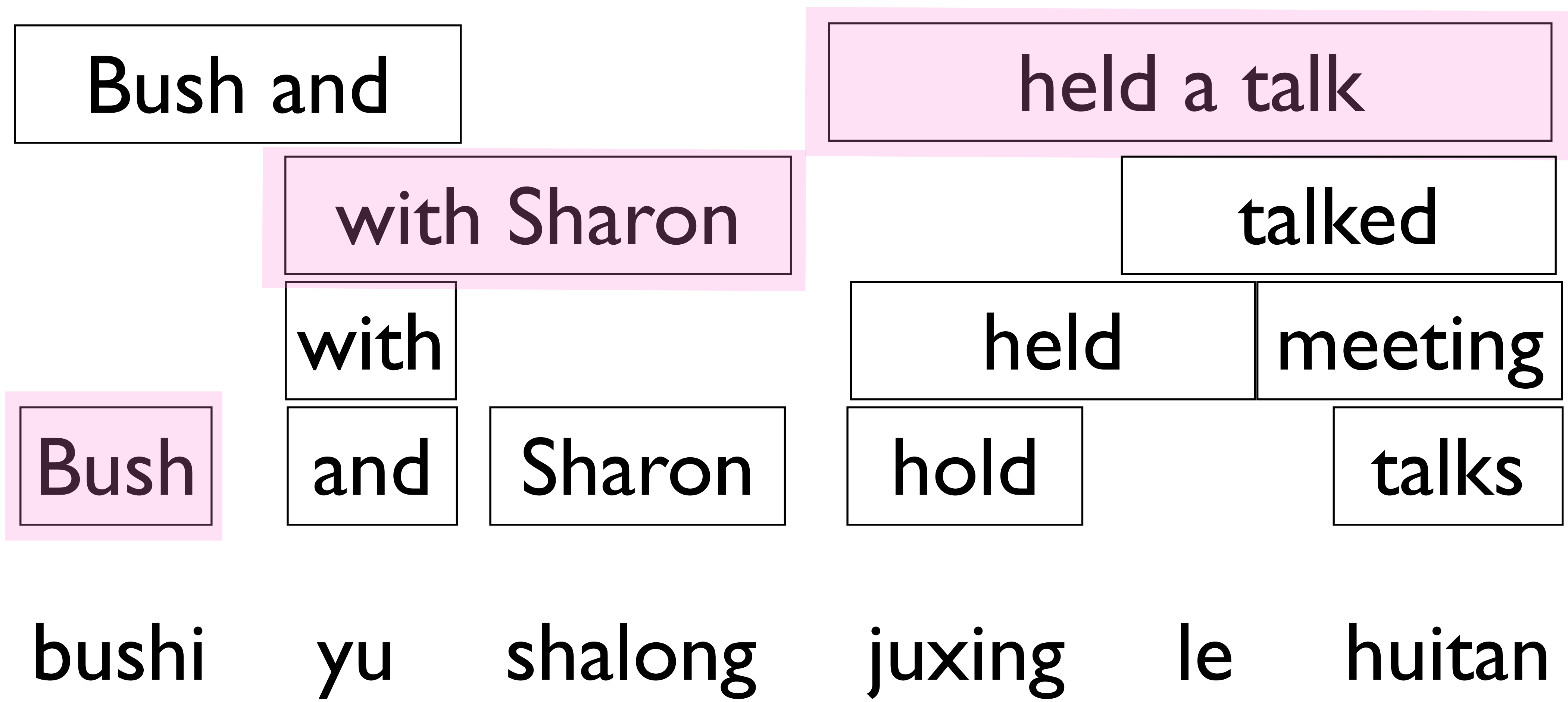
- Given a input sentence f, we can enumerate all possible phrases that match with the source side
- Choose the best phrase pair + ordering

Enumerate Phrase Pairs



- Given a input sentence f, we can enumerate all possible phrases that match with the source side
- Choose the best phrase pair + ordering

Enumerate Phrase Pairs



- Given a input sentence f, we can enumerate all possible phrases that match with the source side
- Choose the best phrase pair + ordering

Phrase-based Search Space

bushi yu shalong juxing le huitan

- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

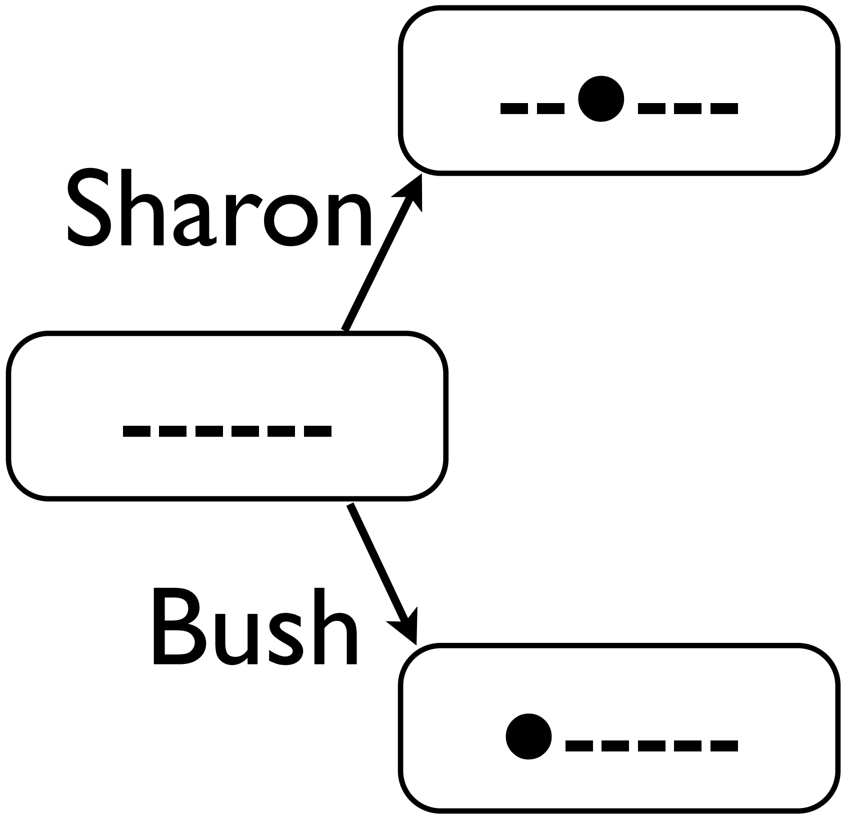
Phrase-based Search Space

------

bushi yu shalong juxing le huitan

- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

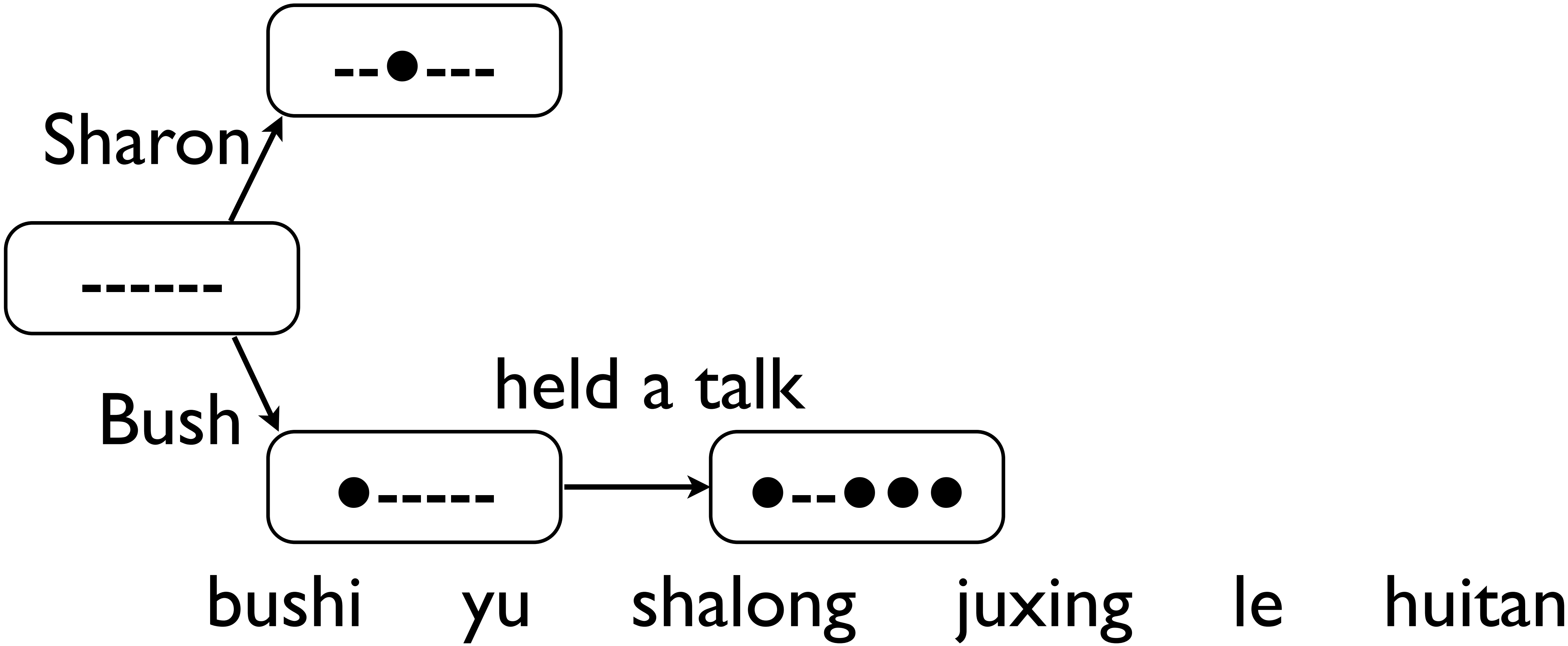
Phrase-based Search Space



bushi yu shalong juxing le huitan

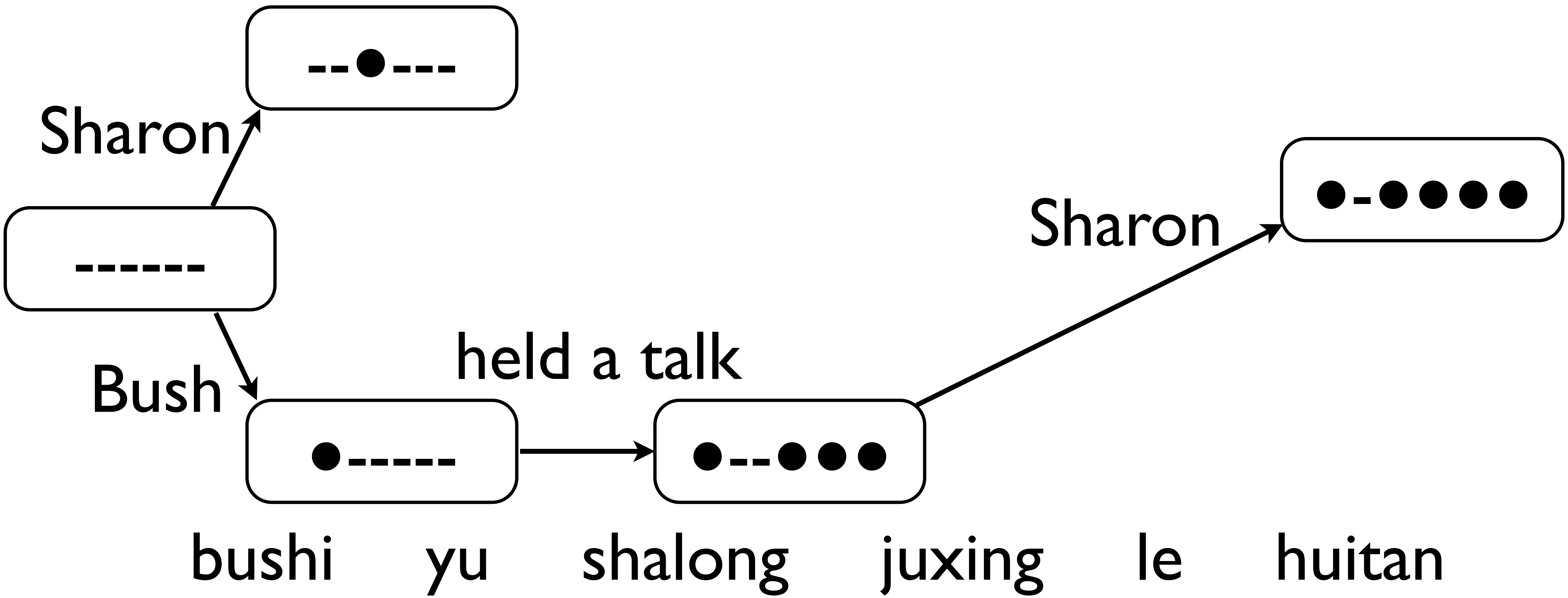
- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



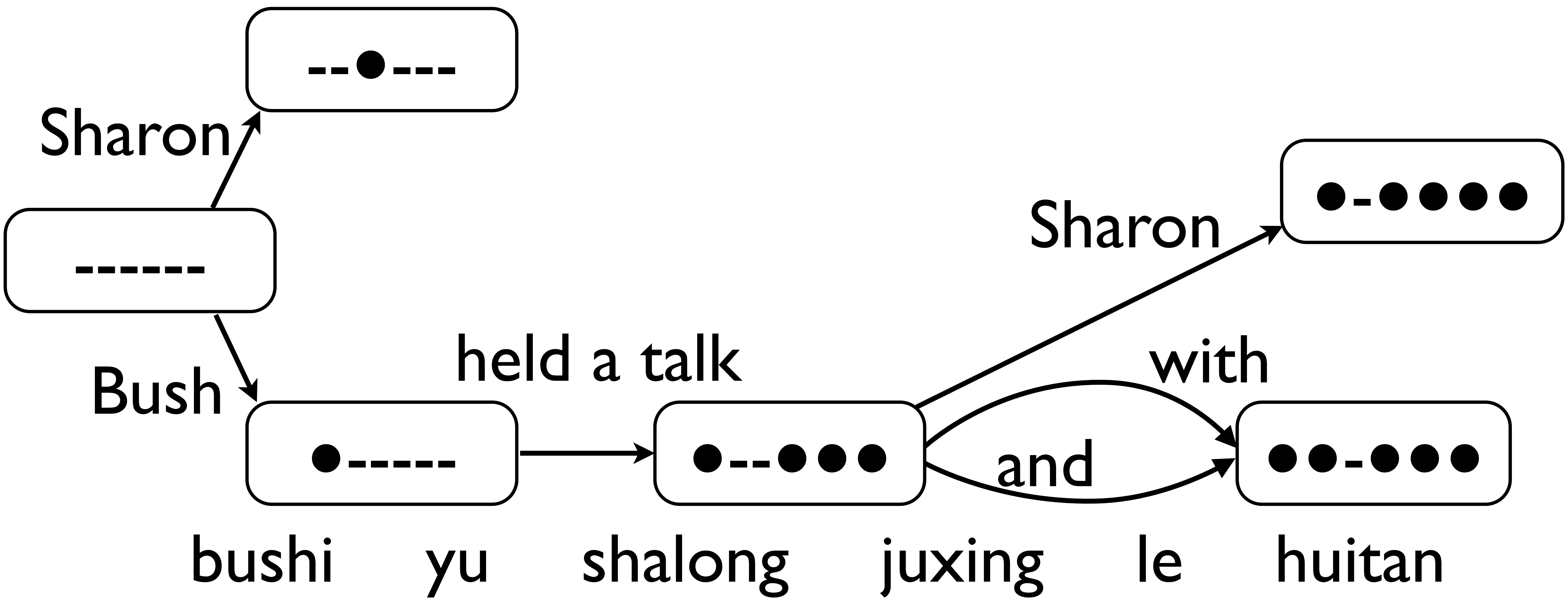
- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



