Elastic-substitution decoding for Hierarchical SMT: efficiency, richer search and double labels (Presentation MT Summit, Nagoya, Japan)

Gideon Maillette de Buy Wenniger[1], Khalil Sima'an[2], Andy Way[1]

gemdbw AT gmail.com k.simaan AT uva.nl

http://computing.dcu.ie/~gmdbwenniger/

[1]ADAPT Center, School of Computing
Dublin City University, Ireland

[2]Statistical Language Processing and Learning Lab
Institute for Logic Language and Computation
University of Amsterdam, the Netherlands

Overview

- Hierarchical statistical machine translation (SMT) has been effective
- Labeled Hierarchical SMT even more so
- Chiang (2010) introduced soft-matching to overcome problems with strict-enforced label-matching constraints

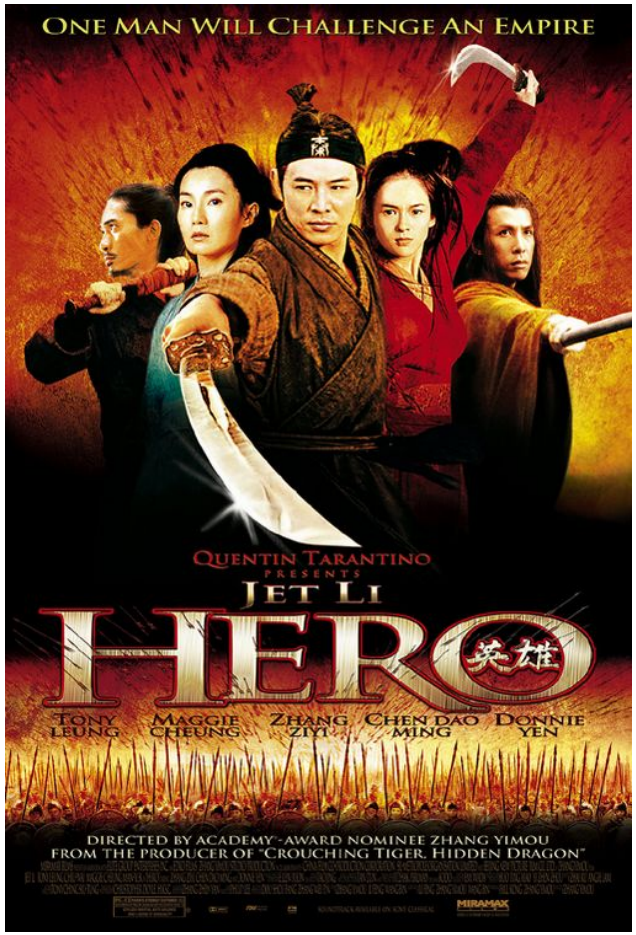
In this work:

- Efficient soft-matching: howto
- More evidence superiority of soft-matching over strict-matching
- Better search during decoding
  - Ensure label variety $\rightarrow$ ensure exploration of matching substitutions
- Combining multiple labels

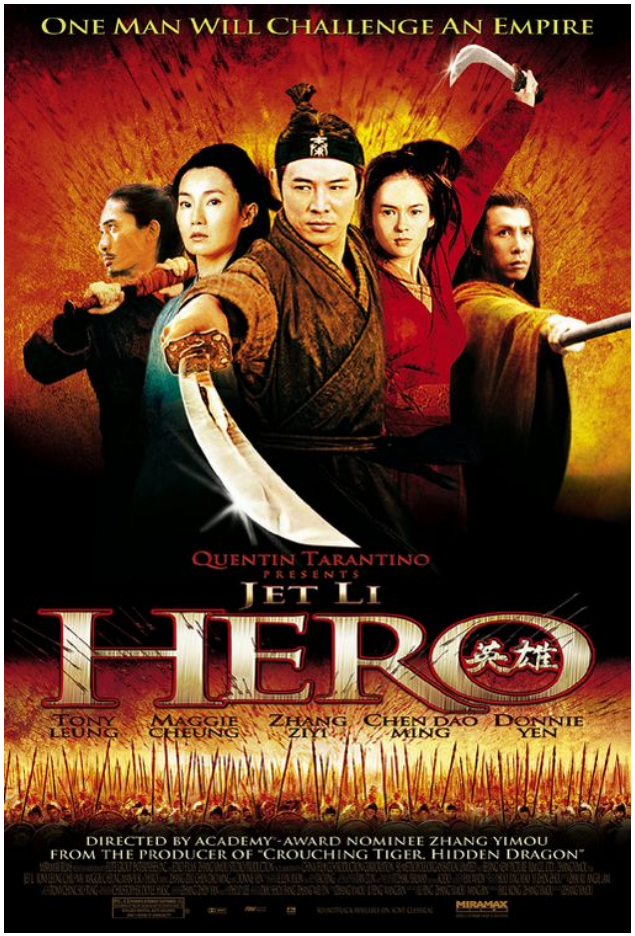
Part 1: Hierarchical Statistical Machine Translation (Hiero)

What is Hiero?

What is Hiero?



What is Hiero?



Eeuh . . . no. Let's try again.

(Hierarchical) Statistical Machine Translation - Basics

Approximating best translation by best derivation

$$\arg\max_{\mathbf{t}} P(\mathbf{t} \mid \mathbf{s}) = \arg\max_{\mathbf{t}} \sum_{\mathbf{d} \in G} P(\mathbf{t}, \mathbf{d} \mid \mathbf{s}) \quad (1)$$

$$\approx \arg\max_{\mathbf{d} \in G} P(\mathbf{t}, \mathbf{d} \mid \mathbf{s}) \quad (2)$$

Log-linear Model:

$$\arg\max_{\mathbf{d} \in G} P(\mathbf{t}, \mathbf{d} \mid \mathbf{s}) \approx \arg\max_{\mathbf{d} \in G} \sum_{i=1}^{|\Phi(\mathbf{d})|} \lambda_i \times \phi_i \quad (3)$$

From SMT to Hierarchical SMT

- Generalize phrase pairs by introducing phrases with variables
  - Rule-table becomes synchronous context-free grammar
- Decoding: search for best synchronous derivation of the input
- Approximate intersection with the language model: cube pruning

Types of Hiero Rules

$X \rightarrow \langle \alpha,\ \gamma \rangle$	Phrase Pair
$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \beta,\ \delta\ X_{\boxed{1}}\ \zeta \rangle$	One gap rule
$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \beta\ X_{\boxed{2}}\ \gamma\ ,\ \delta\ X_{\boxed{1}}\ \zeta\ X_{\boxed{2}}\ \eta\ \rangle$	Two gaps monotone rule
$X \rightarrow \langle \alpha\ X_{\boxed{1}}\ \beta\ X_{\boxed{2}}\ \gamma\ ,\ \delta\ X_{\boxed{2}}\ \zeta\ X_{\boxed{1}}\ \eta\ \rangle$	Two gaps inverted rule

Part 2:
Labeling Hiero

Overview used labels

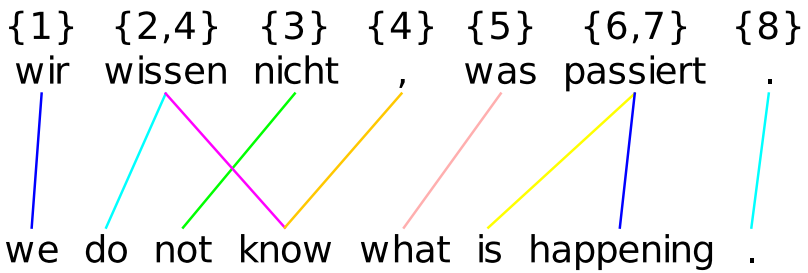
1. Syntax-augmented machine translation labels (SAMT)
2. Pos-tags from phrase boundaries
3. Bilingual Phrase Reordering labels

