Decoding

Philipp Koehn

19 February 2015

Decoding

- We have a mathematical model for translation

$$p(\mathbf{e}|\mathbf{f})$$

- Task of decoding: find the translation $\mathbf{e}_{\text{best}}$ with highest probability

$$\mathbf{e}_{\text{best}} = \text{argmax}_{\mathbf{e}} \; p(\mathbf{e}|\mathbf{f})$$

- Two types of error

  - the most probable translation is bad → fix the model
  - search does not find the most probably translation → fix the search

- Decoding is evaluated by search error, not quality of translations (although these are often correlated)

translation process

Translation Process

- Task: translate this sentence from German into English

er   **geht**   **ja**   **nicht**   **nach**   **hause**
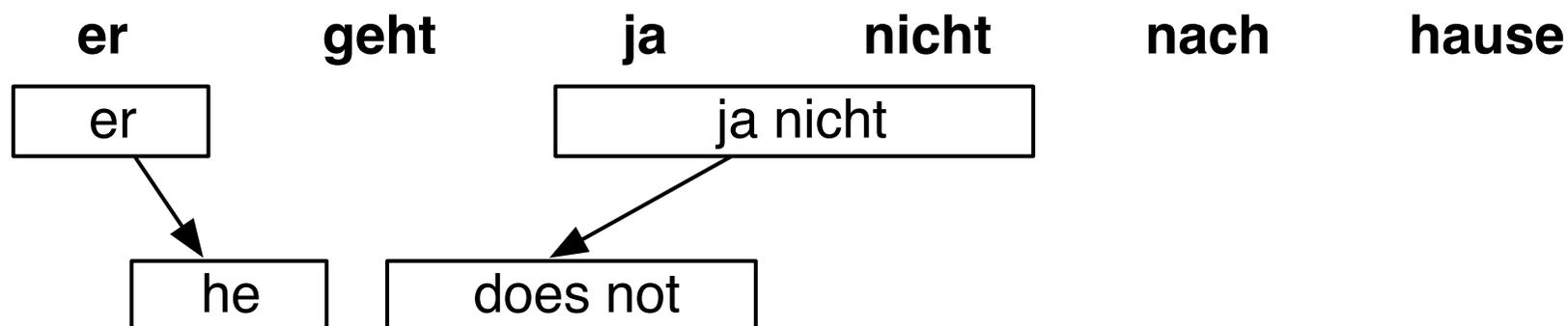
Translation Process