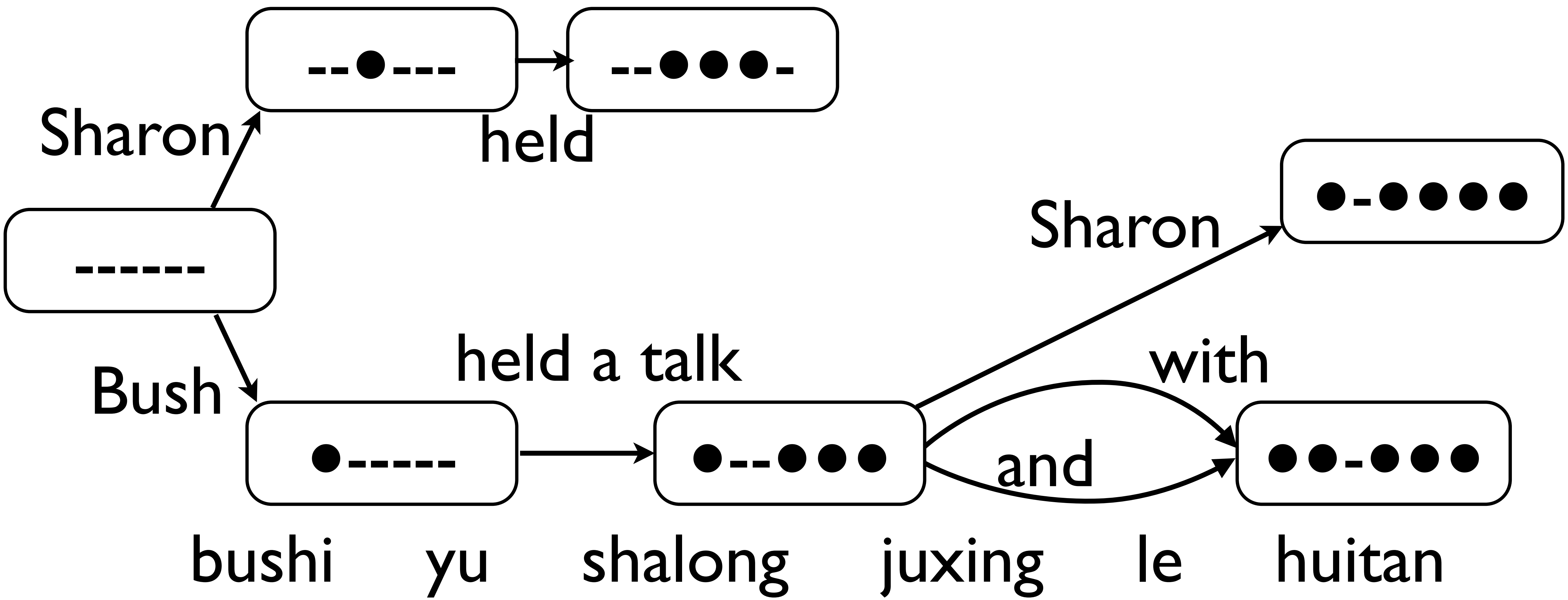
- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



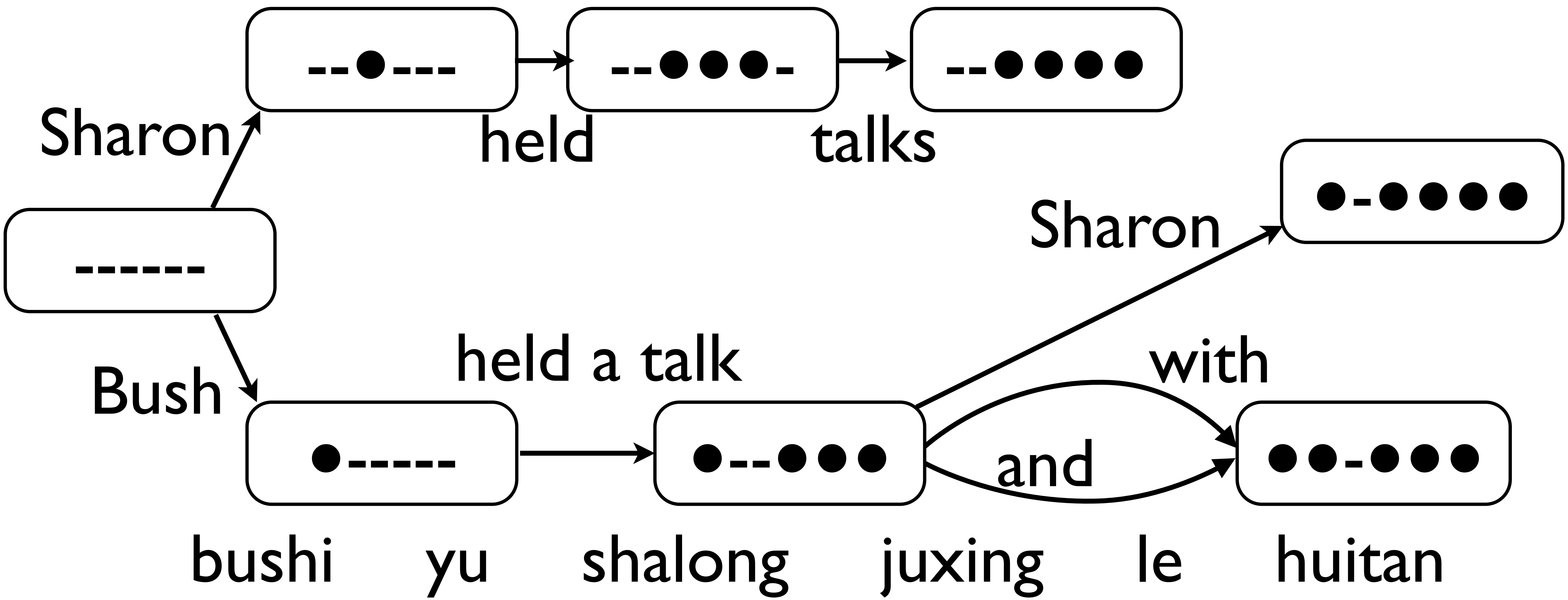
- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



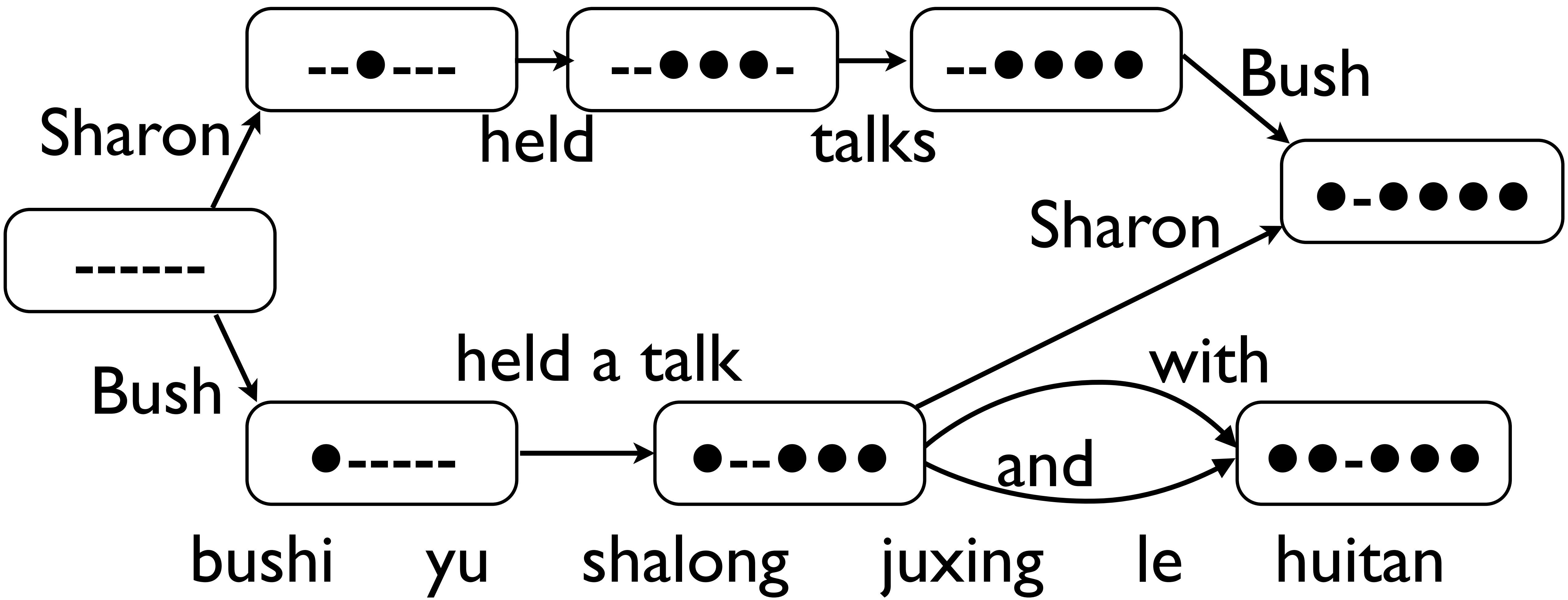
- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Phrase-based Search Space



- Node: bit-vector representing covered source words
- Edge: phrasal translations, strictly left-to-right
- Search space: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)

Traveling Salesman Problem

- NP-hard problem: visit each city only once
- MT as a Traveling Salesman Problem (Knight, 1999)
  - Each source word corresponds to a city
  - A Dynamic Programming solution:
    - State: visited cities (bit-vector)
    - Search space: $O(n^2)$
  - Distortion limit to reduce search space

    i.e. long distortion: ●----- → ●----●

Non-local features

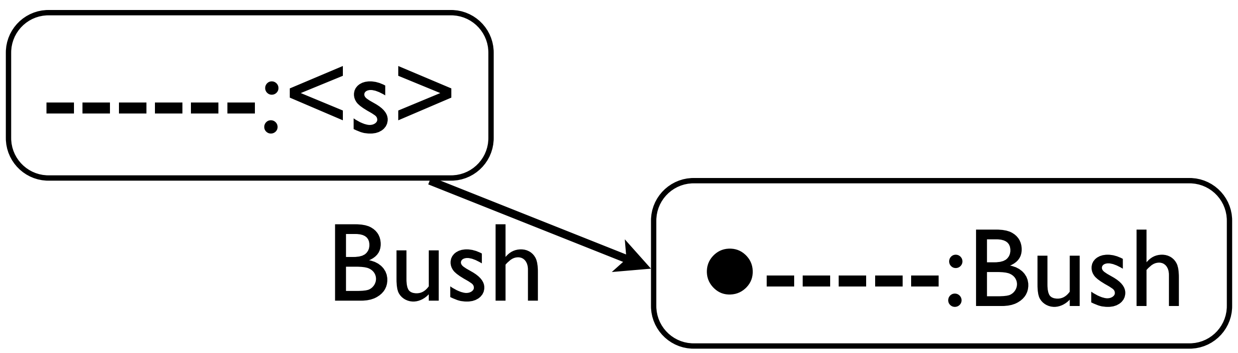
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features

------:<s>

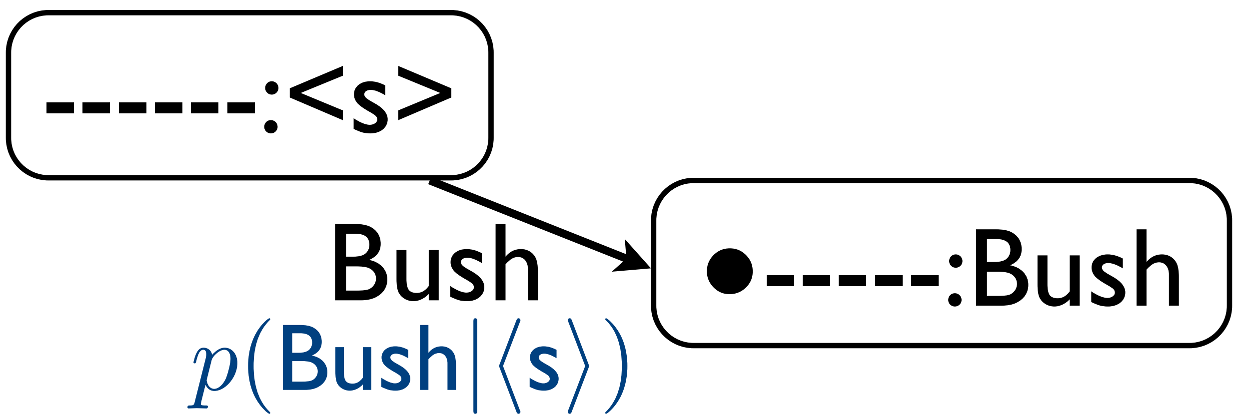
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



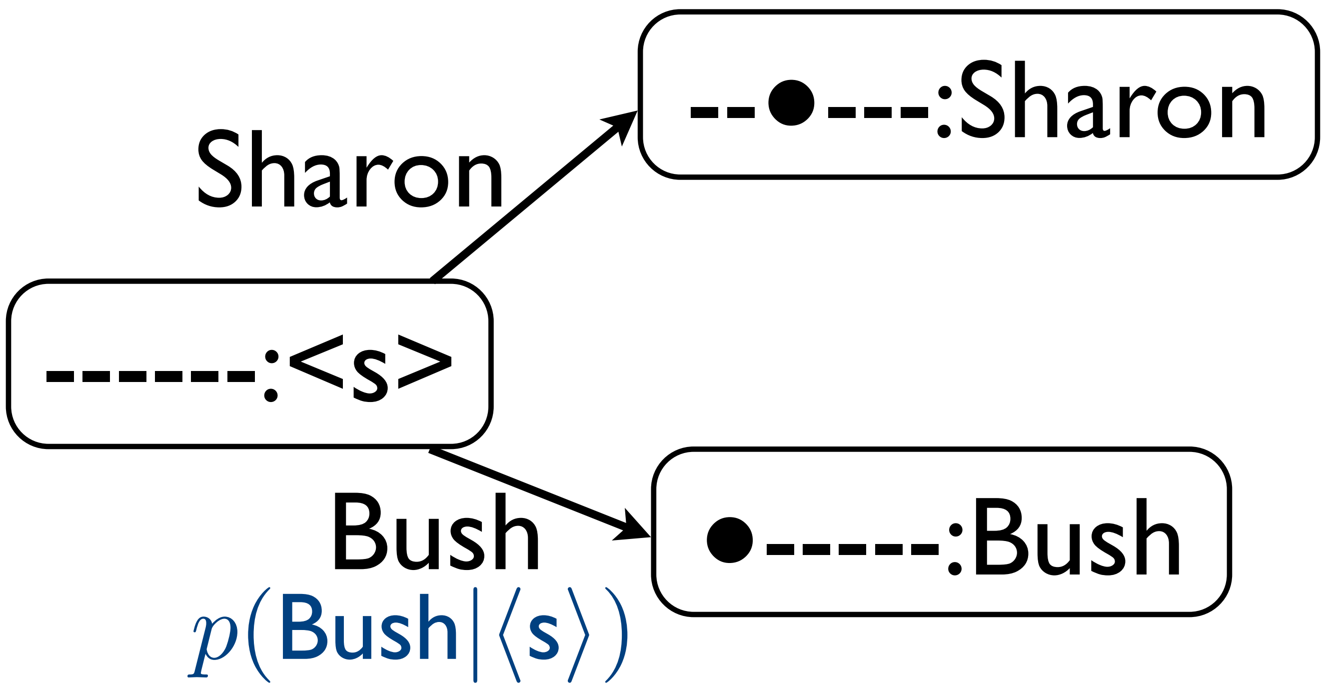
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for “future scoring”: 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