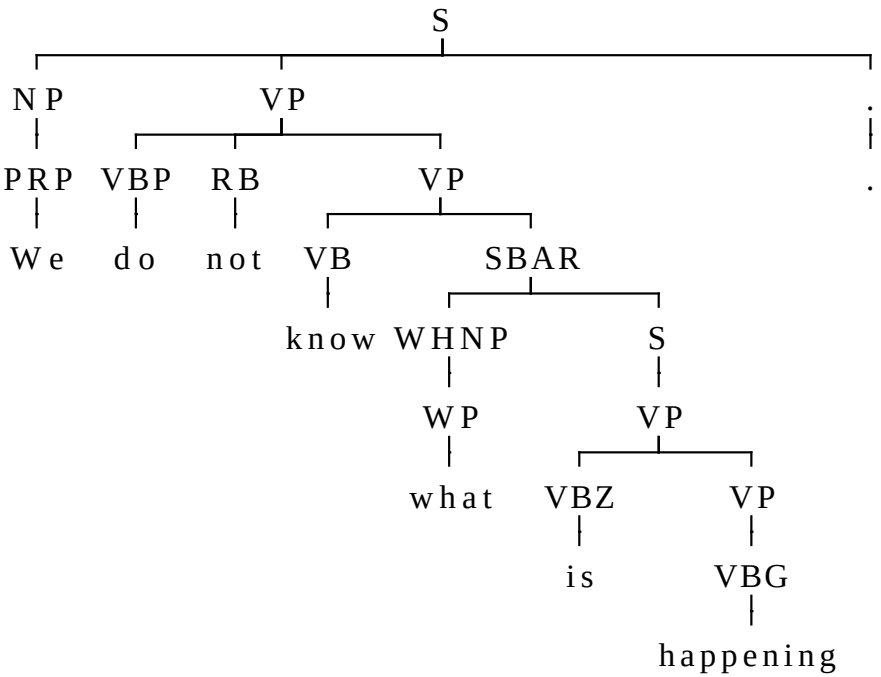
Syntax-Augmented Machine Translation (SAMT)

1. label constituent phrases only
   - Risk coverage loss in strictly syntactic systems
     "Re-structuring, Re-labeling, and Re-aligning for Syntax-Based Machine Translation" (Wang et.al, 2010)
2. add syntax **without coverage loss** w.r.t Hiero
   relaxed syntactic labels akin to Combinatorial Categorial Grammar

C	:	NP – the great wall
C1+C2	:	NP+VB – *she+went*
C1/C2	:	NP/NN – *the great (/wall)*
C1\C2	:	DT\NP – *(the\) great wall*
default	:	FAIL

Example

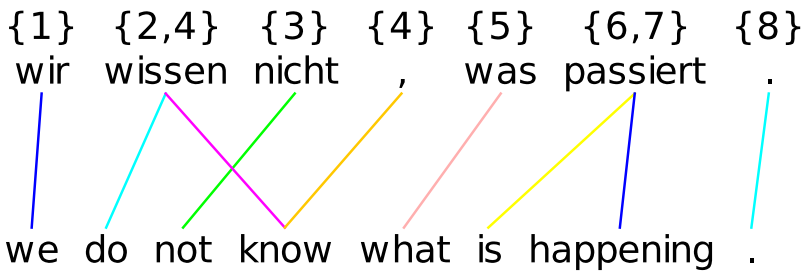


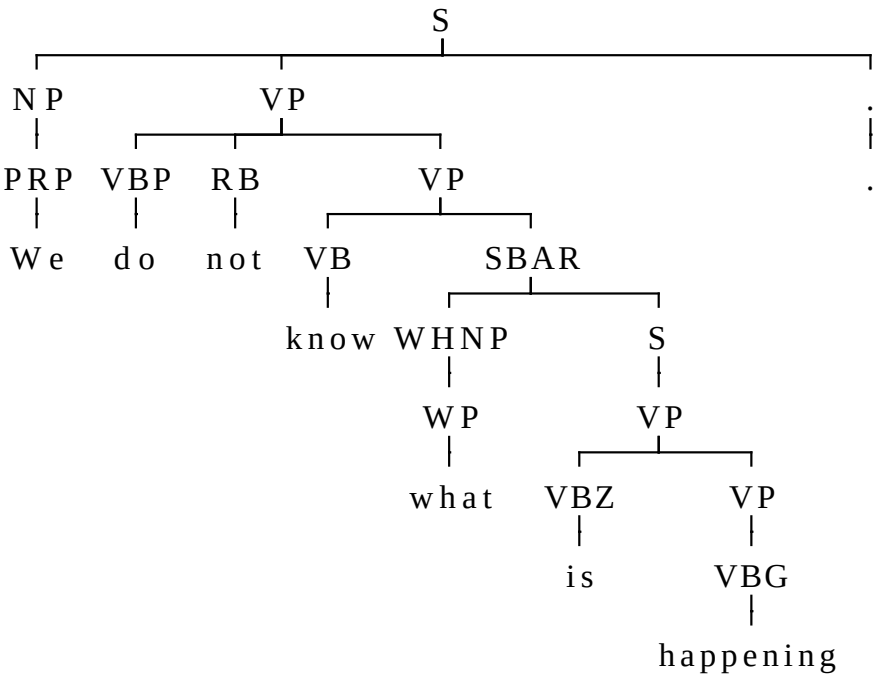


Which label for:

- we / wir (NP:PRP)
- do not know / ...
- is happening . / ...
- do ... happening . / ...
- we do not / ...

Example





Which label for:

- NP:PRP $\rightarrow$ we / wir
- VP/SBAR $\rightarrow$ do not know
- S:VP+. $\rightarrow$ is happening .
- VP+. $\rightarrow$ do ... happening .
- FAIL $\rightarrow$ we do not

Phrase-boundary labels

- Pos-tag the target-side of the corpus
- Concatenate the pos-tags for the phrase-boundary words to form labels
- Similar in spirit to SAMT
- In contrast to SAMT, yields a real (i.e. not “FAIL”) label for any span

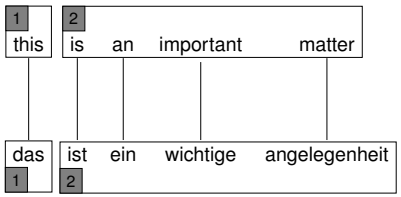
Bilingual Phrase Reordering label categories

- Phrase-Centric
- Parent-Relative

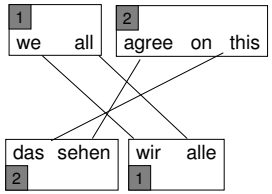
Phrase-centric reordering labels

- Complexity relation between base phrase and direct children, when decomposing the phrase, determines label
- Five cases distinguished, ordered by increasing complexity

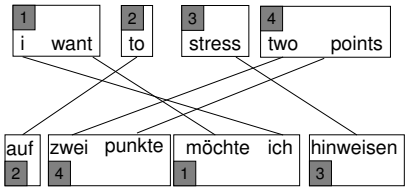
Monotonic



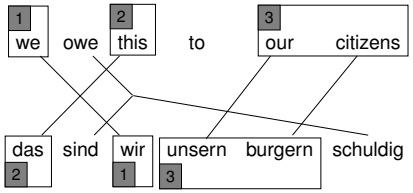
Inversion



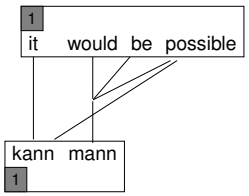
Permutation



Complex



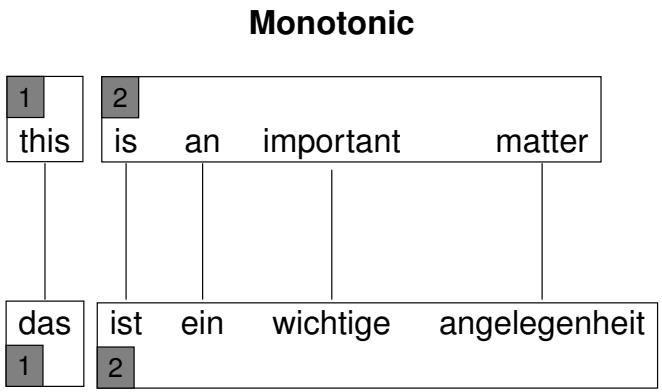
Atomic

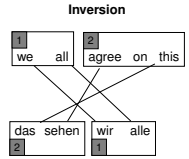


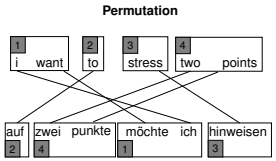
Known labels from ITG and Phrase pair Theory

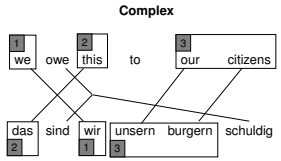
Monotonic

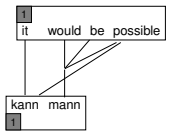
- *Monotonic*: If the alignment can be split into two monotonically ordered parts.