- Task: translate this sentence from German into English

er   **geht**   **ja**   **nicht**   **nach**   **hause**

er

he

- Pick phrase in input, translate

- Task: translate this sentence from German into English

er   **geht**   **ja**   **nicht**   **nach**   **hause**

| er |

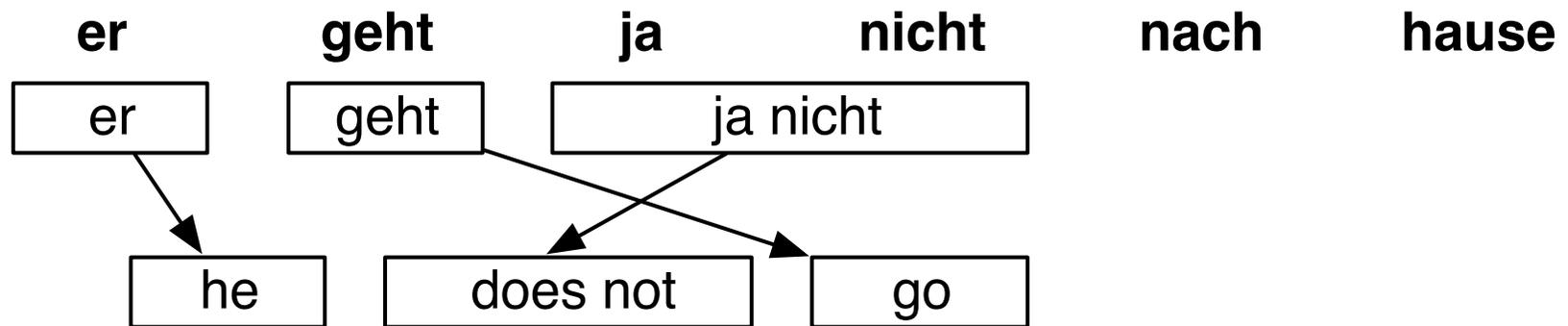| ja nicht |

| he | | does not |

- Pick phrase in input, translate

  – it is allowed to pick words out of sequence reordering
  – phrases may have multiple words: many-to-many translation
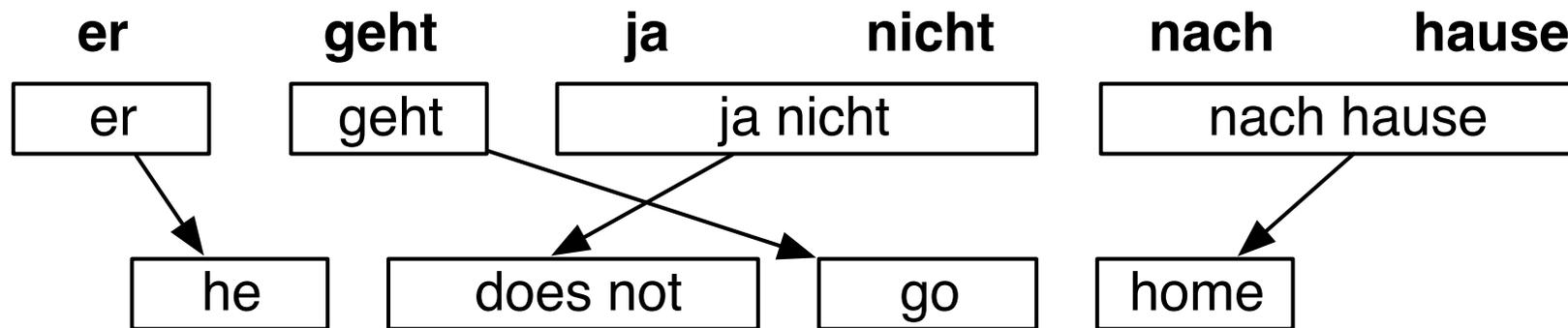
- Task: translate this sentence from German into English

er   **geht**   **ja**   **nicht**   **nach**   **hause**

er		geht		ja nicht

he		does not		go

- Pick phrase in input, translate

- Task: translate this sentence from German into English

- Pick phrase in input, translate

- Probabilistic model for phrase-based translation:

$$\mathbf{e}_{\text{best}} = \text{argmax}_{\mathbf{e}} \prod_{i=1}^{I} \phi(\bar{f}_i | \bar{e}_i)\, d(start_i - end_{i-1} - 1)\, p_{\text{LM}}(\mathbf{e})$$

- Score is computed incrementally for each partial hypothesis

- Components

Phrase translation Picking phrase $\bar{f}_i$ to be translated as a phrase $\bar{e}_i$
  $\rightarrow$ look up score $\phi(\bar{f}_i | \bar{e}_i)$ from phrase translation table
Reordering Previous phrase ended in end_{i-1}, current phrase starts at $start_i$
  $\rightarrow$ compute $d(start_i - end_{i-1} - 1)$
Language model For n-gram model, need to keep track of last $n-1$ words
  $\rightarrow$ compute score $p_{\text{LM}}(w_i | w_{i-(n-1)}, ..., w_{i-1})$ for added words w_i

decoding process

Translation Options

er	geht	ja	nicht	nach	hause
he	is	yes	not	after	house
it	are	is	do not	to	home
, it	goes	, of course	does not	according to	chamber
, he	go	,	is not	in	at home

it is		not	home
he will be		is not	under house
it goes		does not	return home
he goes		do not	do not

is	to
are	following
is after all	not after
does	not to

- not
- is not
- are not
- is not a

- Many translation options to choose from

  – in Europarl phrase table: 2727 matching phrase pairs for this sentence
  – by pruning to the top 20 per phrase, 202 translation options remain
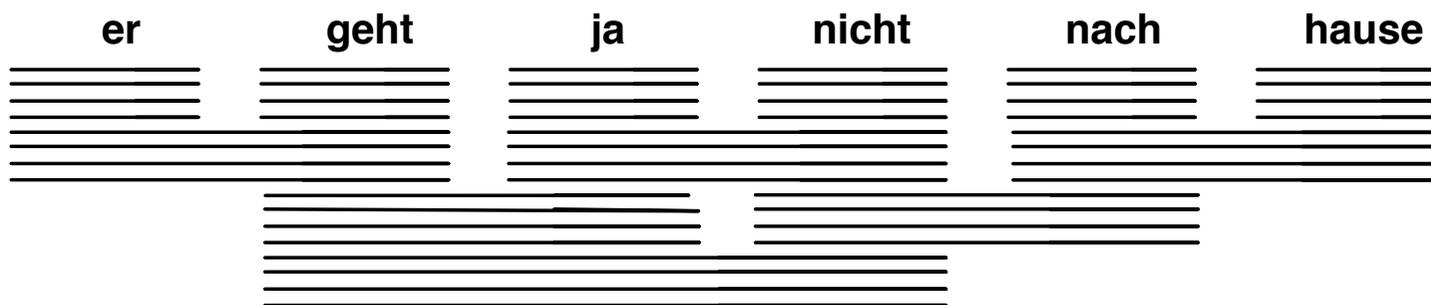
Translation Options

er	geht	ja	nicht	nach	hause
he	is	yes	not	after	house
it	are	is	do not	to	home
, it	goes	, of course	does not	according to	chamber
, he	go		is not	in	at home

it is		not		home
he will be		is not		under house
it goes		does not		return home
he goes		do not		do not

is		to
are		following
is after all		not after
does		not to

| not |
| is not |
| are not |
| is not a |

- The machine translation decoder does not know the right answer
  - picking the right translation options
  - arranging them in the right order