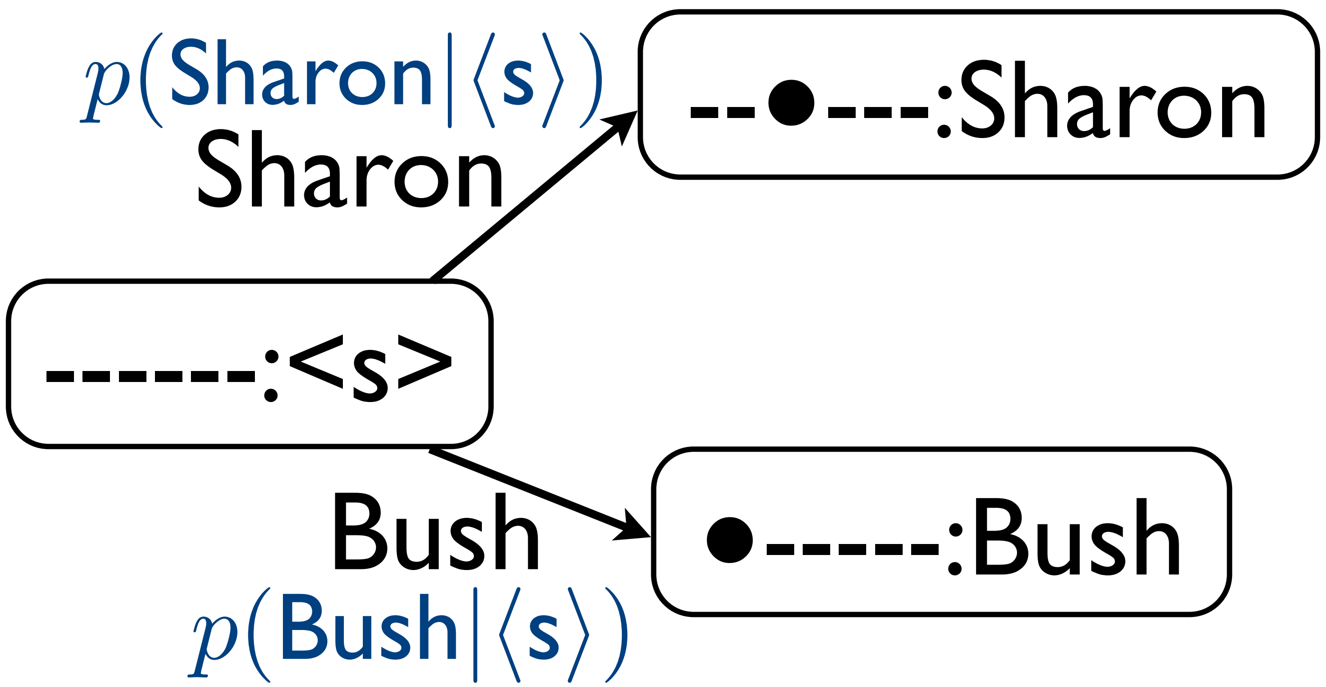
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for “future scoring”: 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



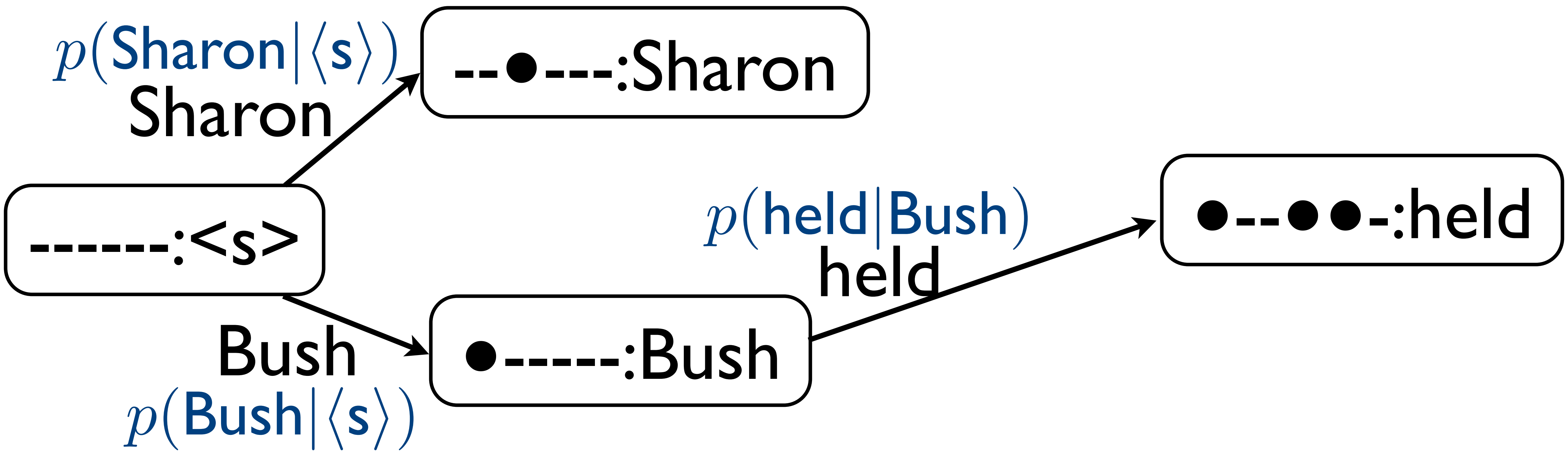
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for “future scoring”: 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



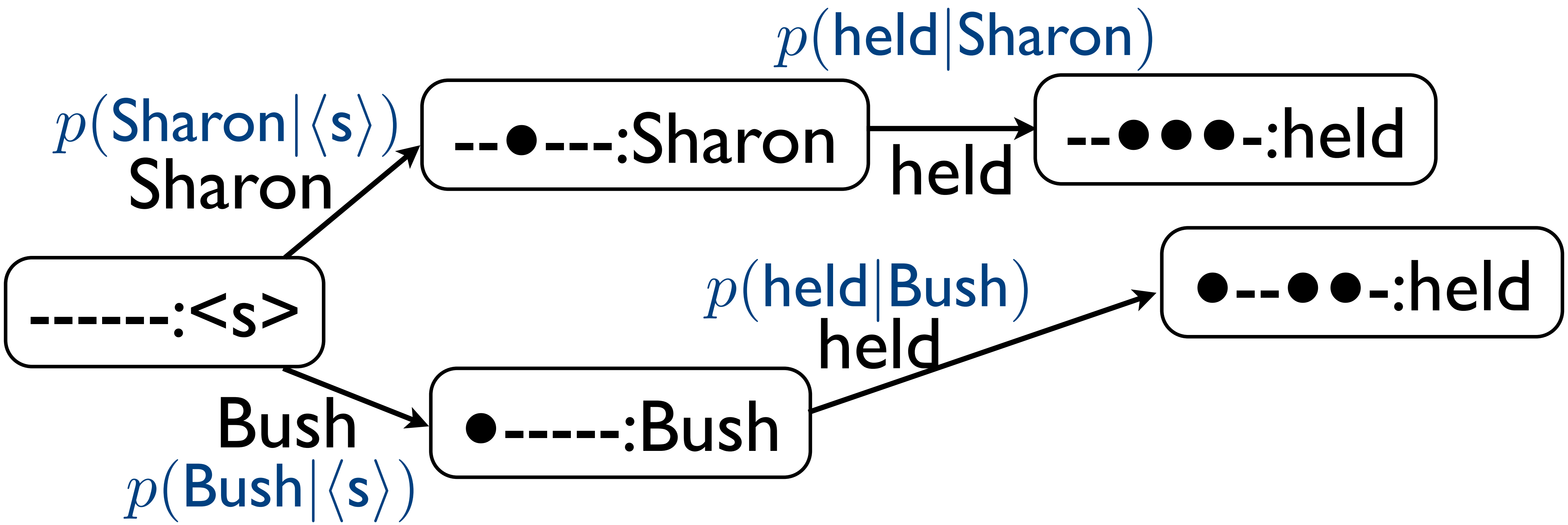
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



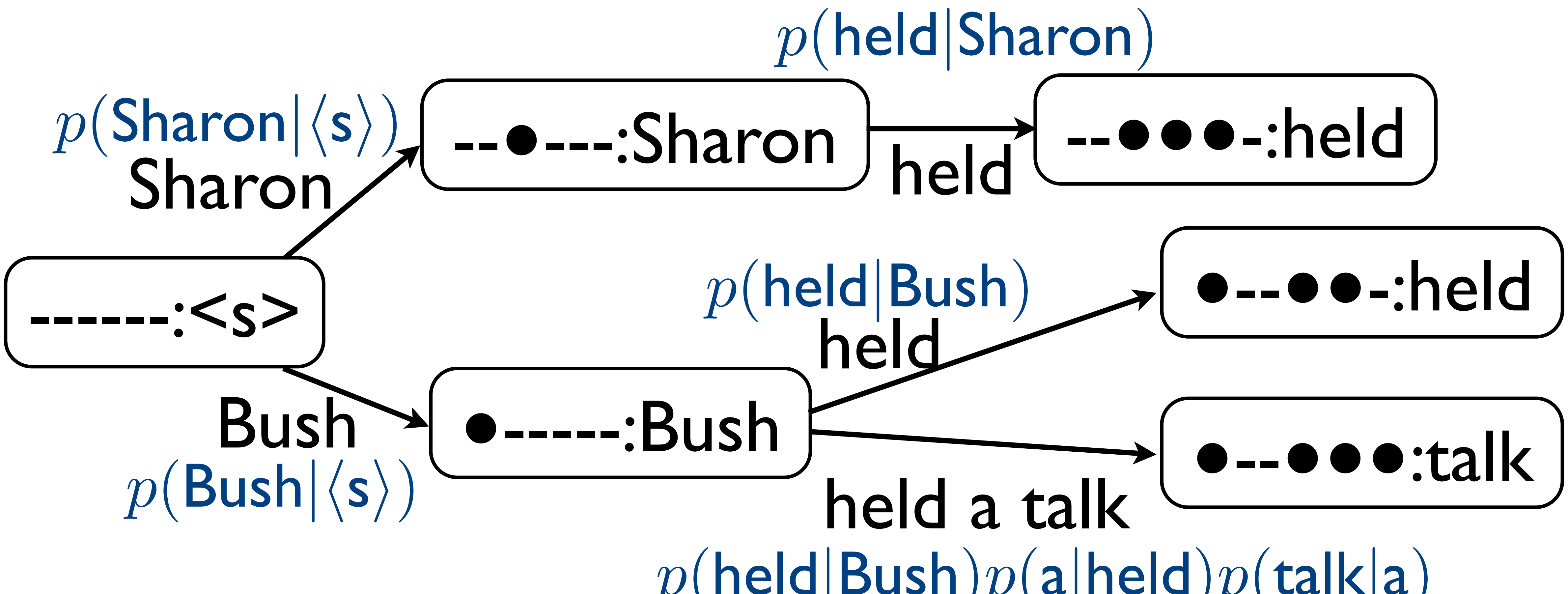
- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

Non-local features



- Features that requires scoring out of phrases: bigram language model
- Additional state representation required for "future scoring": 1-word for bigram LM
- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM

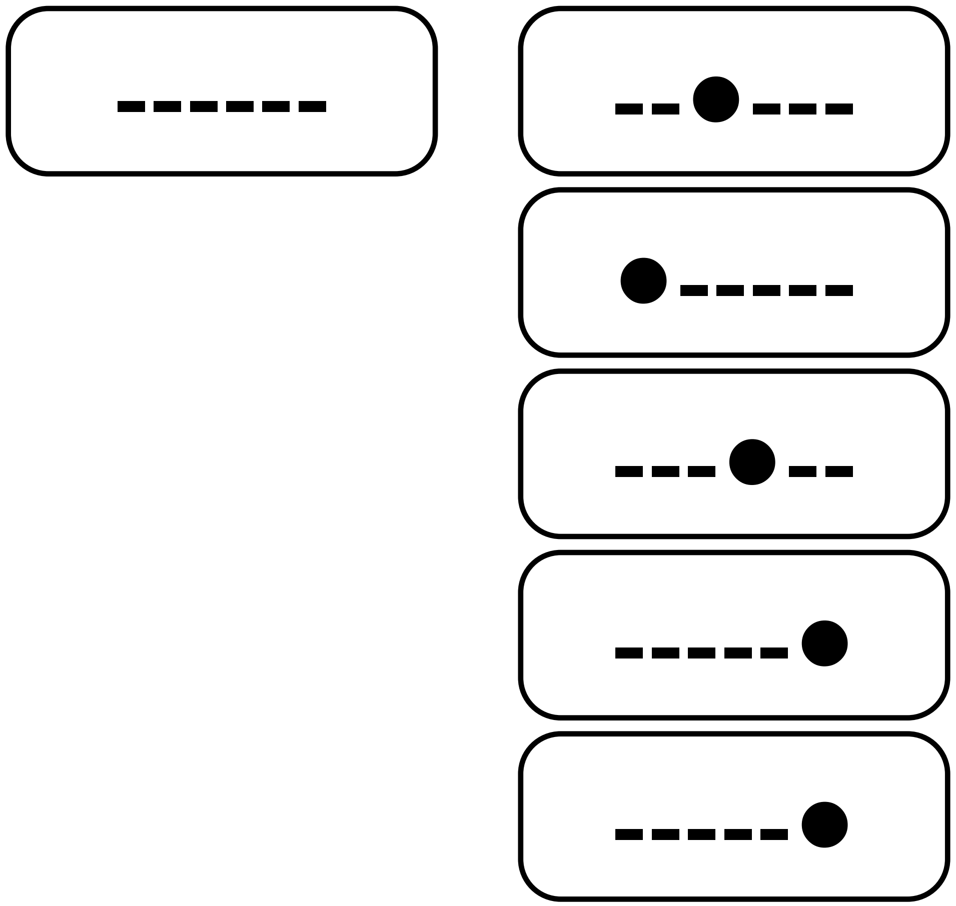
Phrase-based Decoding

- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding

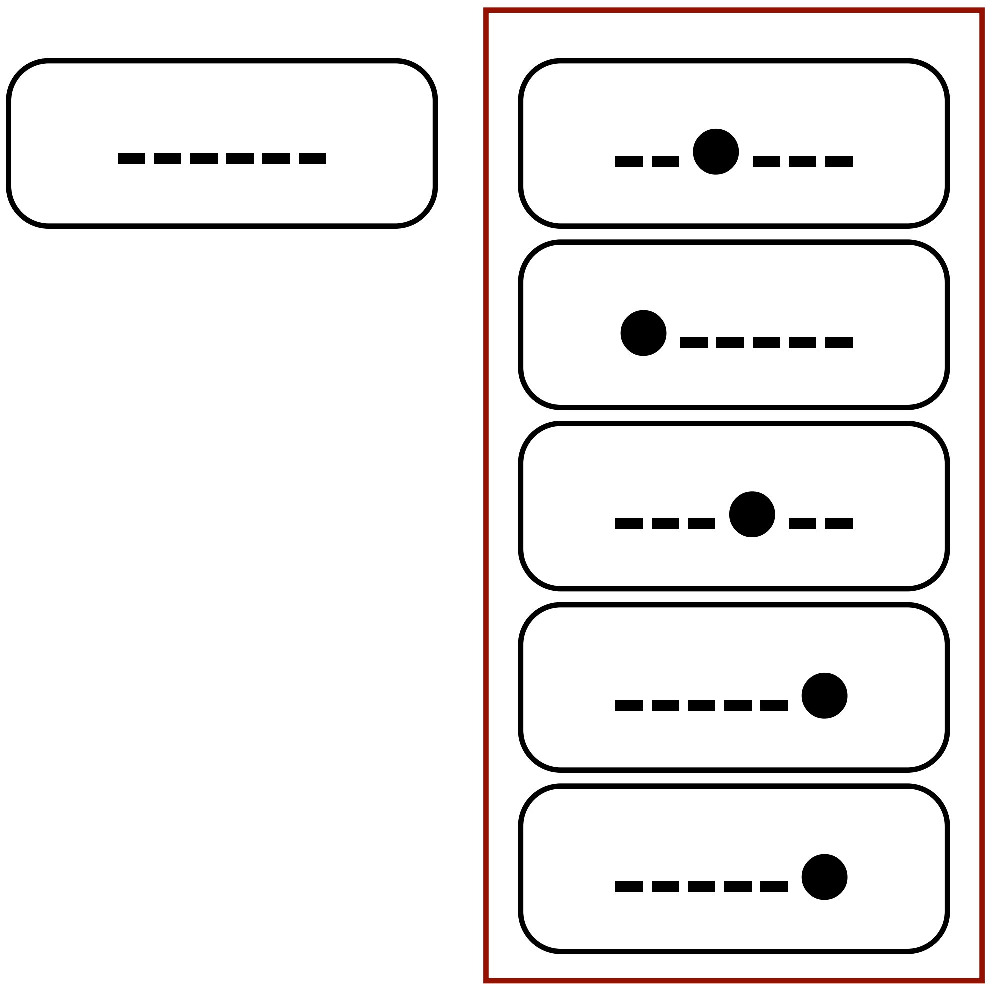
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