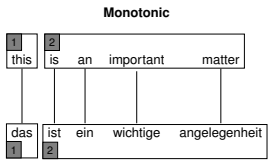


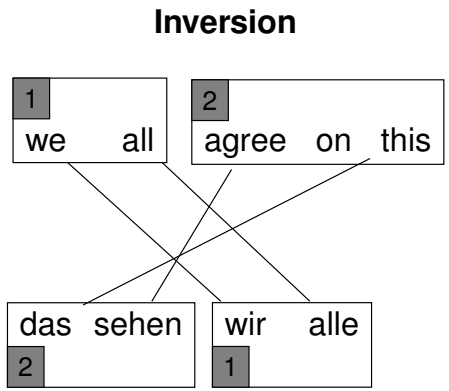


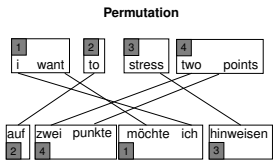


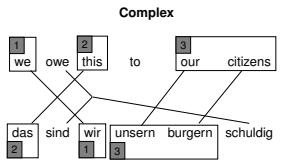
Inverted

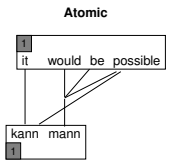
- *Inverted*: If the alignment can be split into two inverted parts.





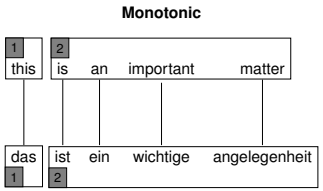


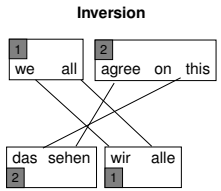


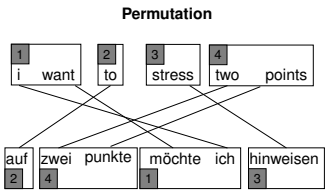


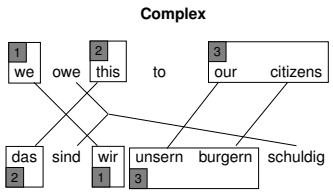
Atomic

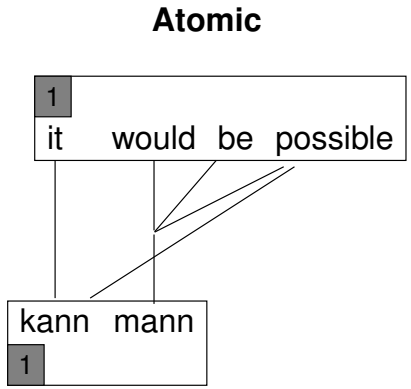
- *Atomic*: If the alignment does not allow the existence of smaller (child) phrase pairs.







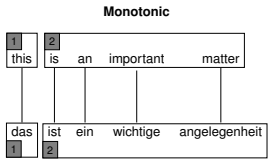


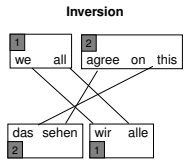


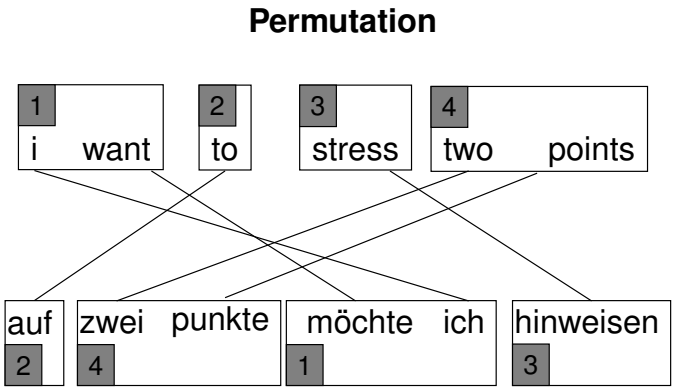
New labels based on HATs

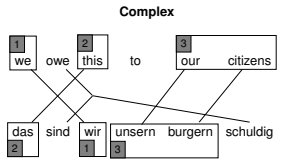
Permutation

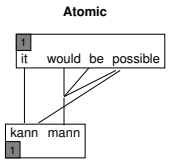
- *Permutation*: If the alignment can be factored as a permutation of more than 3 parts.





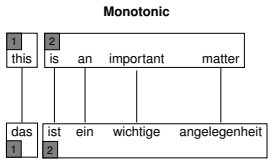


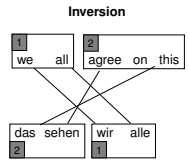


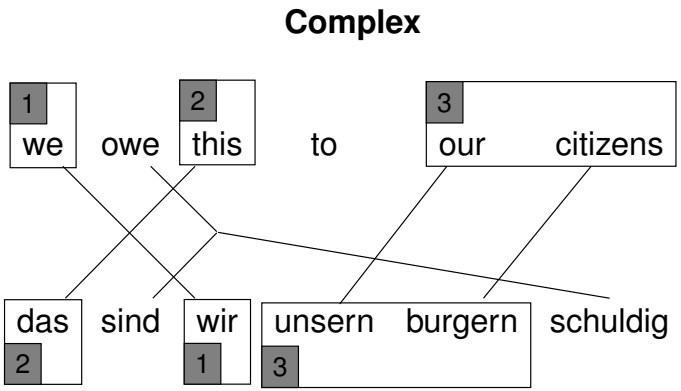


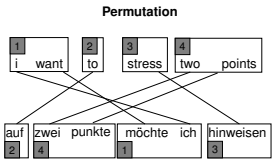
Complex

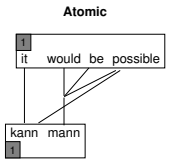
- *Complex*: No alignment factorization as a permutation of parts, but smaller phrase pair is contained (i.e., it is composite).







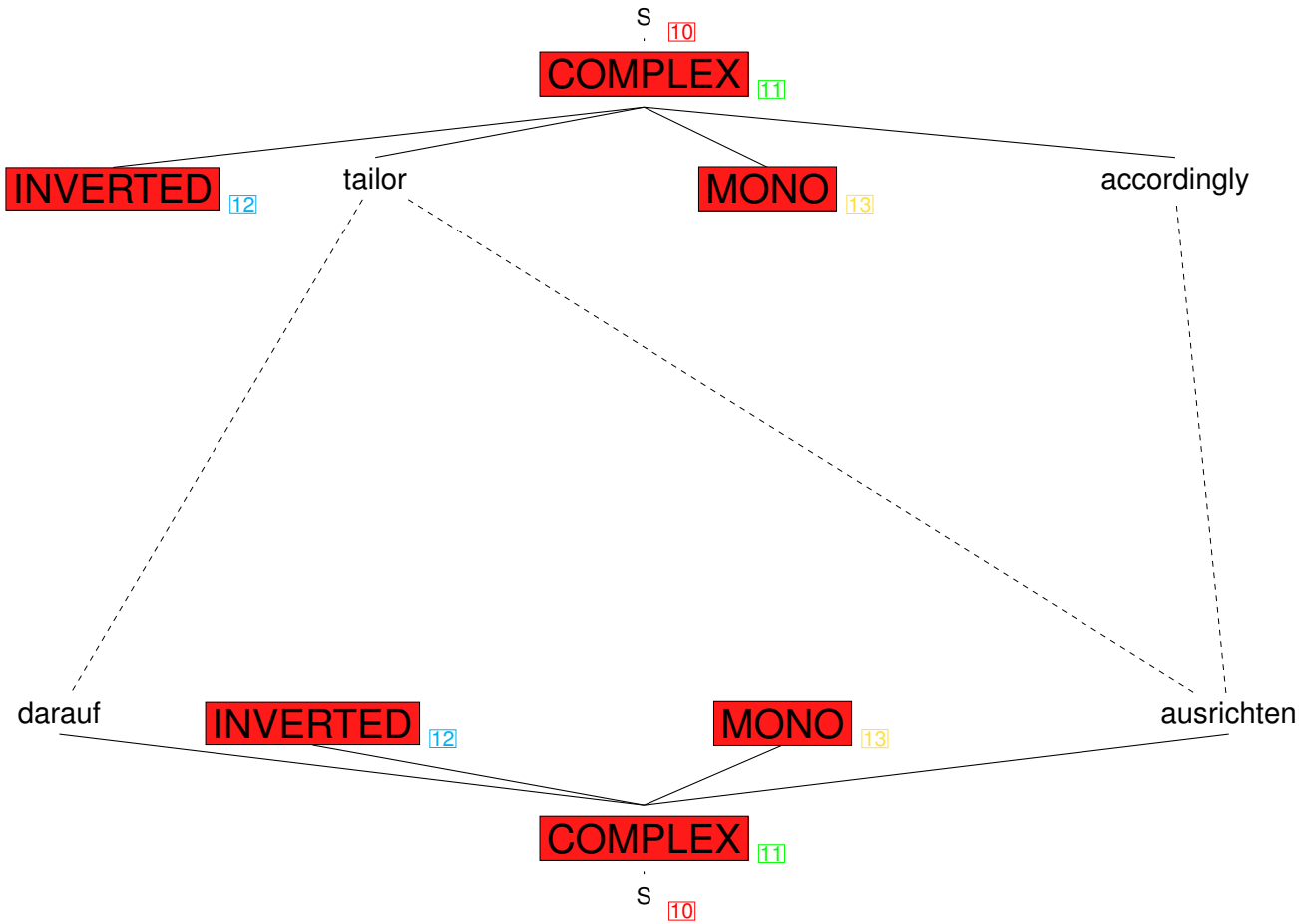




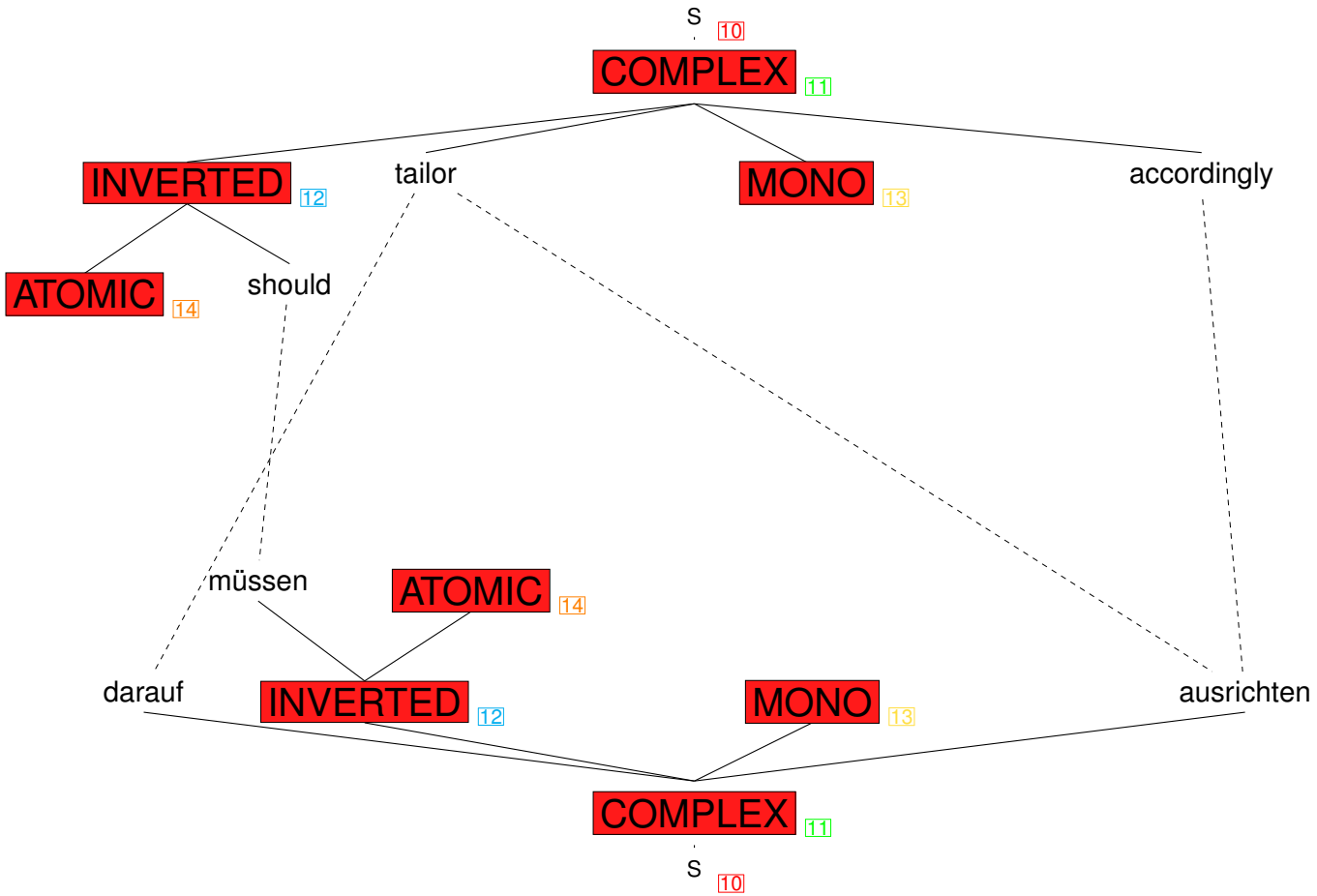
Phrase-Centric labeled derivation

Phrase-Centric labeled derivation

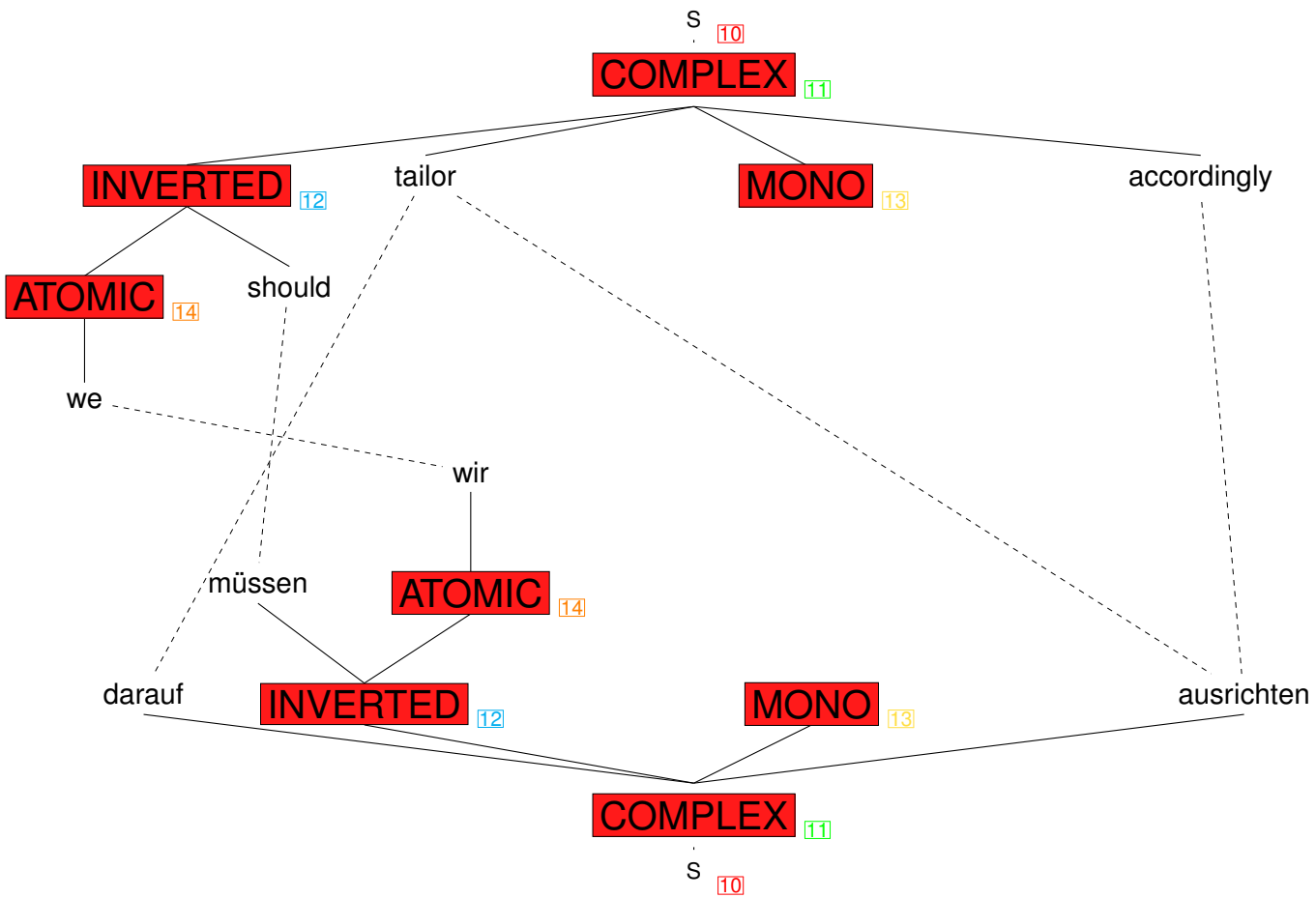
Phrase-Centric labeled derivation



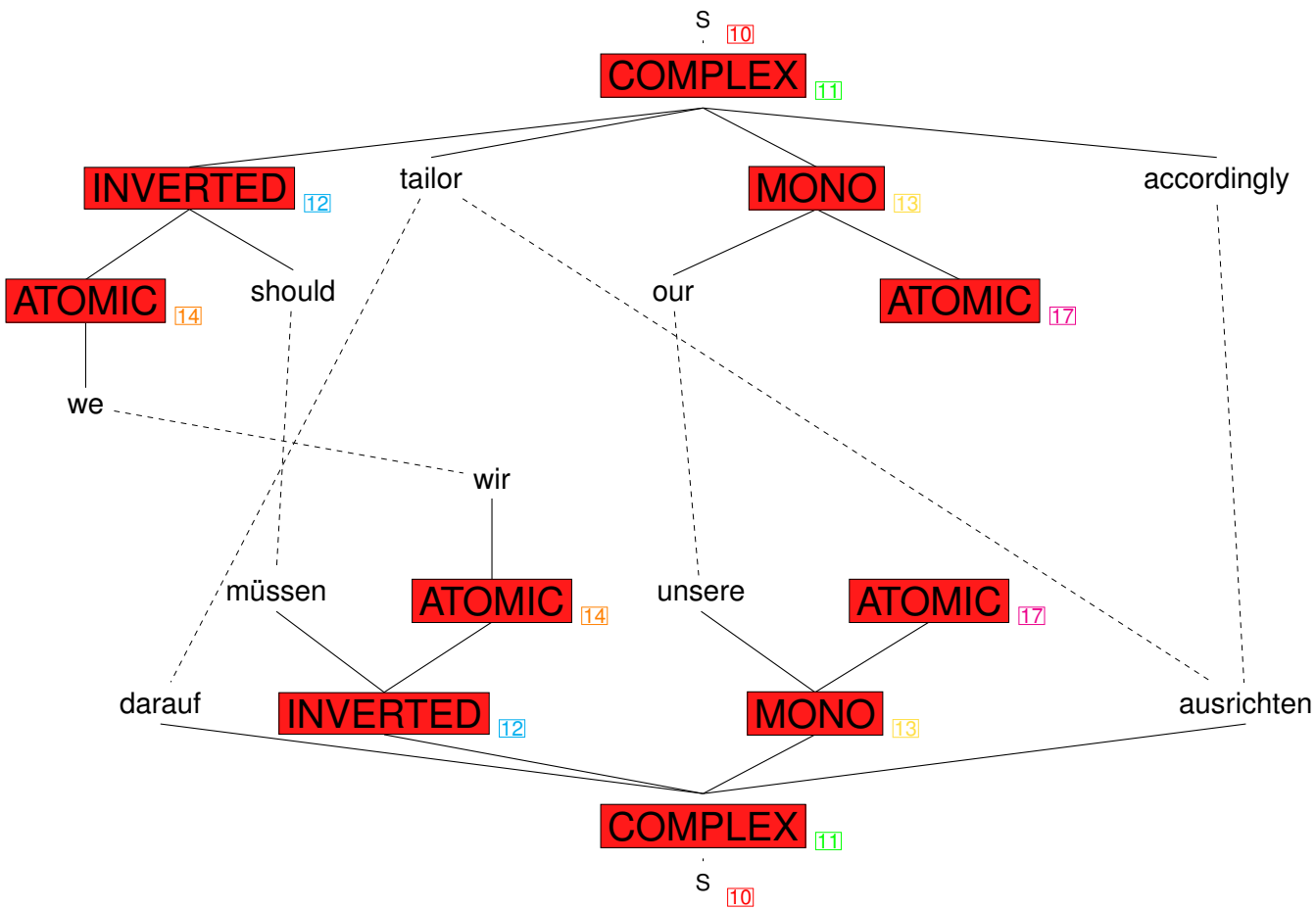
Phrase-Centric labeled derivation



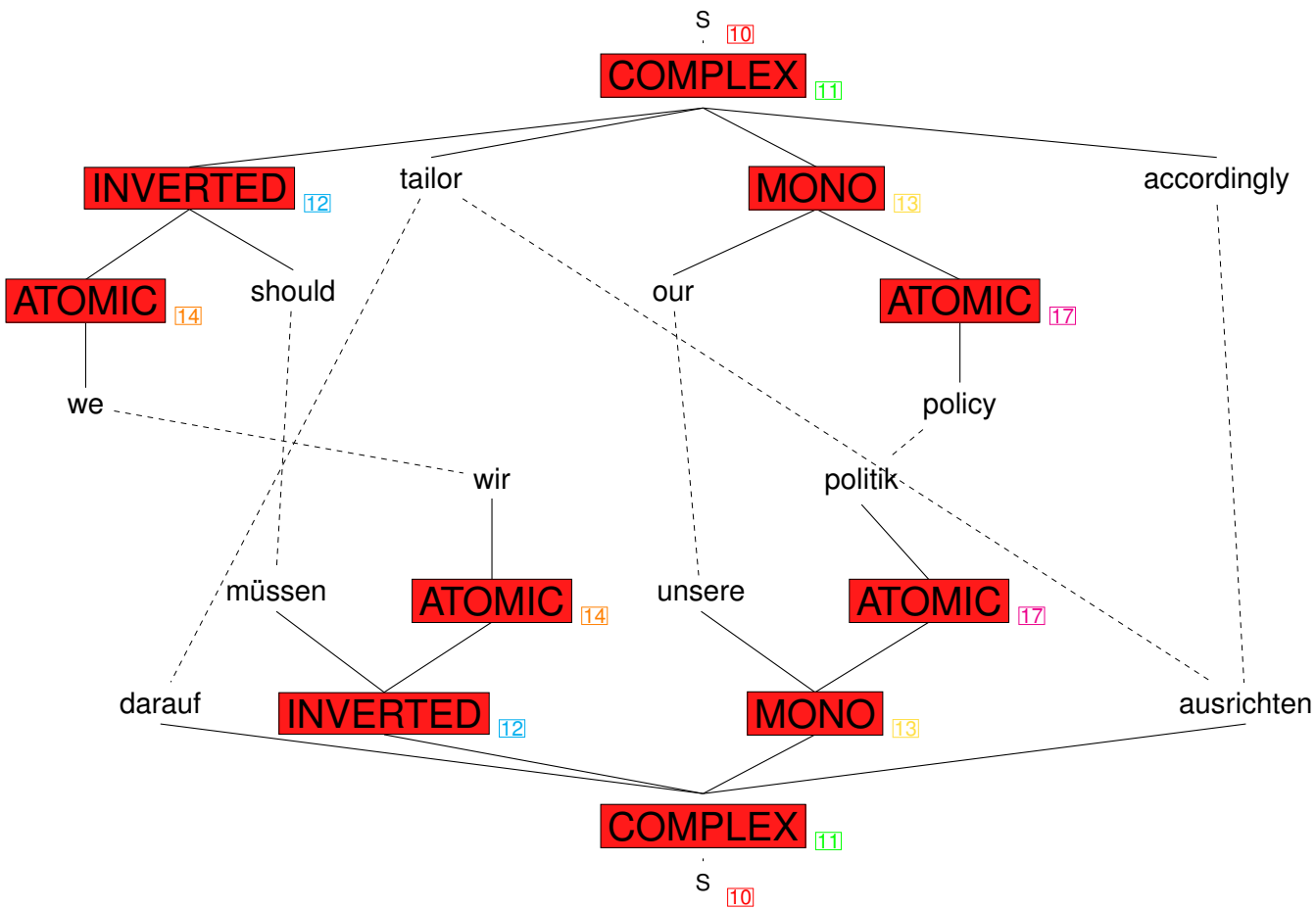
Phrase-Centric labeled derivation



Phrase-Centric labeled derivation



Phrase-Centric labeled derivation



Parent-Relative reordering labels

- Describe type of reordering relative to embedding “parent” phrase
- First-order view on reordering
- 9 distinct labels indicating different forms of monontone, inverted and discontinous reordering relations

Part 3:
Soft-matching for labeled Hiero

Motivation soft-matching

- With strict-matching, many labeled variants of the same rule, and hence Hiero derivation exist
  - Further increase spurious ambiguity
- Strict-matching blocks prospective valid translations
- Quality of labels not always assured
- Some labels may be partly interchangeable, especially for Heuristic labeling schemes

Two variants of “label relaxation”

1. Preference Grammars
2. Soft-matching / Elastic-substitution decoding (Chiang,2010)

Preference Grammars (Venugopal et.al, 2009)

- “Pack” the distinct labeled versions of a Hiero rule type
- Associate with every Hiero rule type a distribution over different labelings
- Use dynamic programming to incrementally produce a kind of in-product between label distributions of substituting and substituted-to rules
- Loosely: approximates summation derirvations over alternatively labeled-versions same Hiero rule type

Soft matching

- Any label may match any other label
- Use per Hiero rule typ only single most frequent version amongst all variants
- Add features that mark specific label substitutions
  - Allows tuning to learn preference over substitutions

How to make it fast?