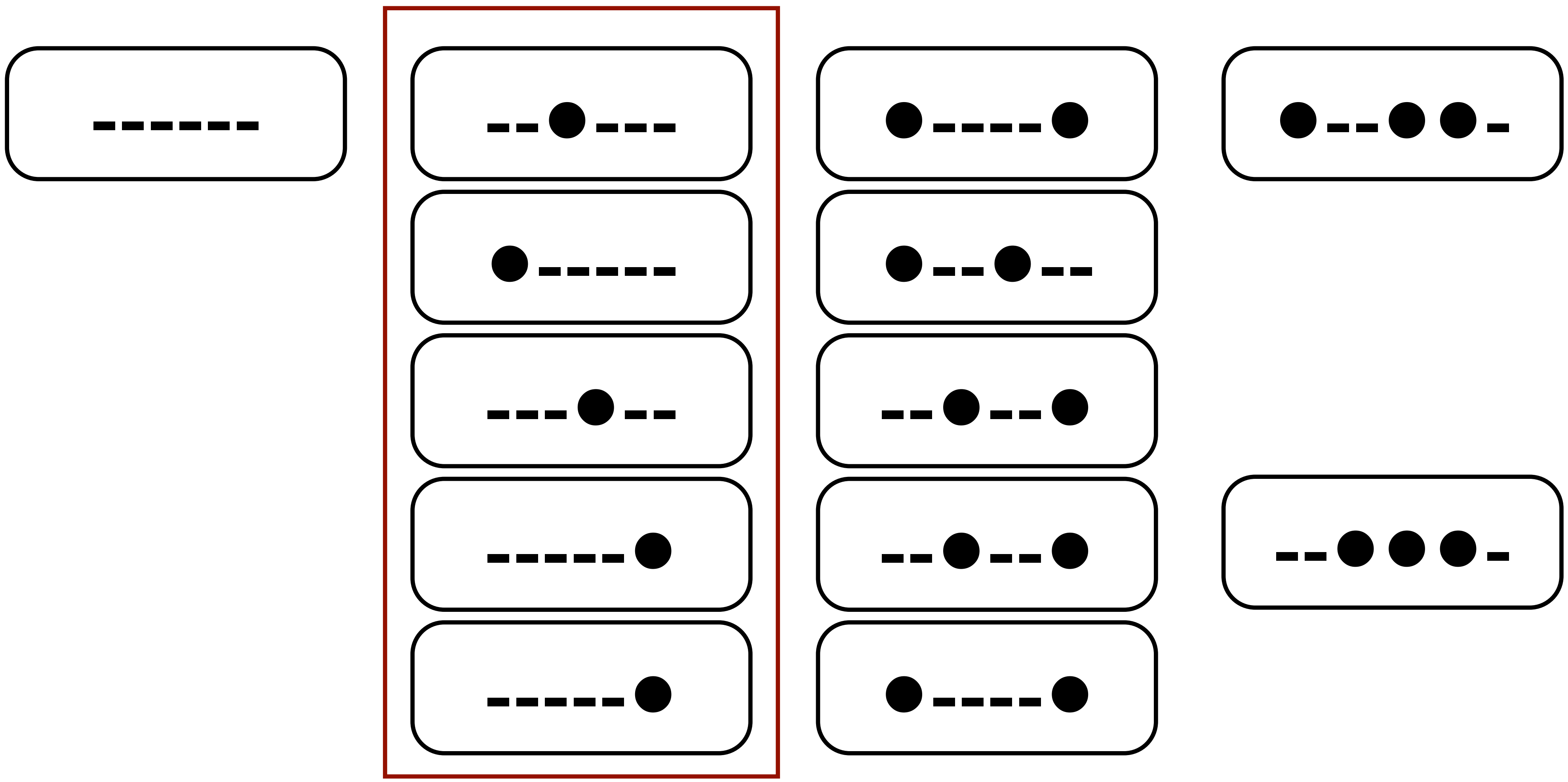
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



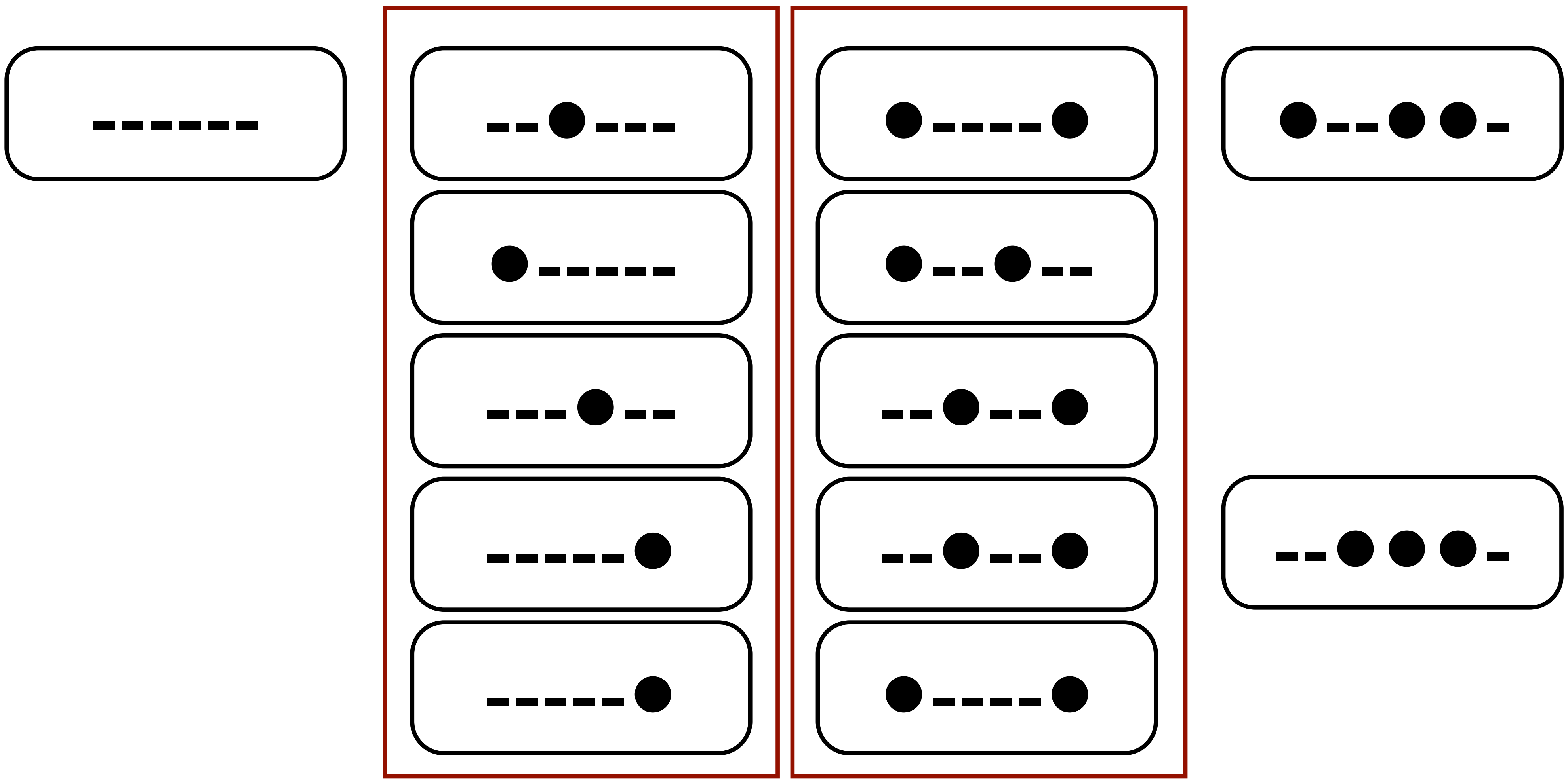
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



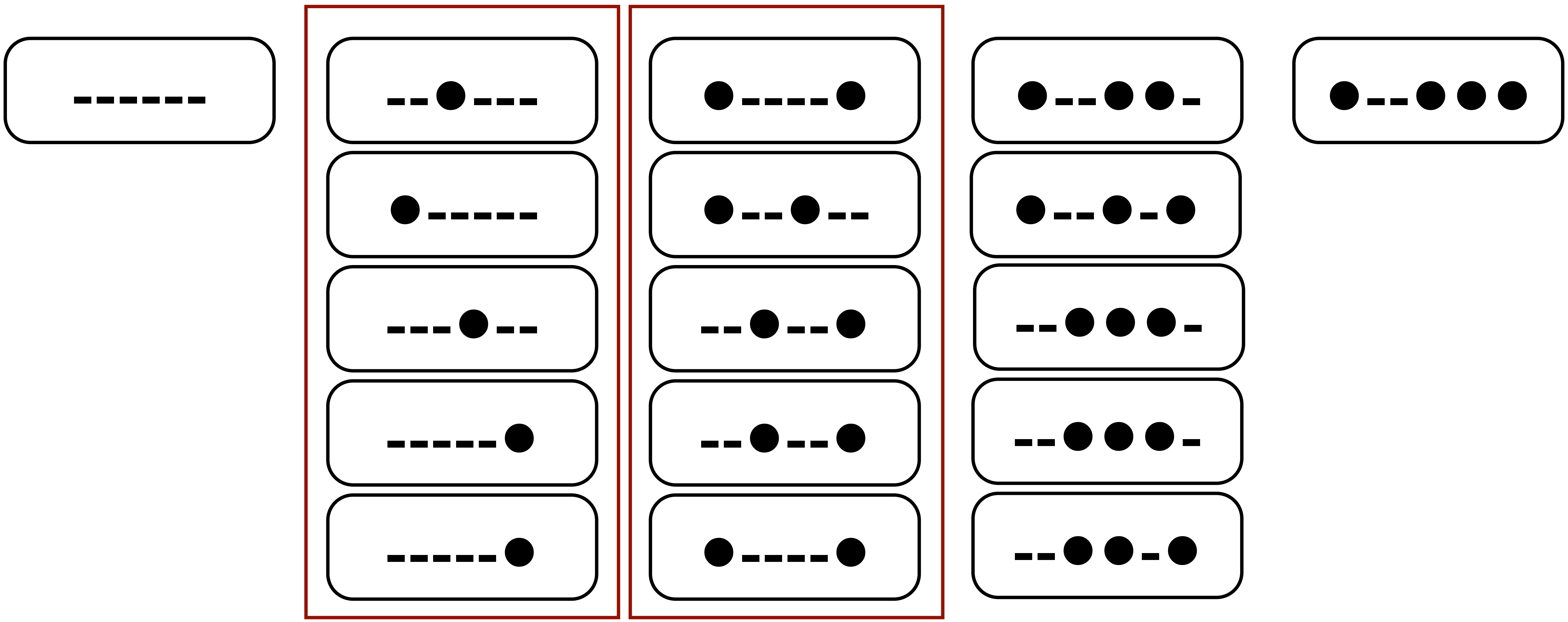
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



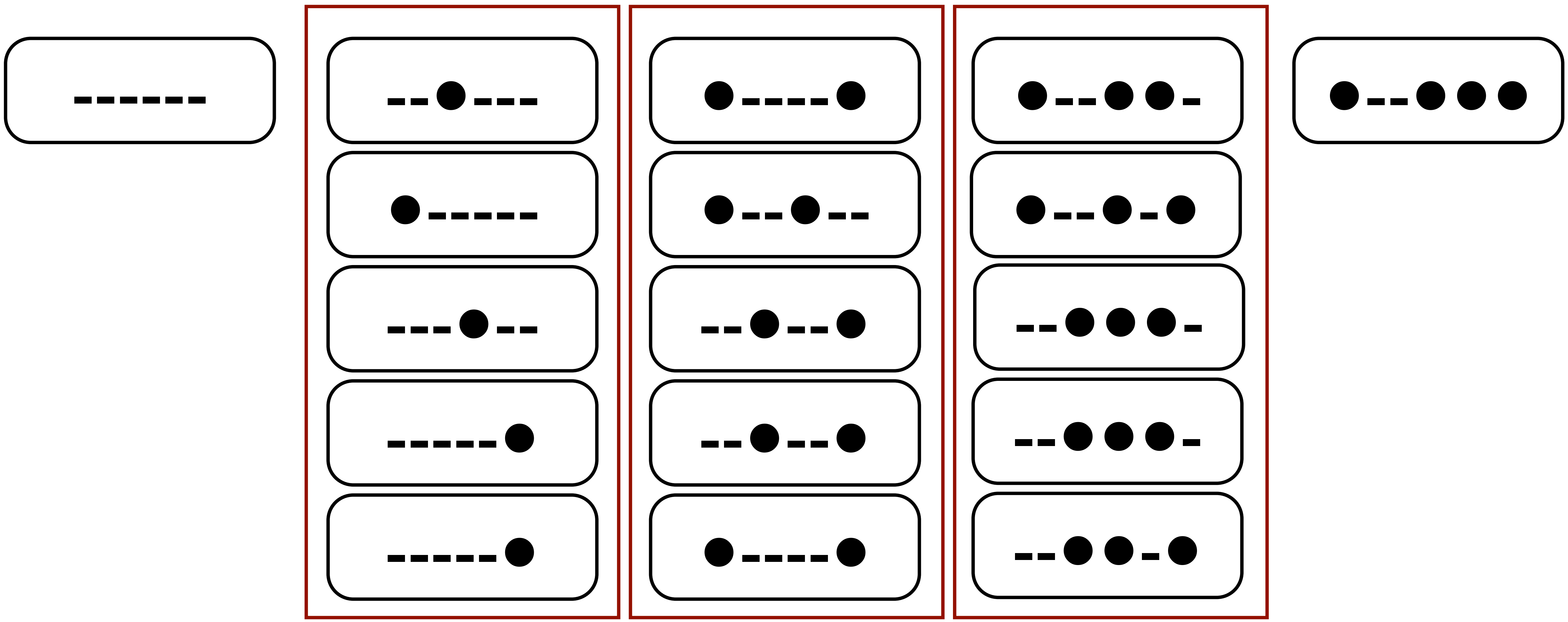
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