Requirements:

- Quickly retrieve all rules, while ignoring the labels
- Still distinguish between glue nonterminals (“GOAL”) and all other notnterminals

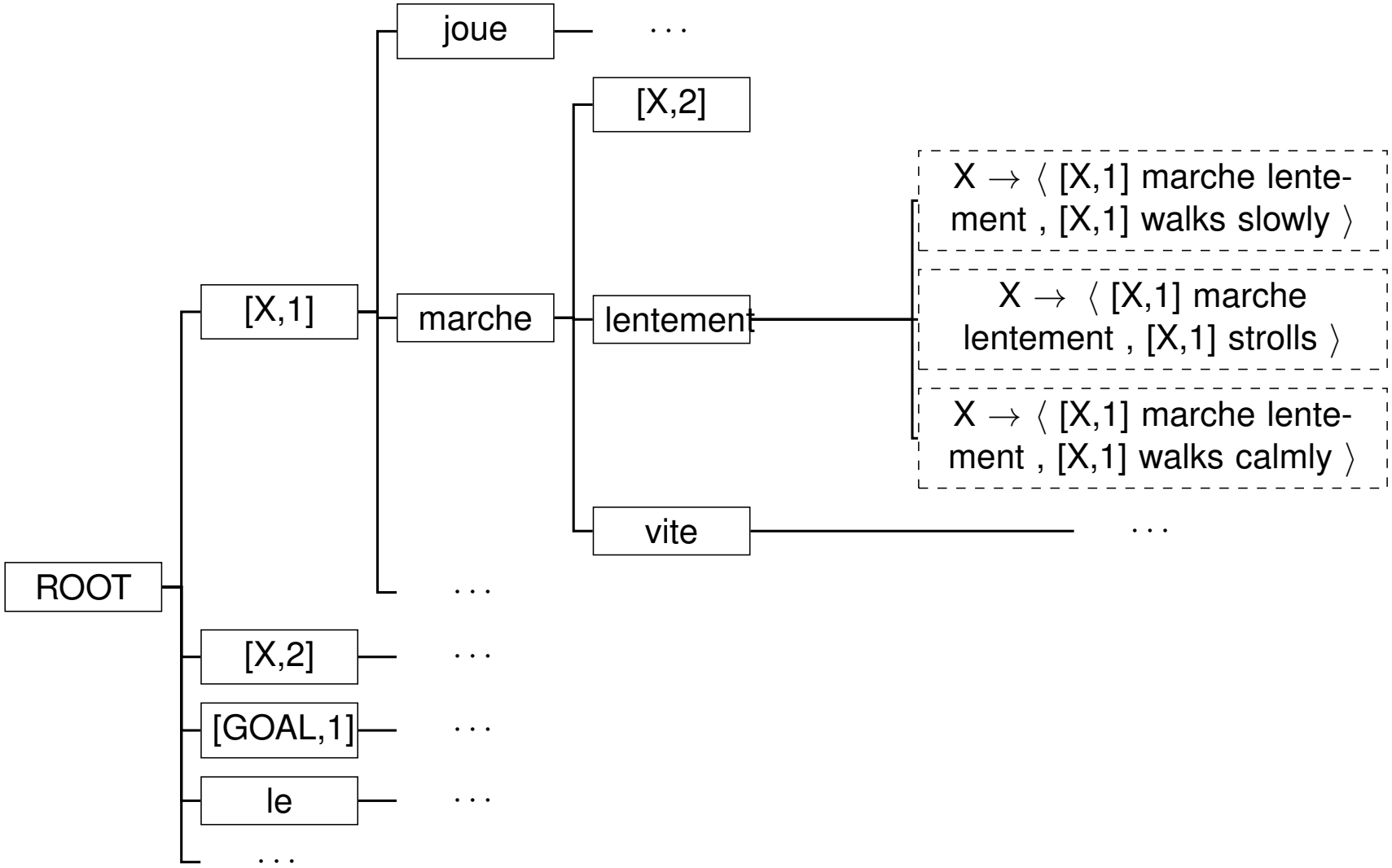
The solution . . .

Trie it!

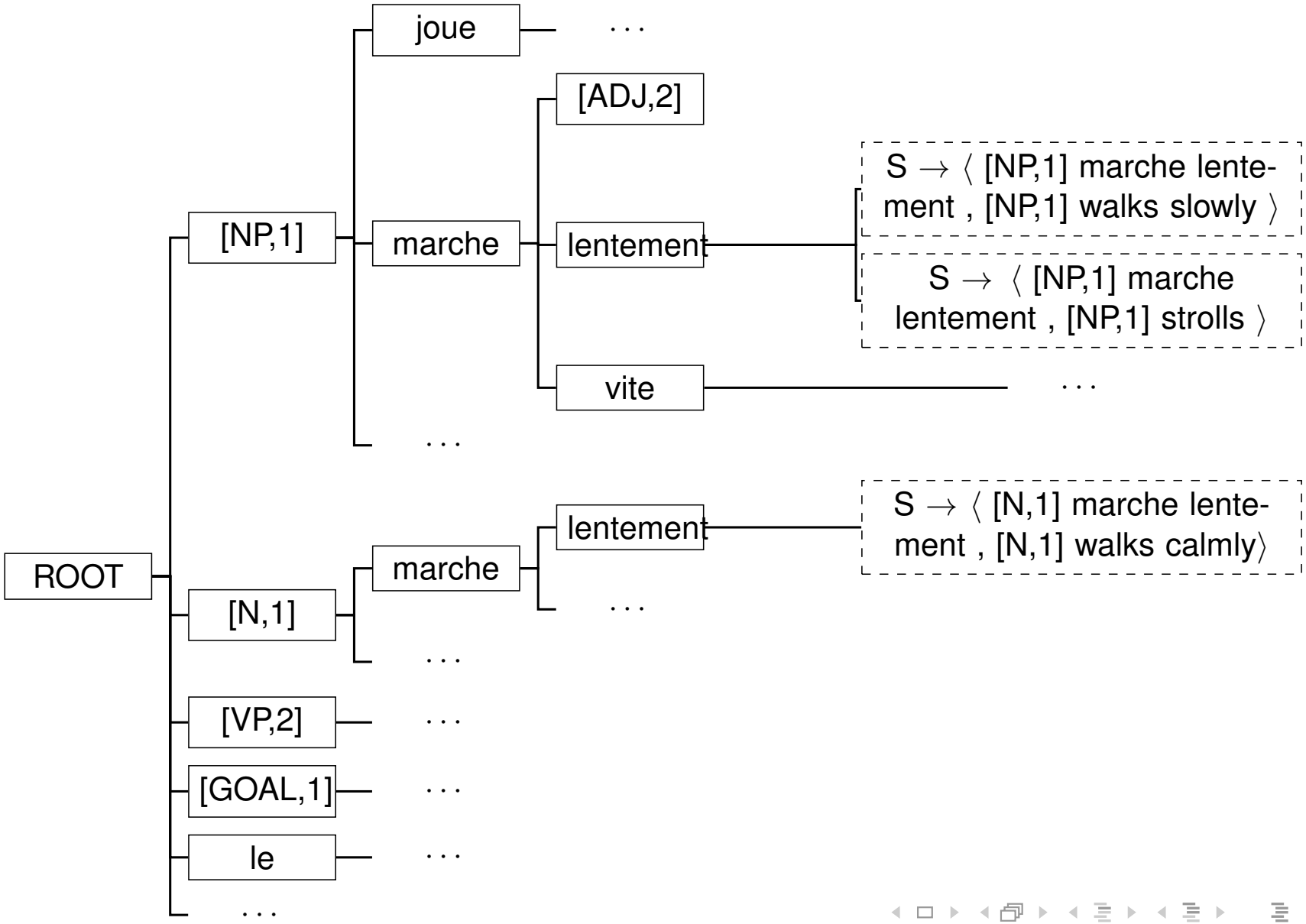
What are tries?

- Tries are also known as Prefix-trees
- Tries efficiently store $\langle$ key-list, value$\rangle$ pairs, when there are many common prefixes amongst the key-lists
- This is exactly the case of Hiero grammars
- While a single flat hashtables could also be used, that would typically require more memory

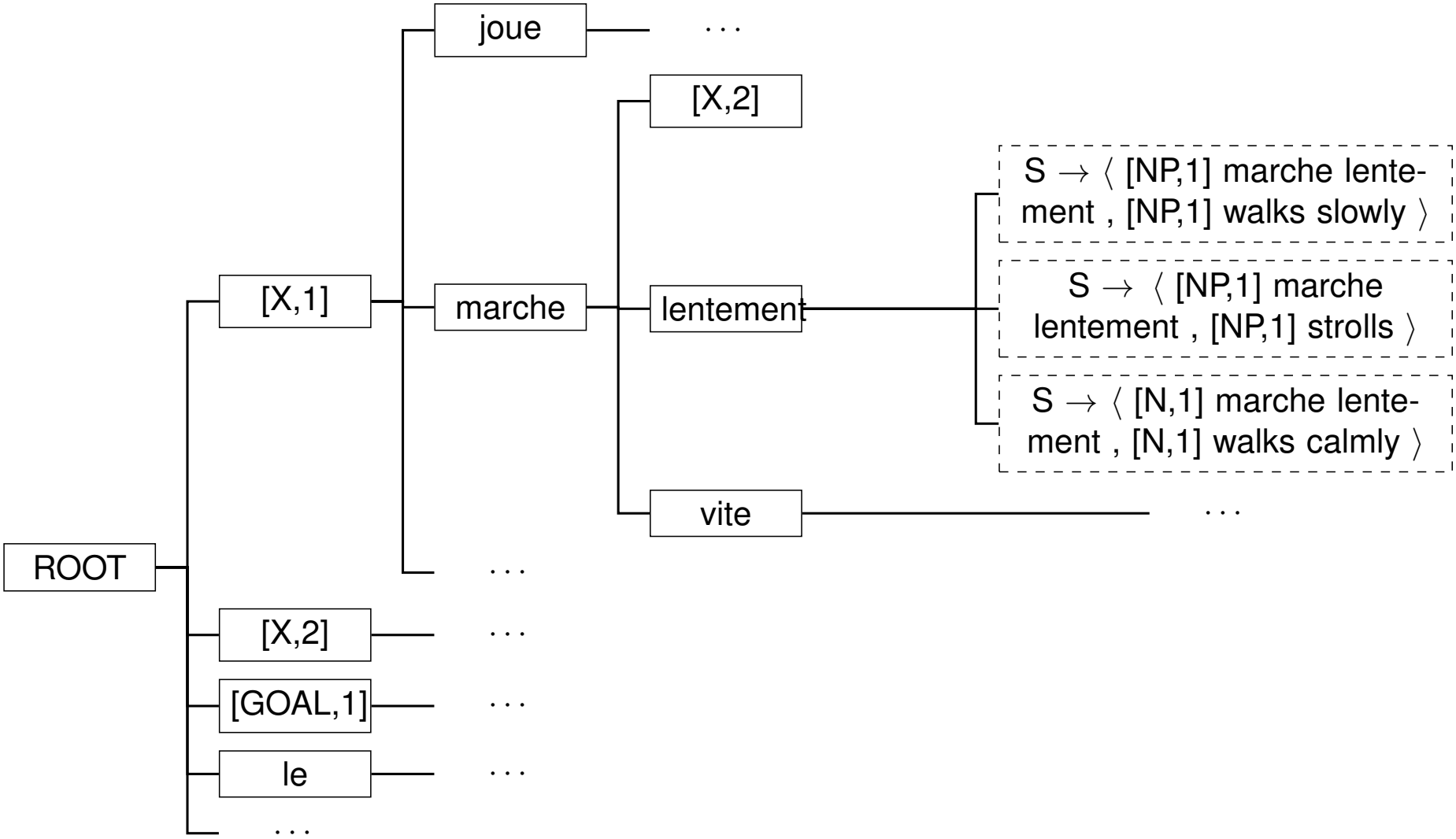
Unlabeled Rule Trie



Strict-Matching Rule Trie



Soft-Matching Rule Trie



Observations Trie for soft-matching

- Labels are removed in the internal nodes
- But the GOAL label is maintained
- Labels are kept at the final rule-lists in the leaf nodes
- **Crucial part**: retain labels only in the final rule lists, not in the index structure itself !

Recap and outlook

Introduced so far:

- Hiero
- Hiero labeling schemes
- Motivation soft-matching
- Implementation soft-matching

Now:

- Experiments strict-matching against soft-matching for the discussed labels

Experimental Setup

- Chinese-English translation experiments
- Data properties

dataset type	size	data origin
train	7.34M	*MultiUn + Hong Kong Parallel Text*
dev	2K	*Multiple Translation Chinese*
test	2K	*Multiple Translation Chinese*

  - Max sentence length 40
- Language model
  - 4-gram language model
  - Kneser-Ney discounting

Results for labeled systems with strict or soft label matching

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
Hiero (H)	31.63	30.56	13.15	**59.28**	58.03	97.15	3.34
Hiero-0^{th}-Str	31.90 ▲*H*	30.79 ▲*H*	13.45	60.11 ▼*H*	59.68 ▲*H*	98.65 ■*H*	2.87
Hiero -0^{th}	32.03▲*H*	30.70▲*H*▼*S*	13.42▲*H*	59.58▼*H*▲*S*	58.87▲*H*▼*S*	97.87■*H*■*S*	8.99
Hiero-1^{st}-Str	31.77	30.62	13.20	60.13 ▼*H*	59.89 ▲*H*	98.47 ■*H*	4.63
Hiero -1^{st}	32.35▲*H*▲*S*	30.98▲*H*▲*S*	13.75▲*H*▲*S*	60.26▼*H*	60.01▲*H*	99.11■*H*▲*S*	8.45
SAMT-Str	31.87 △*H*	30.61	13.38	59.97 ▼*H*	59.94 ▲*H*	98.46 ■*H*	25.59
SAMT	32.40▲*H*▲*S*	31.20▲*H*▲*S*	14.01▲*H*▲*S*	60.19▼*H*▽*S*	60.38▲*H*△*S*	99.37■*H*■*H*	8.09
BoundaryTag-Str	32.26▲*H*	30.94▲*H*	13.91▲*H*	60.20▼*H*	58.78▲*H*	**98.98**▲*H*	29.29
BoundaryTag	**32.77**▲*H*▲*S*	**31.27**▲*H*▲*S*	**14.17**▲*H*▲*S*	60.15▼*H*	**60.83**▲*H*▲*S*	99.72■*H*■*S*	8.60

Part 4:
Beyond basic Soft-matching: extended search, shuffling and double labels

Extending the search

1. Increase the pop-limit (amount of brute-force search)
2. *Explore all rule labelings* : during cube pruning initialization we allow all matching distinctly labeled versions of the same rule source side to be evaluated
3. Combine both (1) and (2)

Next: important detail - shuffling

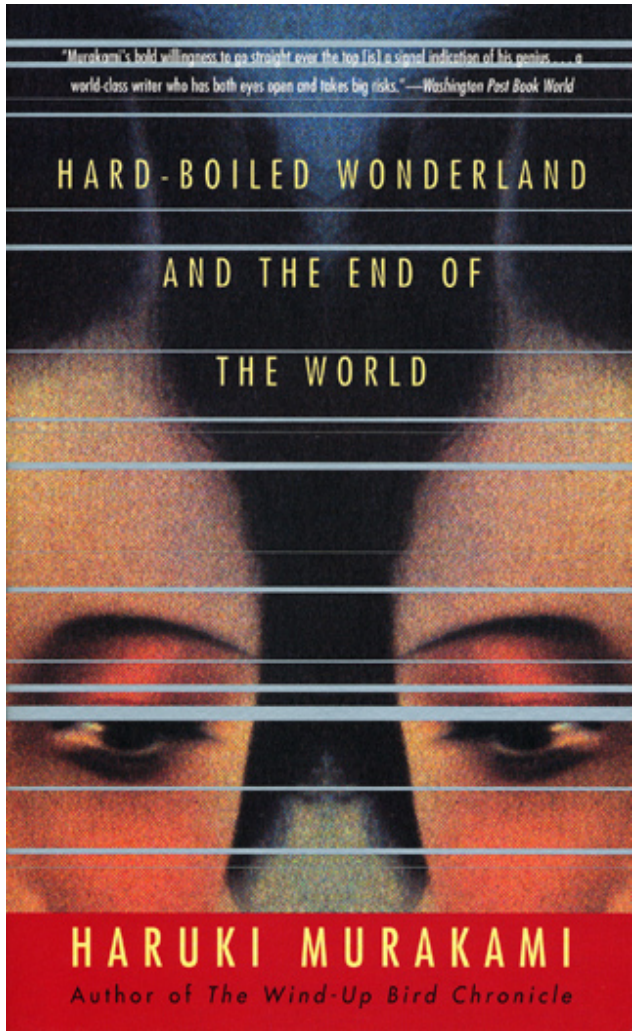
Shuffling: the origin

"Shuffling, son. I'm talking shuffling. I want you to launder and shuffle. That's why I called on you. If it was simple brainwash laundry, there wouldn't have been any need t'call you."