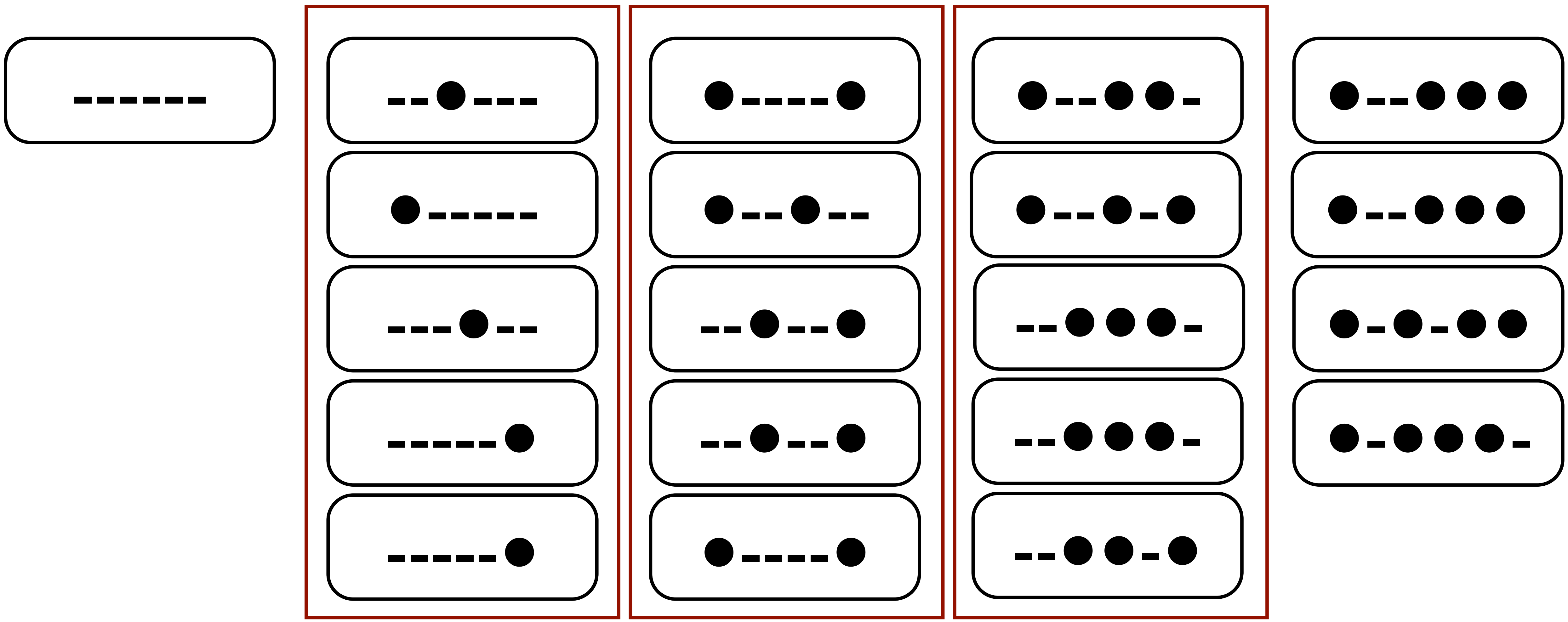
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



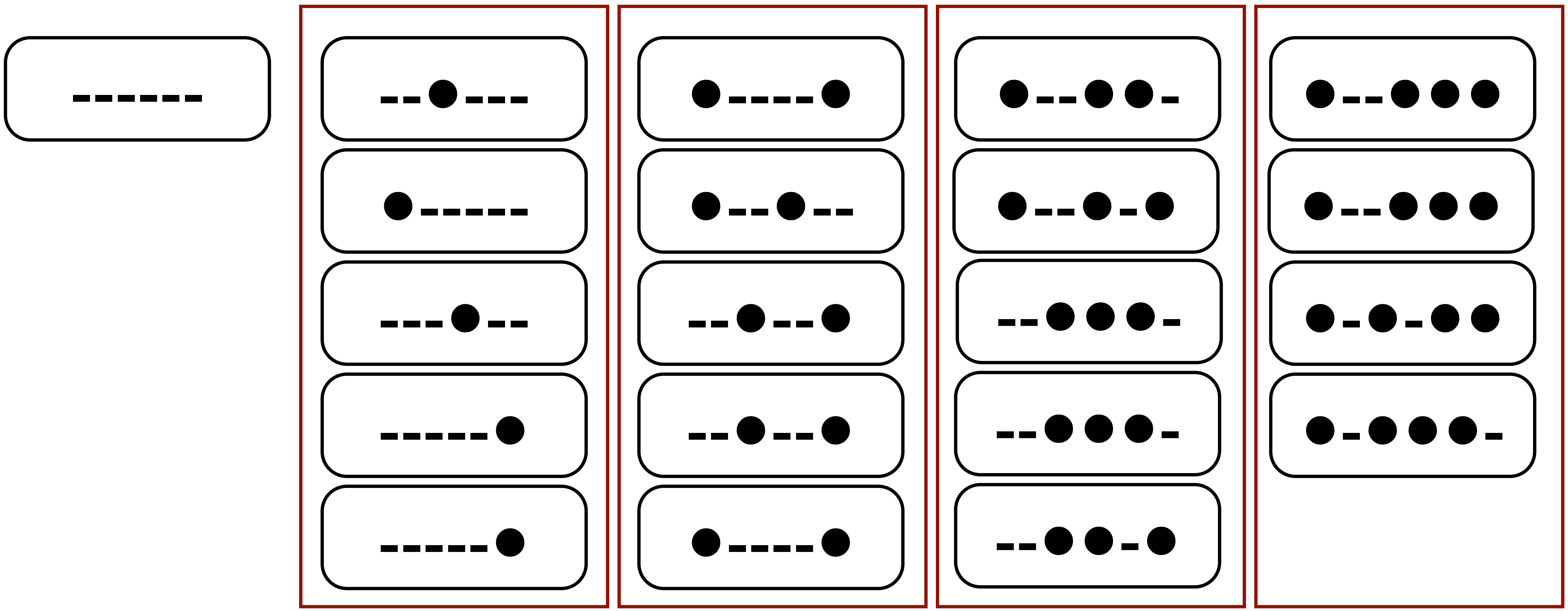
- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Phrase-based Decoding



- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first

Pruning

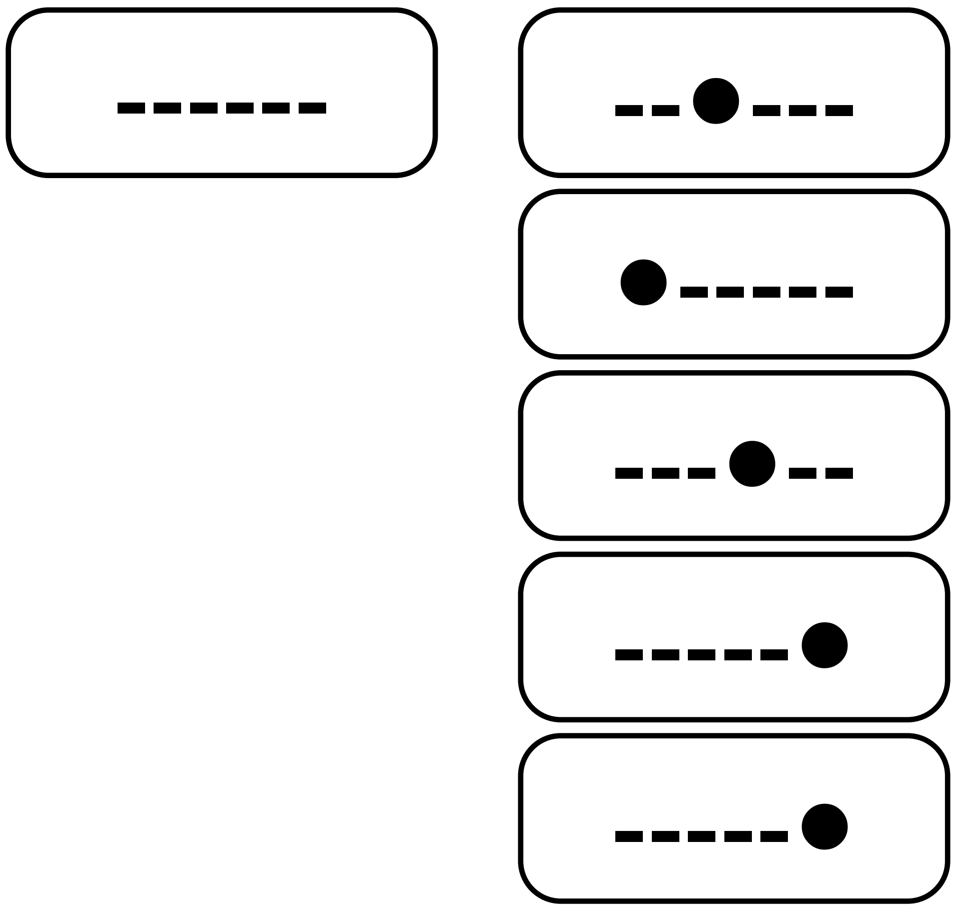
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning

`------`

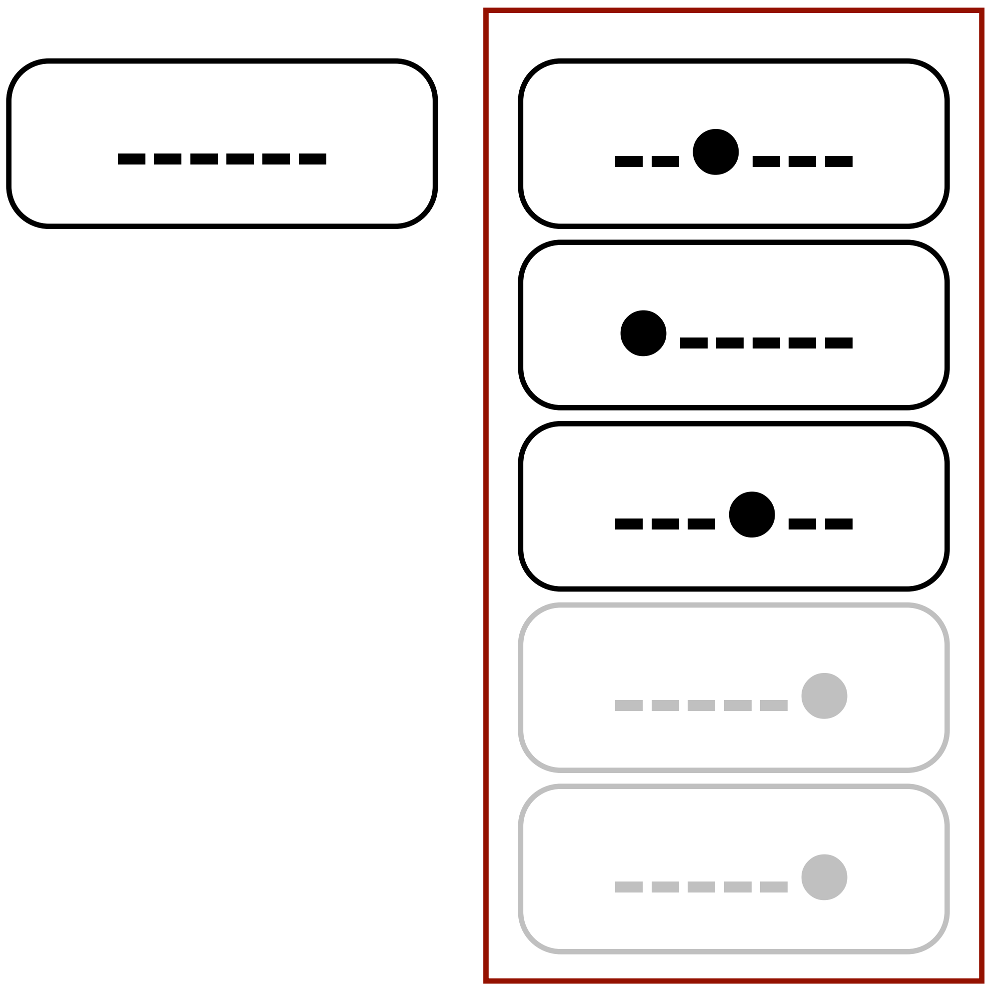
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



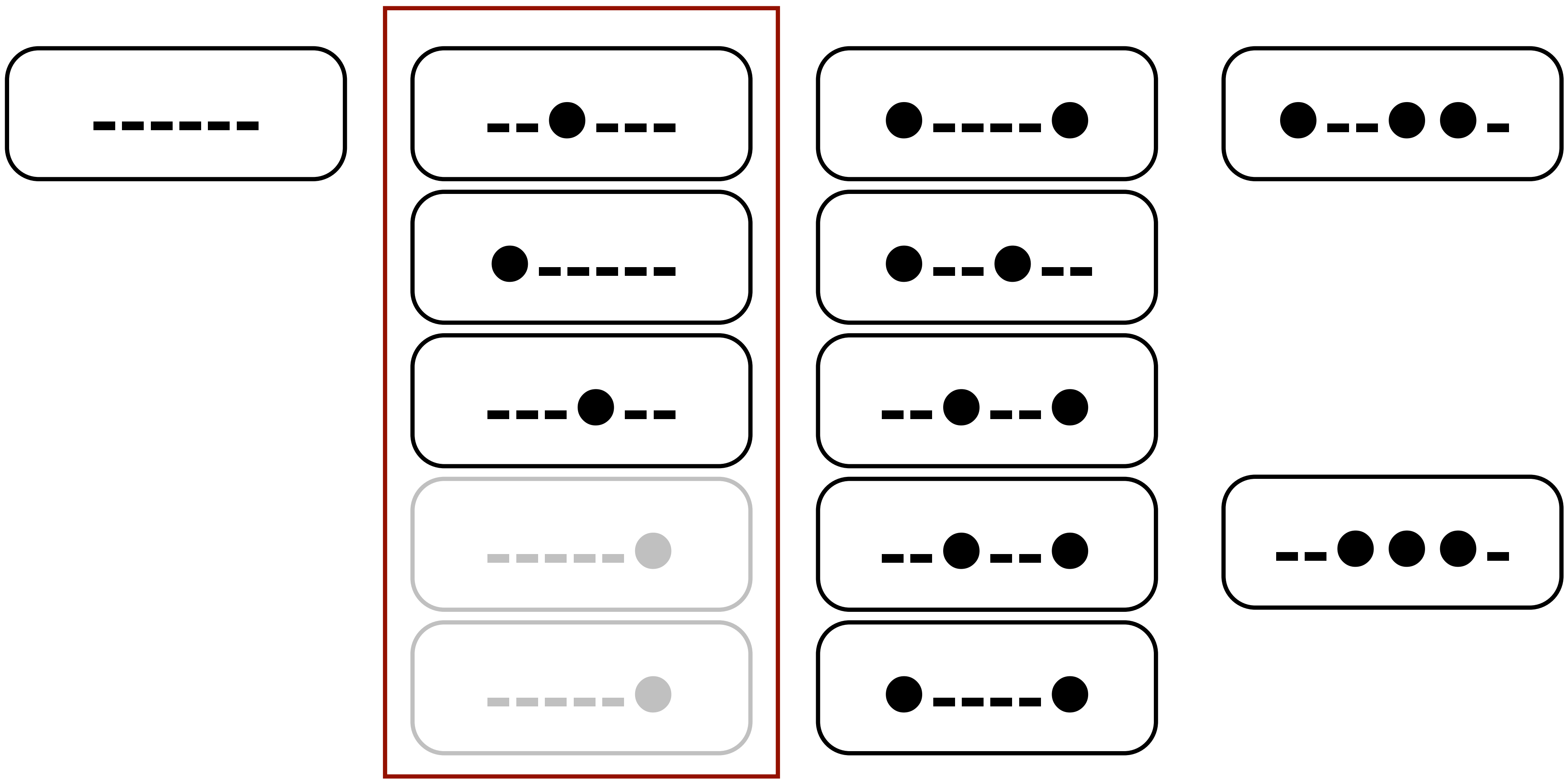
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