"I don't get it," I said, recrossing my legs."How do you know about shuffling? That's classified information. No outsider's supposed to know about it."

"Well, I do. I've got a pretty open pipeline to the top of the system."

"Okay, then run this thorough your pipeline. Shuffling procedures are completely frozen at this time. Don't ask me why. Obviously some kind of trouble. Whatever the case, shuffling is now prohibited."



Shuffling in this work

Shuffling is the **randomization** of the order of the rules **before** they are evaluated and sorted by score in the cube-pruning queue

The effect of shuffling, an example

- Translating the source phrase “Elle marche lentement”
  - Evaluation order and final order **without** shuffling.

Evaluation order without shuffling:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly (A)
2. S -> [NP,1] marche lentement, [NP,1] strolls (B)
3. S -> [N,1] marche lentement, [N,1] walks calmly (C)
4. **S -> Elle marche [ADV,1], She walks [ADV,1] (D)**

Final order after scoring:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly |||
   Elle marche lentement / She walks slowly ||| -4.0 (A)
2. S -> [NP,1] marche lentement, [NP,1] strolls |||
   Elle marche lentement / She strolls ||| -5.0 (B)
3. S -> [N,1] marche lentement, [N,1] walks calmly |||
   Elle marche lentement / She walks calmly ||| -5.0 (C)
4. **S -> Elle marche [ADV,1], She walks [ADV,1]**
   Elle marche lentement / She walks leisurely ||| -5.0 (D)

The effect of shuffling, an example

- Translating the source phrase “Elle marche lentement”
  - Evaluation order and final order **with** shuffling.

Evaluation order with shuffling (i.e. random):

1. **S -> Elle marche [ADV,1], She walks [ADV,1]** (D)
2. S -> [NP,1] marche lentement, [NP,1] walks slowly (A)
3. S -> [N,1] marche lentement, [N,1] walks calmly (C)
4. S -> [NP,1] marche lentement, [NP,1] strolls (B)

Final order after scoring:

1. S -> [NP,1] marche lentement, [NP,1] walks slowly |||
   Elle marche lentement / She walks slowly ||| -4.0 (A)

2. **S -> Elle marche [ADV,1], She walks [ADV,1]**
   Elle marche lentement / She walks leisurely ||| -5.0 (D)

3. S -> [N,1] marche lentement, [N,1] walks calmly |||
   Elle marche lentement / She walks calmly ||| -5.0 (C)

4. S -> [NP,1] marche lentement, [NP,1] strolls |||
   Elle marche lentement / She strolls ||| -5.0 (B)

Shuffling: observations

- Shuffling only affects relative order rules that tie for same score
- Avoid “lazy tuning” : shuffling can encourage more robust tuning, by eliminating repeated order for hypotheses with same score

Results extended search experiments

System Name	Pop-limit	Explore all rule labelings	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
Hiero	1000	——-	31.63	30.56	13.15	59.28	58.03	97.15	3.34
Hiero 0^{th} (B_0)	1000	NO	32.03▲H	30.70▲H	13.42▲H	59.58▼H	58.87▲H	97.87■H	8.99
Hiero 0^{th}	2000	NO	32.24▲H▲B_0	30.66▲H	13.41	**59.01**▲H▲B_0	58.59△H	97.42■H■B_0	16.13
Hiero 0^{th}	4000	NO	32.30▲H▲B_0	30.85▲H▲B_0	13.71▲H▲B_0	59.56▼H	59.23▲H△B_0	98.19■H■B_0	28.27
Hiero 0^{th}	1000	YES	32.18▲H	30.77▲H△B_0	13.55▲H△B_0	59.23 ▲B_0	58.74▲H	97.69■H■B_0	9.32
Hiero 0^{th}-Sh	1000	YES	32.25▲H▲B_0	30.80▲H▲B_0	13.60▲H▲B_0	59.44▽H	59.12▲H	97.99■H	10.14
Hiero 0^{th}-Sh	2000	YES	32.47▲H▲B_0	30.87▲H▲B_0	13.69▲H▲B_0	59.51▼H	59.27▲H△B_0	98.27■H■B_0	16.59
Hiero 0^{th}-Sh	4000	YES	32.26▲H▲B_0	30.77▲H△B_0	13.55▲H△B_0	59.12△H▲B_0	58.72▲H	97.55■H■B_0	37.75
Hiero 1^{st} (B_1)	1000	NO	32.35▲H	30.98▲H	13.75▲H	60.26▼H	**60.01**▲H	99.11■H	8.45
Hiero 1^{st}	2000	NO	32.40▲H	31.00▲H	13.74▲H	60.36▼H	59.93▲H	99.15■H	15.40
Hiero 1^{st}	4000	NO	32.49▲H	30.95▲H	13.75▲H	60.29▼H	60.00▲H	99.15■H	28.84
Hiero 1^{st}	1000	YES	32.50▲H△B_1	31.03▲H△B_1	13.80▲H	60.14▼H	59.77▲H	99.05■H	9.55
Hiero 1^{st}-Sh	1000	YES	32.66▲H▲B_1	**31.05**▲H▲B_1	**13.88**▲H	60.01▼H▲B_1	59.99▲H	99.14■H	10.16
Hiero 1^{st}-Sh	2000	YES	32.62▲H▲B_1	30.90▲H▼B_1	13.70▲H	59.72▼H▲B_1	59.41▲H▼B_1	98.60■H■B_1	21.79
Hiero 1^{st}-Sh	4000	YES	**32.70**▲H▲B_1	30.99▲H	13.83▲H	60.22▼H	59.91▲H	**99.20**■H	40.81

Some observations:

- Exploration all rule labelings more effective than increasing pop-limit
- Both combined give best results
- Shuffling helps

Combining labels

- Fast soft-matching implementation enables combining labels
- We simply concatenate two labels
- For the label-substitution features, we add features for each part separately, to reduce sparsity and hence overfitting

Combining labels: results

System Name	BLEU ↑	METEOR ↑	BEER ↑	TER ↓	KRS ↑	Length	CPU time
Hiero	31.63	30.56	13.15	**59.28**	58.03	97.15	3.34
Hiero 0^{th} + *Hiero* 1^{st}	32.30▲H▲H_0	30.95▲H▲H_0	13.75▲H▲H_0	60.16▼H▼H_0	60.13▲H▲H_0	99.07▲H■H_0	7.18
SAMT+ *Hiero* 1^{st}	32.57▲H△H_1	31.07▲H▼S▲H_1	13.84▲H▽S	59.94▼H▲H_1	60.18▲H	99.13■H■S	11.63
Bnd.Tags+ *Hiero* 1^{st}	**32.65**▲H▲H_1	**31.36**▲H▲B▲H_1	**14.16**▲H▲H_1	60.21▼H	**61.46**▲H▲B▲H_1	**99.86**■H■H_1	10.32

- *BoundaryTags+Hiero* 1^{st} is the overall best system for METEOR and KRS; nearly best for BLEU and BEER
  - Adding Hiero 1^{st} label in addition to BoundaryTags further improves the word-order

Summary

- Discussed Hiero + labels, and motivated and described soft-matching
- Explained how to make soft-matching fast, and why this is important
- Ecplored several extensions to boost the effectiveness of soft-matching:
  1. Extending the search space to ensure exploration matching substitutions
  2. Shuffling as a way to reduce issues with extending the search space, and to reduce overfitting
  3. Experiments with double labels

Conclusions

- Efficient soft-matching is important for practical usefullness of the method but non-trivial. We provided the details of a fast implementation.
- Soft matching is typically superior to strict matching
- Extension of the search that diversifies the explored labels can boost performance
- Double labels can work, and further boost performance, despite risks of overfitting etc

Questions?