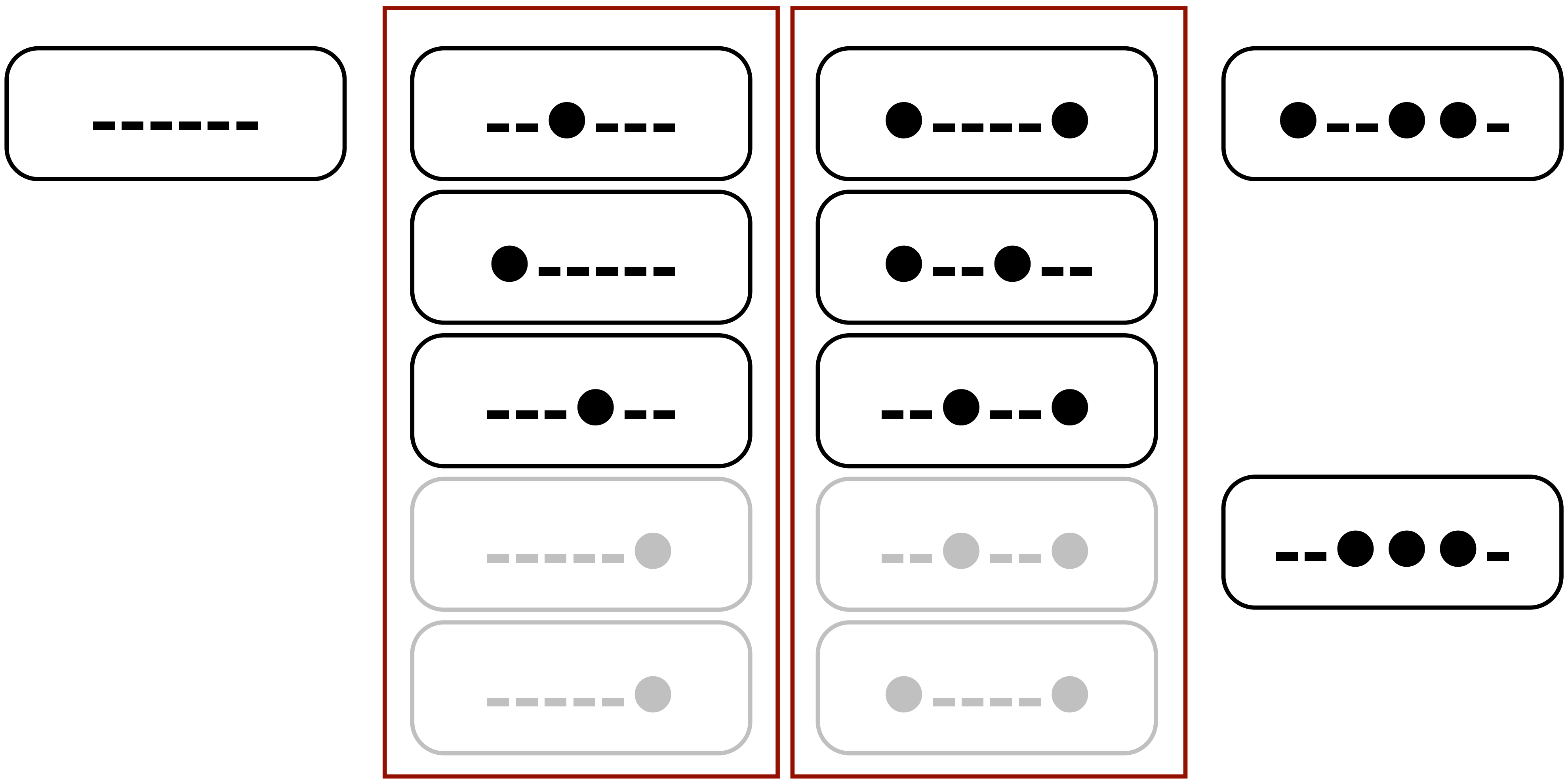
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



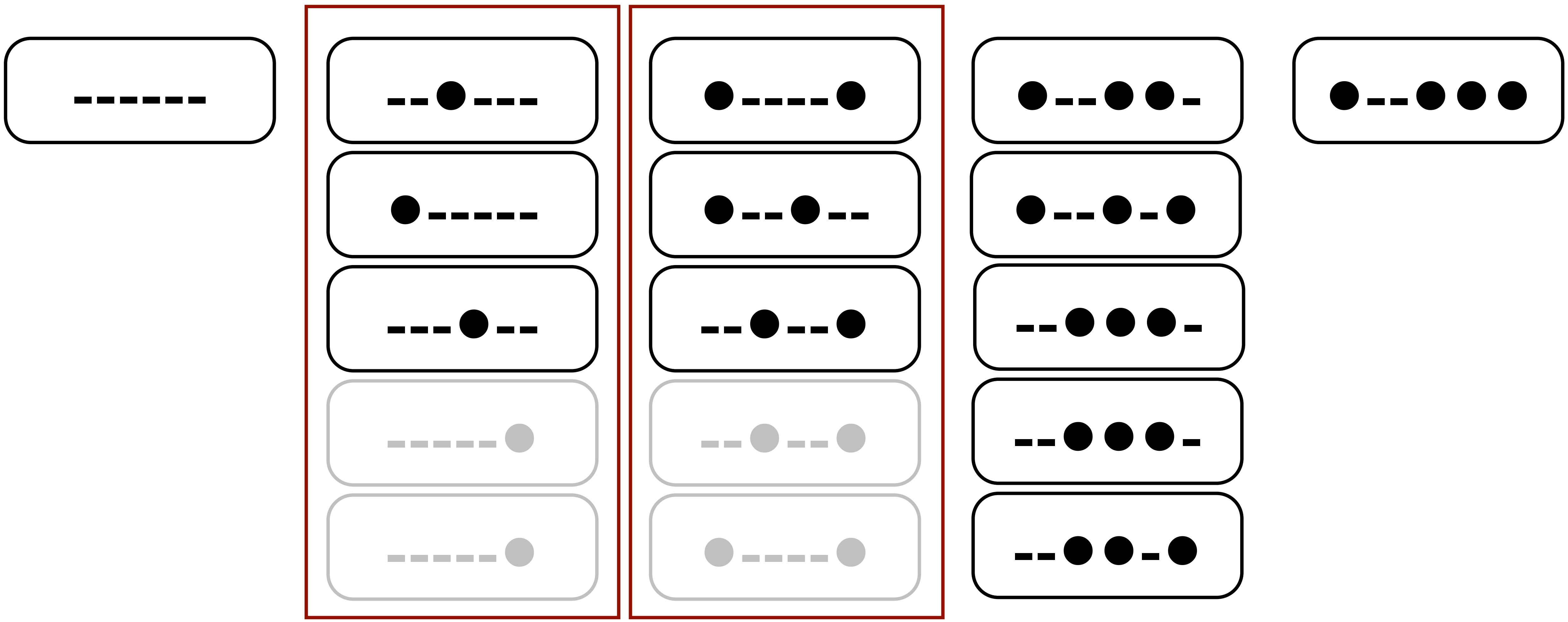
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



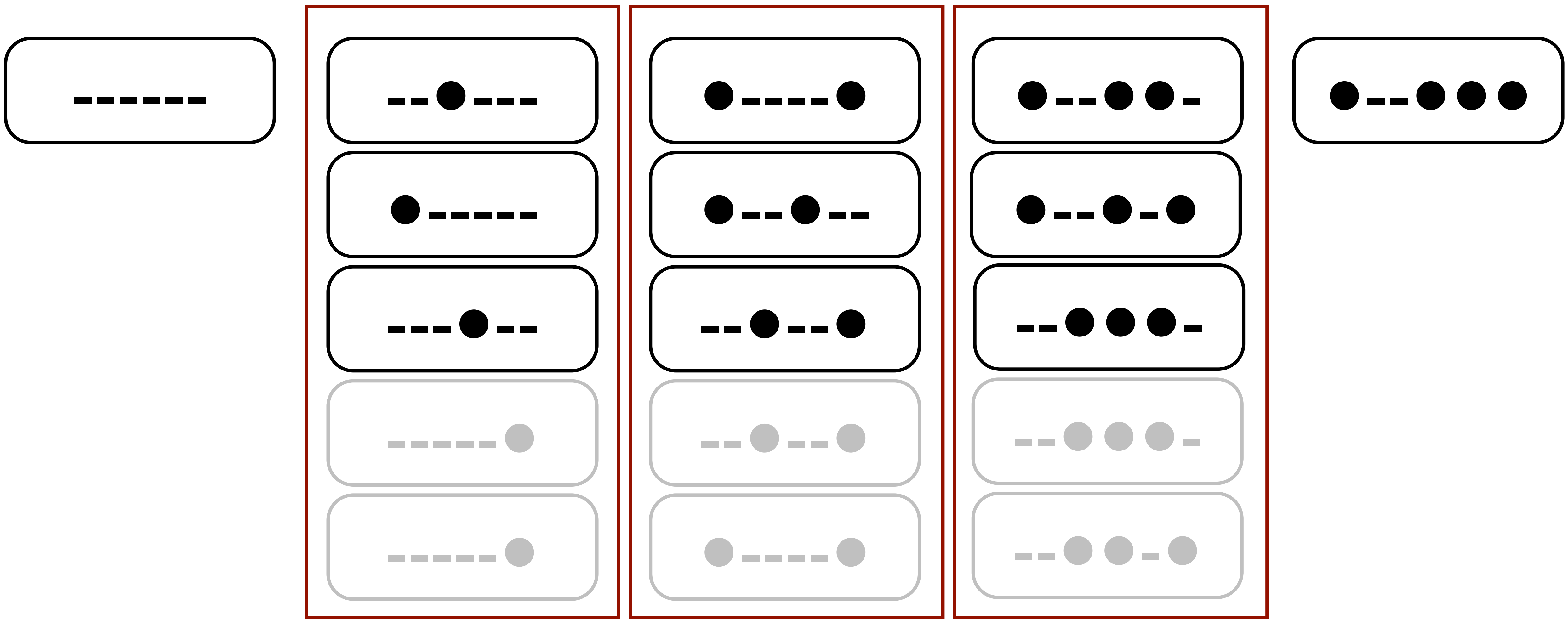
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



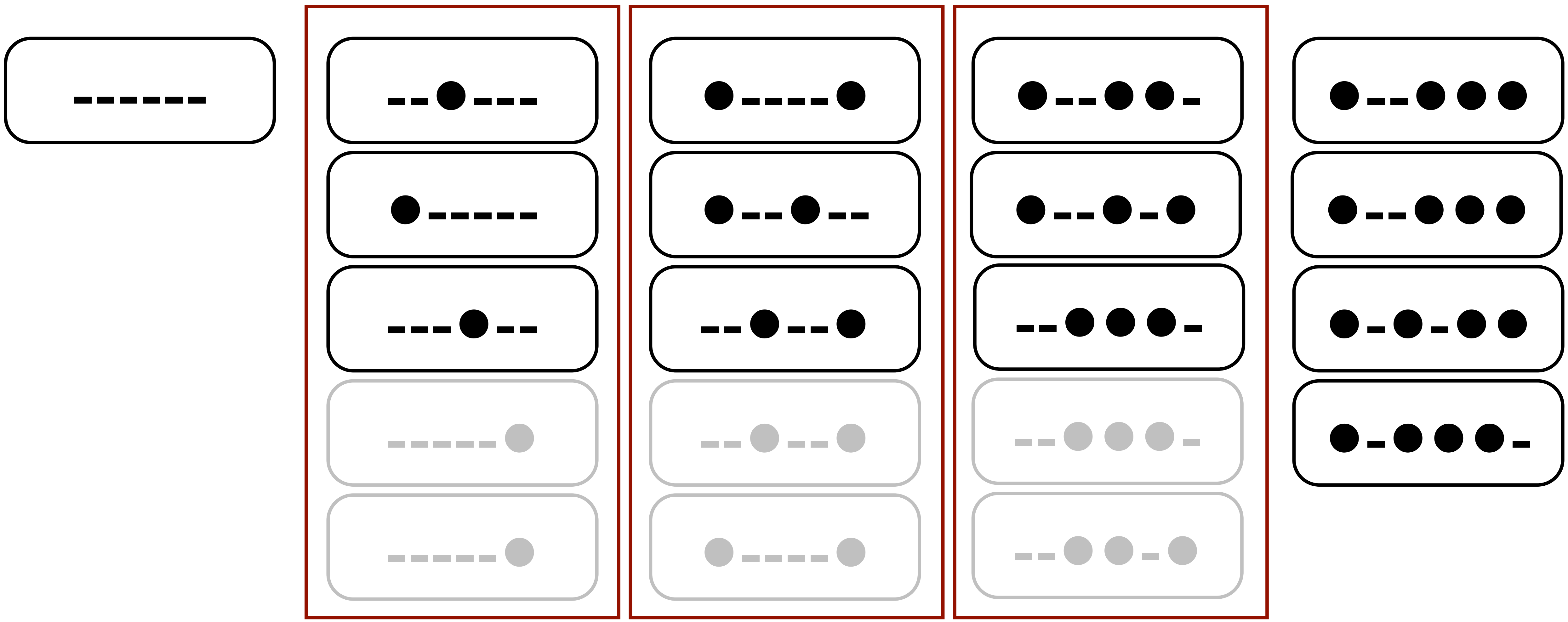
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



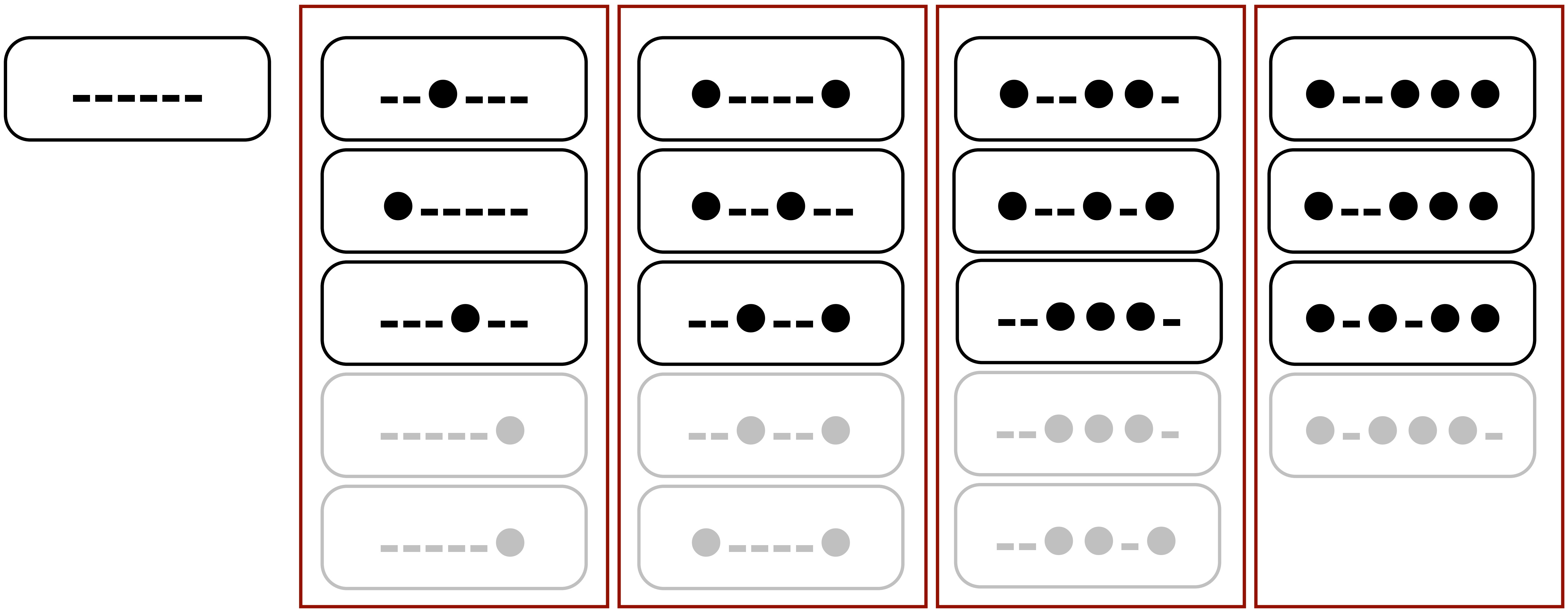
- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Pruning



- Prune hypotheses in a bin sharing the same cardinality
- Expand survived hypotheses only

Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

Questions

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

Tuning

$$\begin{aligned}\hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} \frac{\exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\right)}{\sum_{\mathbf{e}', \phi'} \exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}', \phi', \mathbf{f})\right)} \\ &= \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \phi, \mathbf{f})\end{aligned}$$

- Three popular objectives (in SMT) for tuning w
  - (Direct) Error Minimization (Och, 2003)
  - Maximum Entropy (Och and Ney, 2002)
  - Large Margin (Watanabe et al., 2007; Chiang et al., 2008; Hopkins and May, 2011)

(Direct) Minimum Error

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \sum_{s=1}^{S} l(\operatorname*{argmax}_{\mathbf{e}} \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \mathbf{f}_s), \mathbf{e}_s)$$

- MERT (Minimum ERror Training)
- Standard in SMT (but not in other NLP areas, such as tagging etc.)
  - We can incorporate arbitrary error functions, l
  - "Summation" can be replaced by document-wise BLEU specific summation
  - 10+ real valued features

n-best Approximation

procedure MERT($\{(\mathbf{e}_s, \mathbf{f}_s)\}_{s=1}^{S}$)
    for $n = 1...N$ **do**
        Decode and generate nbest list using $\mathbf{w}$
        Merge nbest list
        for $k = 1...K$ **do**
            for each parameter $m = 1...M$ **do**
                Solve one dimensional optimization
            end for
            update $\mathbf{w}$
        end for
    end for
end procedure

- N iterations, with each iteration, n-bests are generated and merged
  - K iterations, with each iteration, M dimensions are tried (M = # of features), and w is updated

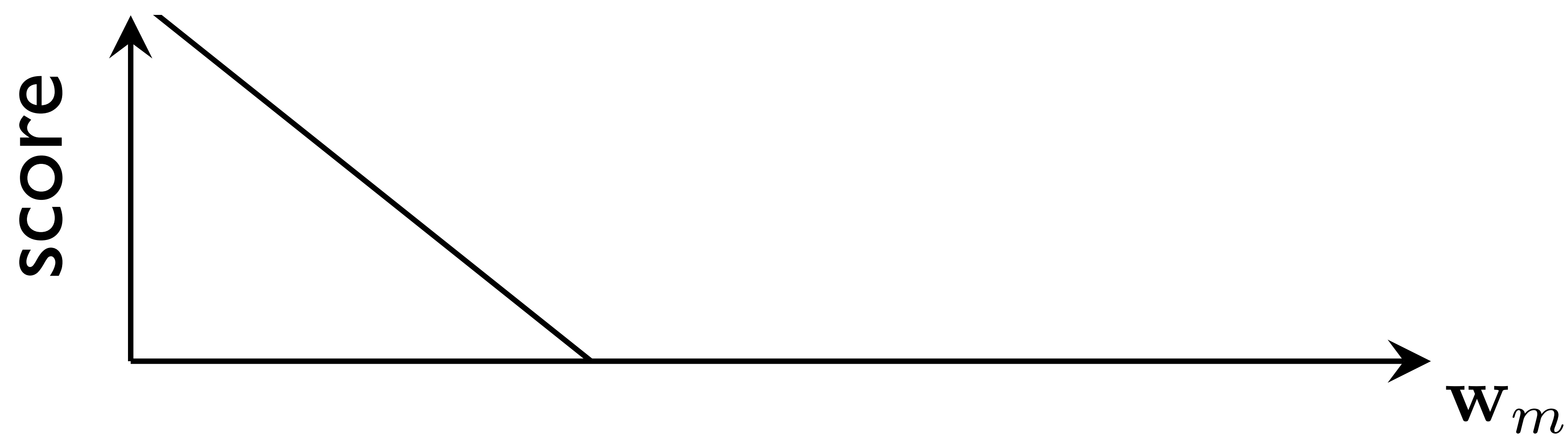
Efficient Line Search

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}_m^\top \cdot \underbrace{\mathbf{h}_m(\mathbf{e}, \mathbf{f}_s)}_{\text{slope}} + \underbrace{\mathbf{w}_{m\_}^\top \cdot \mathbf{h}_{m\_}(\mathbf{e}, \mathbf{f}_s)}_{\text{constant}}$$

- If we choose one dimension m, and others fixed, we can treat each hypothesis e as a "line"
- Compute convex hull of a set of "lines"

Efficient Line Search

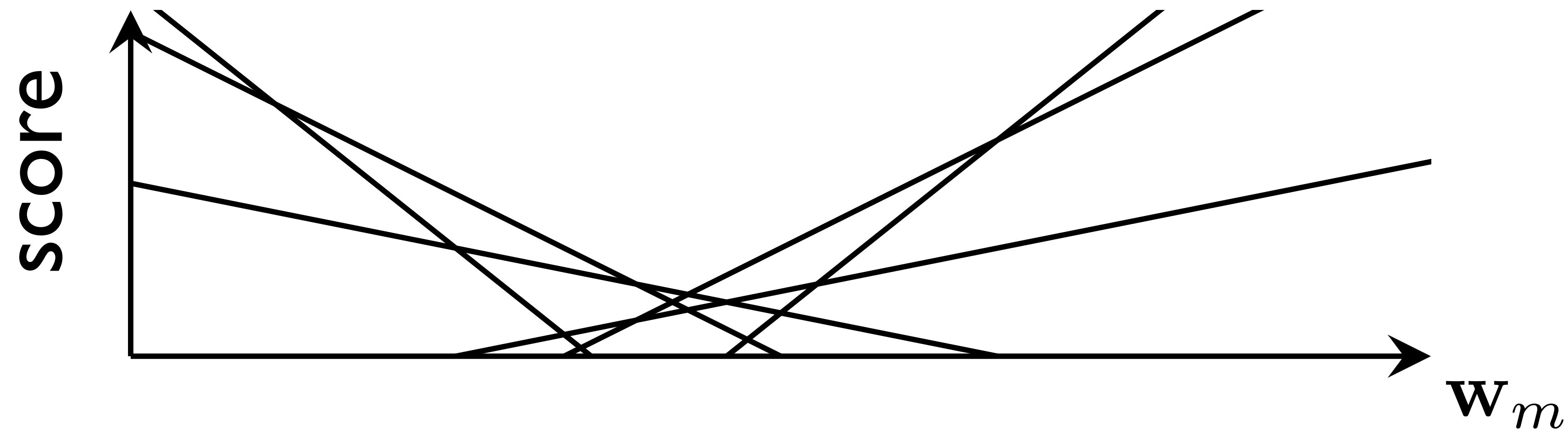
$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}_m^\top \cdot \underbrace{\mathbf{h}_m(\mathbf{e}, \mathbf{f}_s)}_{\text{slope}} + \underbrace{\mathbf{w}_{m\_}^\top \cdot \mathbf{h}_{m\_}(\mathbf{e}, \mathbf{f}_s)}_{\text{constant}}$$



- If we choose one dimension m, and others fixed, we can treat each hypothesis e as a "line"
- Compute convex hull of a set of "lines"

Efficient Line Search

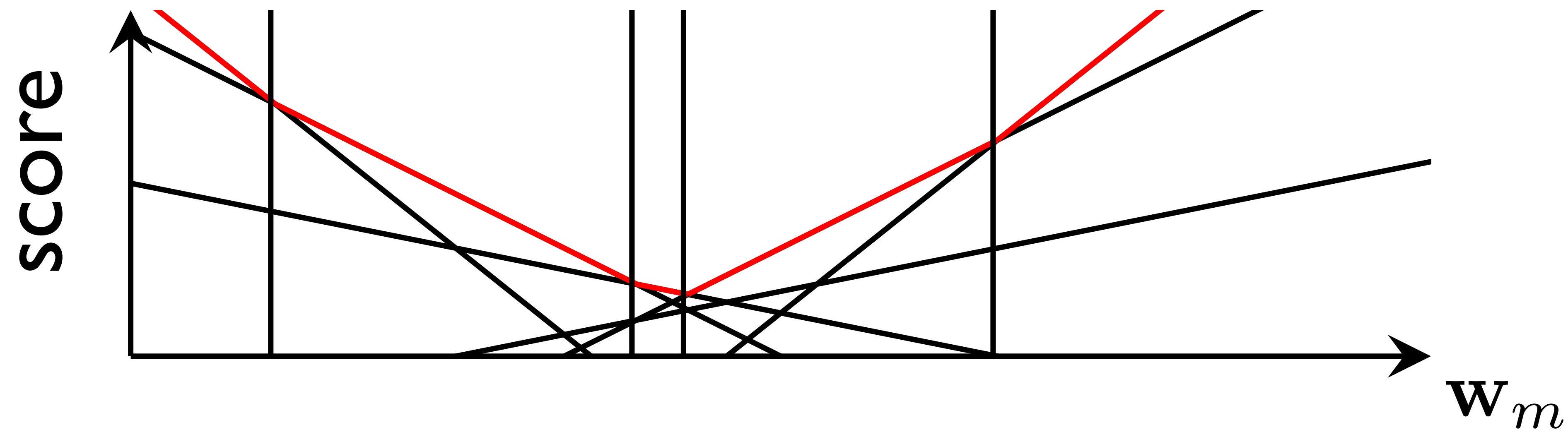
$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \mathbf{w}_m^\top \cdot \underbrace{\mathbf{h}_m(\mathbf{e}, \mathbf{f}_s)}_{\text{slope}} + \underbrace{\mathbf{w}_{m\_}^\top \cdot \mathbf{h}_{m\_}(\mathbf{e}, \mathbf{f}_s)}_{\text{constant}}$$



- If we choose one dimension m, and others fixed, we can treat each hypothesis e as a "line"
- Compute convex hull of a set of "lines"

Efficient Line Search

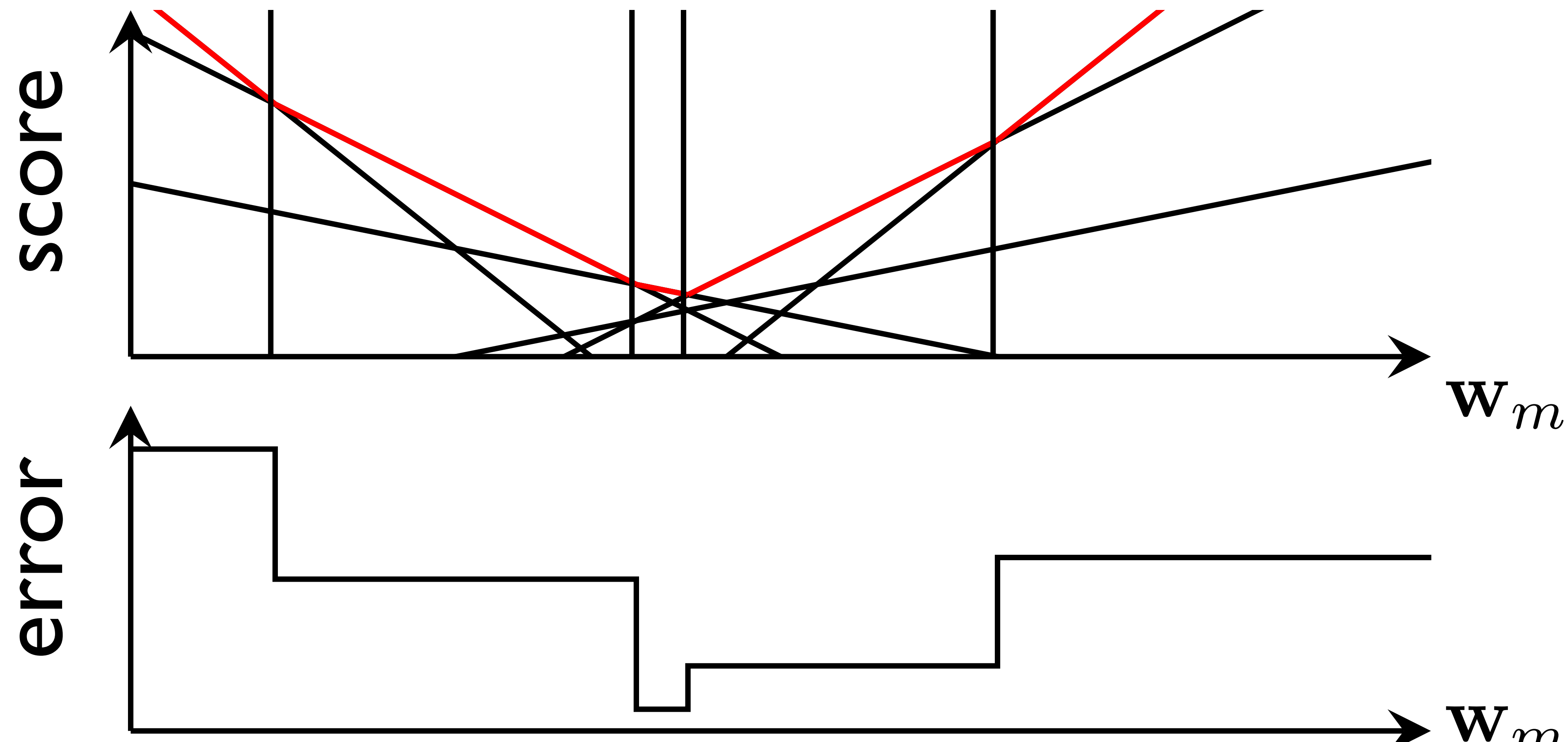
$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, \mathbf{w}_m^\top \cdot \underbrace{\mathbf{h}_m(\mathbf{e}, \mathbf{f}_s)}_{\text{slope}} + \underbrace{\mathbf{w}_{m\_}^\top \cdot \mathbf{h}_{m\_}(\mathbf{e}, \mathbf{f}_s)}_{\text{constant}}$$



- If we choose one dimension m, and others fixed, we can treat each hypothesis e as a "line"
- Compute convex hull of a set of "lines"

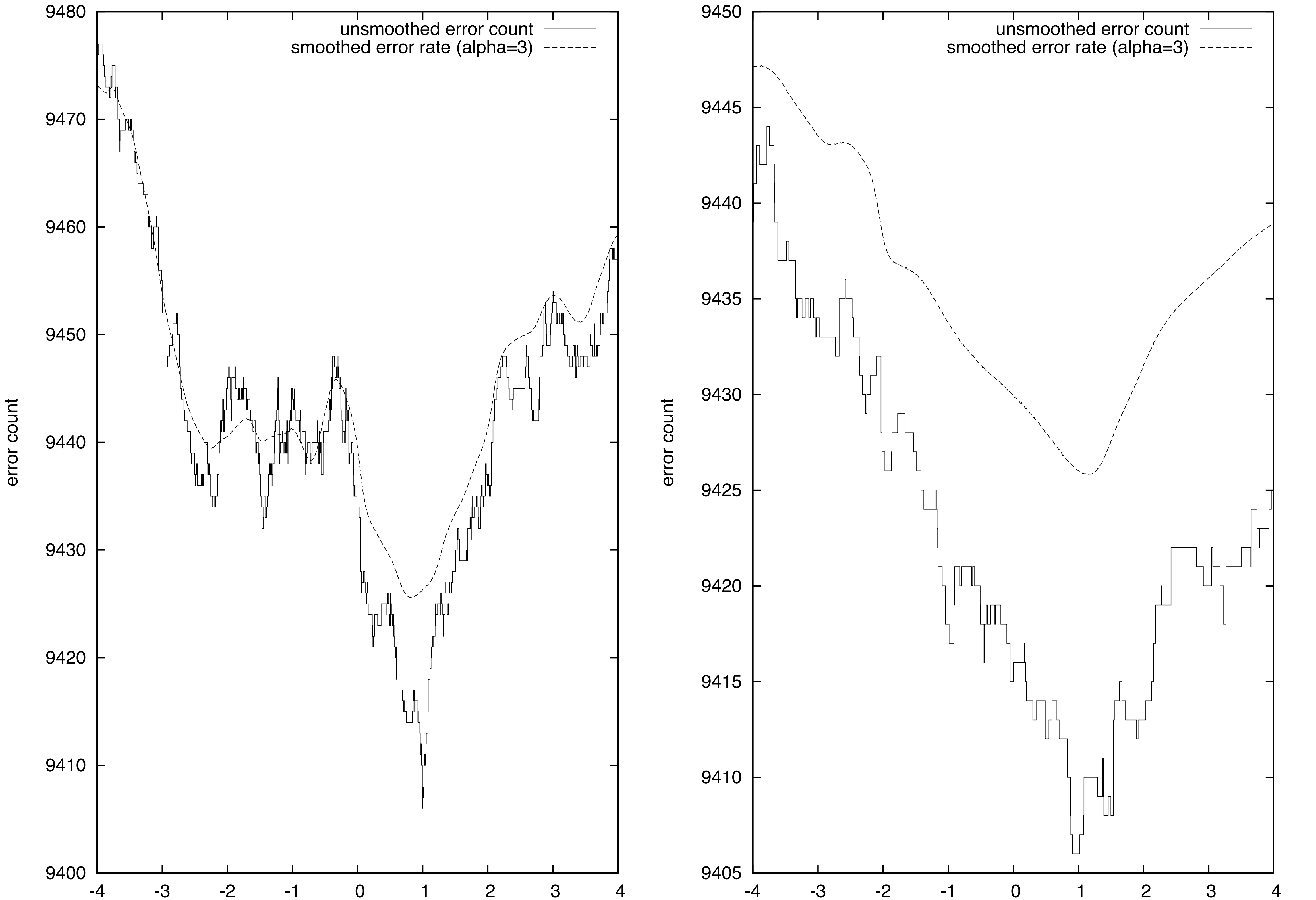
Efficient Line Search

$$\hat{\mathbf{e}} = \operatorname*{argmax}_{\mathbf{e}} \underbrace{\mathbf{w}_m^\top \cdot \mathbf{h}_m(\mathbf{e}, \mathbf{f}_s)}_{\text{slope}} + \underbrace{\mathbf{w}_{m\_}^\top \cdot \mathbf{h}_{m\_}(\mathbf{e}, \mathbf{f}_s)}_{\text{constant}}$$



- If we choose one dimension m, and others fixed, we can treat each hypothesis e as a "line"
- Compute convex hull of a set of "lines"

Error Surface



(Och, 2003)

MERT in Practice

- Many random starting points (Macherey et al., 2008; Moore and Quirk, 2008)
- Many random directions (Macherey et al., 2008)
- Error count smoothing (Cer et al., 2008)
- Regularization (Hayashi et al., 2009)
- Multi-dimensional search by efficiently computing convex hull (Galley and Quirk, 2011)

Maximum Entropy

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 - \sum_{s=1}^{S} \log \frac{\sum_{\mathbf{e}^* \in \mathsf{ORACLE}(\mathbf{f}_s)} \exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}^*, \mathbf{f}_s)\right)}{\sum_{\mathbf{e}' \in \mathsf{GEN}(\mathbf{f}_s)} \exp\left(\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}', \mathbf{f}_s)\right)}$$

- Minimize the negative log-likelihood of generating good translations (Och and Ney, 2002)
- ORACLE is a subset of GEN, a set of hypotheses with minimum loss
- Optimized by L-BFGS or SGD
- Potentially large # of features as in NLP tasks

Why Not MaxEnt?

error criterion used in training	mWER [%]	mPER [%]	BLEU [%]	NIST	# words
confidence intervals	+/- 2.7	+/- 1.9	+/- 0.8	+/- 0.12	-
MMI	**68.0**	*51.0*	11.3	5.76	21933
mWER	*68.3*	*50.2*	13.5	6.28	22914
smoothed-mWER	*68.2*	*50.2*	13.2	6.27	22902
mPER	*70.2*	*49.8*	15.2	*6.71*	24399
smoothed-mPER	*70.0*	**49.7**	15.2	*6.69*	24198
BLEU	76.1	53.2	**17.2**	6.66	28002
NIST	73.3	*51.5*	*16.4*	**6.80**	26602

- In Och and Ney (2002), they used
    - WER to select oracle translations
    - n-best merging approach to approximate summation as in MERT

Large Margin

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{s=1}^{S} \sum_{\mathbf{e}_s^*} \sum_{\mathbf{e}'_s} \xi_{s,\mathbf{e}_s^*,\mathbf{e}'_s}$$

$$\mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}_s^*, \mathbf{f}_s) - \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}'_s, \mathbf{f}_s) \geq l(\mathbf{e}'_s, \mathbf{e}_s^*) - \xi_{s,\mathbf{e}_s^*,\mathbf{e}'_s}$$

$$\mathbf{e}_s^* \in \mathsf{ORACLE}(\mathbf{f}_s)$$

$$\mathbf{e}'_s \in \mathsf{GEN}(\mathbf{f}_s)$$

- Structured output learning approach
- Very hard to enumerate all possible e' and oracle translations e*
- Solution: online learning or n-best approximation

Online Learning

Require: $\{(\mathbf{f}_s, \mathbf{e}_s)\}_{s=1}^{S}$

$\mathbf{w}^1 = \{0\}$
$t = 1$
for $1 \ldots N$ **do**
  $s \sim \mathrm{random}(1, S)$
  $\hat{\mathbf{e}} \in \mathsf{GEN}(\mathbf{f}_s, \mathbf{w}^{t-1})$
  if $l(\hat{\mathbf{e}}, \mathbf{e}_s) \geq 0$ **then**
    $\mathbf{w}^{t+1} = \mathbf{w}^t + \mathbf{h}(\mathbf{e}_s, \mathbf{f}_s) - \mathbf{h}(\hat{\mathbf{e}}, \mathbf{f}_s)$
    $t = t + 1$
  end if
end for
return $\mathbf{w}^t$ or $\frac{1}{N} \sum_{i=1}^{N} \mathbf{w}^j$

- Averaged perceptron (Liang et al., 2006)
- Scale to large data, but each iteration requires decoding + weight update

Online Large Margin

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}'} \frac{\lambda}{2}||\mathbf{w}' - \mathbf{w}||^2 + \max\left(l_s - \mathbf{w}'^\top \cdot \Delta\mathbf{h}_s\right)$$

$$\begin{aligned}
\hat{\mathbf{e}}_s &= \operatorname*{argmax}_{e} \mathbf{w}^\top \cdot \mathbf{h}(\mathbf{e}, \mathbf{f}_s) \\
l_s &= l(\hat{\mathbf{e}}_s) - l(\mathbf{e}_s^*) \\
\Delta\mathbf{h}_s &= \mathbf{h}(\hat{\mathbf{e}}_s, \mathbf{f}_s) - \mathbf{h}(\mathbf{e}^*, \mathbf{f}_s)
\end{aligned}$$

- line 7 is replaced by the solution of the above equation
- Still, requires decoding + update in each iteration
- Hard to determine when to stop (watch another dev data)

Ranking Approach

$$\hat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \sum_{s=1}^{S} \sum_{\mathbf{e}''_s} \sum_{\mathbf{e}'_s} \xi_{s,\mathbf{e}''_s,\mathbf{e}'_s}$$

$$-\log\left(1 + \exp(-\mathbf{w}^\top \cdot \Delta\mathbf{h}_{\mathbf{e}''_s,\mathbf{e}'_s})\right) \geq -\xi_{s,\mathbf{e}''_s,\mathbf{e}'_s}$$

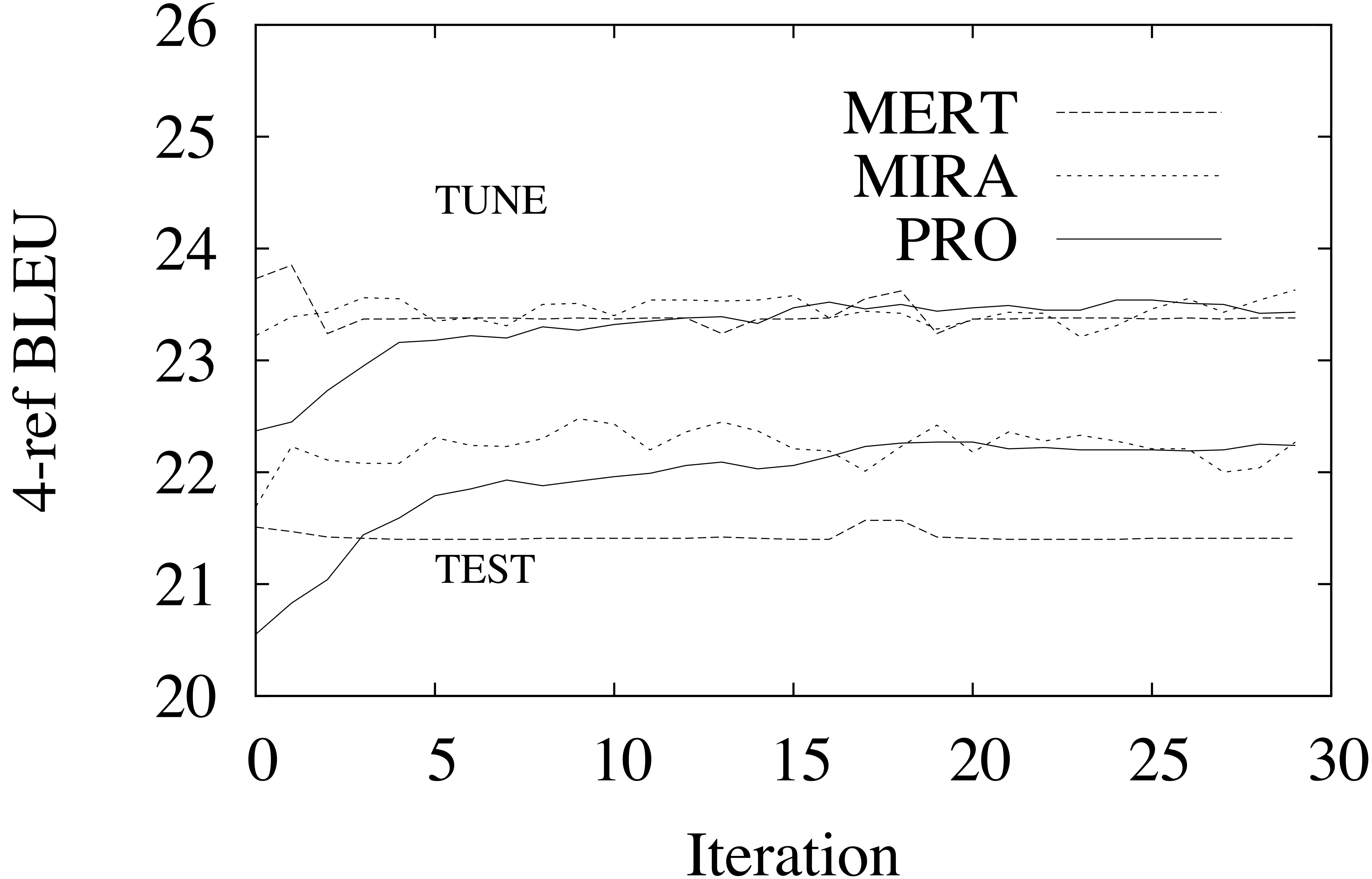
$$\mathbf{e}''_s, \mathbf{e}'_s \in \mathsf{GEN}(\mathbf{f}_s)$$

$$l(\mathbf{e}'_s, \mathbf{e}''_s) > 0$$

$$\Delta\mathbf{h}_{\mathbf{e}''_s,\mathbf{e}'_s} = \mathbf{h}(\mathbf{e}''_s, \mathbf{f}_s) - \mathbf{h}(\mathbf{e}'_s, \mathbf{f}_s)$$

- An n-best approximation approach (Hopkins and May, 2011)
- Pair-wise comparison of all the hypotheses
- logistic-loss (or 0-1 loss): use an off-the-shelf binary classifier

Results



- Reranking is competitive to MERT and MIRA, and scales to large # of features

Conclusion

- Training: How to learn phrases and parameters (Φ and h)?
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (w)?

References

- P. Koehn, *Statistical Machine Translation*. Cambridge University Press, 2009.
- L. Huang and D. Chiang, ``Forest rescoring: Faster decoding with integrated language models," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (Prague, Czech Republic), pp. 144--151, Association for Computational Linguistics, June 2007.
- D. Marcu and W. Wong, ``A phrase-based, joint probability model for statistical machine translation," in *Proc. of EMNLP-2002*, (Philadelphia, PA), July 2002.
- P. Blunsom, T. Cohn, C. Dyer, and M. Osborne, ``A gibbs sampler for phrasal synchronous grammar induction," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, (Suntec, Singapore), pp. 782--790, Association for Computational Linguistics, August 2009.
- G. Neubig, T. Watanabe, E. Sumita, S. Mori, and T. Kawahara, ``An unsupervised model for joint phrase alignment and extraction," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 632-641, Association for Computational Linguistics, June 2011.

References

- K. Knight, ``Decoding complexity in word-replacement translation models," *Comput. Linguist.*, vol. 25, pp. 607-615, December 1999.
- F. J. Och, ``Minimum error rate training in statistical machine translation," in *Proc. of ACL 2003*, (Sapporo, Japan), pp. 160--167, July 2003.
- F. J. Och and H. Ney, ``Discriminative training and maximum entropy models for statistical machine translation," in *Proc. of ACL 2002*, (Philadelphia, PA), pp. 295--302, 2002.
- T. Watanabe, J. Suzuki, H. Tsukada, and H. Isozaki, ``Online Large-Margin Training for Statistical Machine Translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, (Prague, Czech Republic), pp. 764--773, June 2007.
- D. Chiang, Y. Marton, and P. Resnik, ``Online large-margin training of syntactic and structural translation features," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, (Honolulu, Hawaii), pp. 224-233, Association for Computational Linguistics, October 2008.
- M. Hopkins and J. May, ``Tuning as ranking," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (Edinburgh, Scotland, UK.), pp. 1352-1362, Association for Computational Linguistics, July 2011.

References

- W. Macherey, F. Och, I. Thayer, and J. Uszkoreit, ``Lattice-based minimum error rate training for statistical machine translation," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, (Honolulu, Hawaii), pp. 725--734, Association for Computational Linguistics, October 2008.

- R. C. Moore and C. Quirk, ``Random restarts in minimum error rate training for statistical machine translation," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, (Manchester, UK), pp. 585--592, Coling 2008 Organizing Committee, August 2008.

- D. Cer, D. Jurafsky, and C. D. Manning, ``Regularization and search for minimum error rate training," in *Proceedings of the Third Workshop on Statistical Machine Translation*, (Columbus, Ohio), pp. 26--34, Association for Computational Linguistics, June 2008.

- K. Hayashi, T. Watanabe, H. Tsukada, and H. Isozaki, ``Structural Support Vector Machines for Log-Linear Approach in Statistical Machine Translation," in *Proc. of the International Workshop on Spoken Language Translation*, (Tokyo, Japan), pp. 144--151, 2009.

- M. Galley and C. Quirk, ``Optimal search for minimum error rate training," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, (Edinburgh, Scotland, UK.), pp. 38--49, Association for Computational Linguistics, July 2011.

- P. Liang, A. Bouchard-Coˆte ´, D. Klein, and B. Taskar, ``An end-to-end discriminative approach to machine translation," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, (Sydney, Australia), pp. 761--768, Association for Computational Linguistics, July 2006.