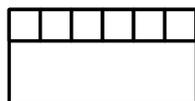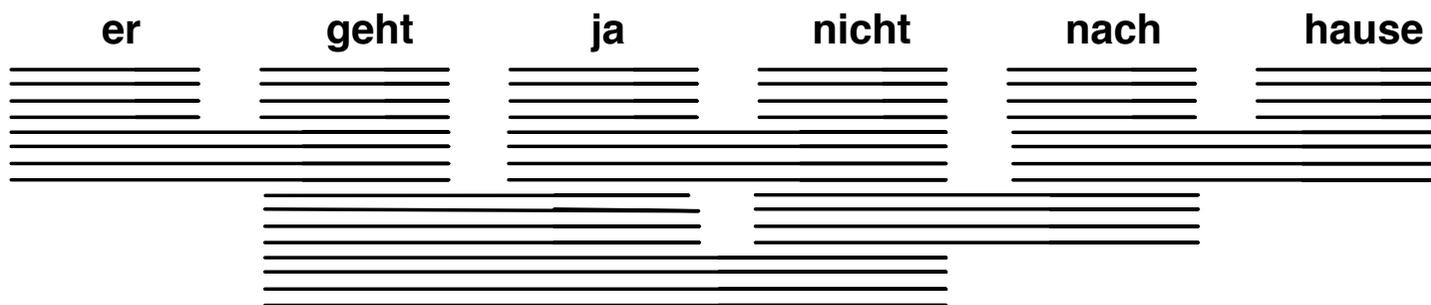→ Search problem solved by heuristic beam search

er   **geht**   **ja**   **nicht**   **nach**   **hause**
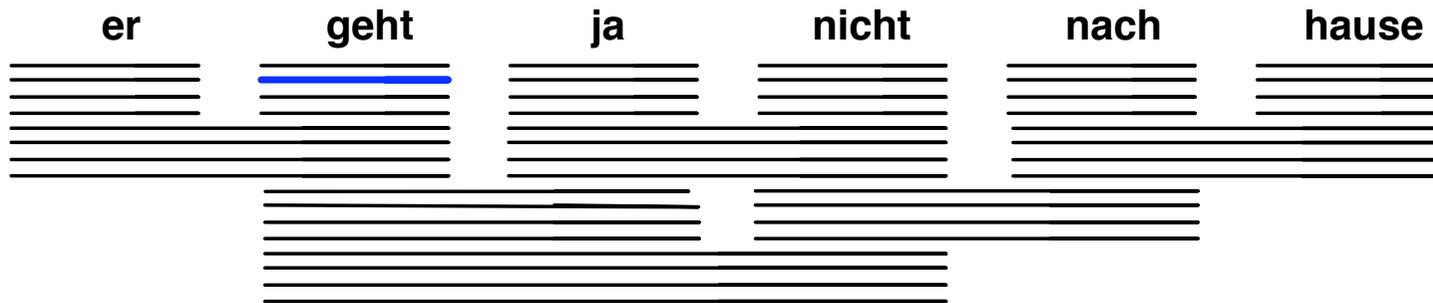
consult phrase translation table for all input phrases

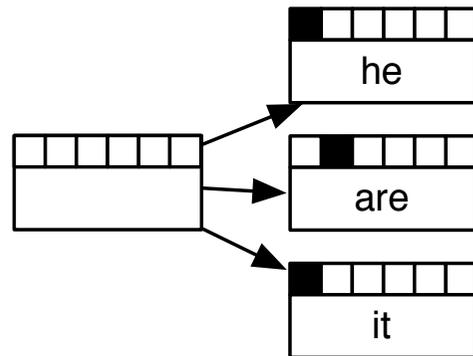er   geht   ja   nicht   nach   hause

initial hypothesis: no input words covered, no output produced

Decoding: Hypothesis Expansion

er   geht   ja   nicht   nach   hause

are
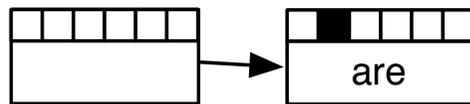
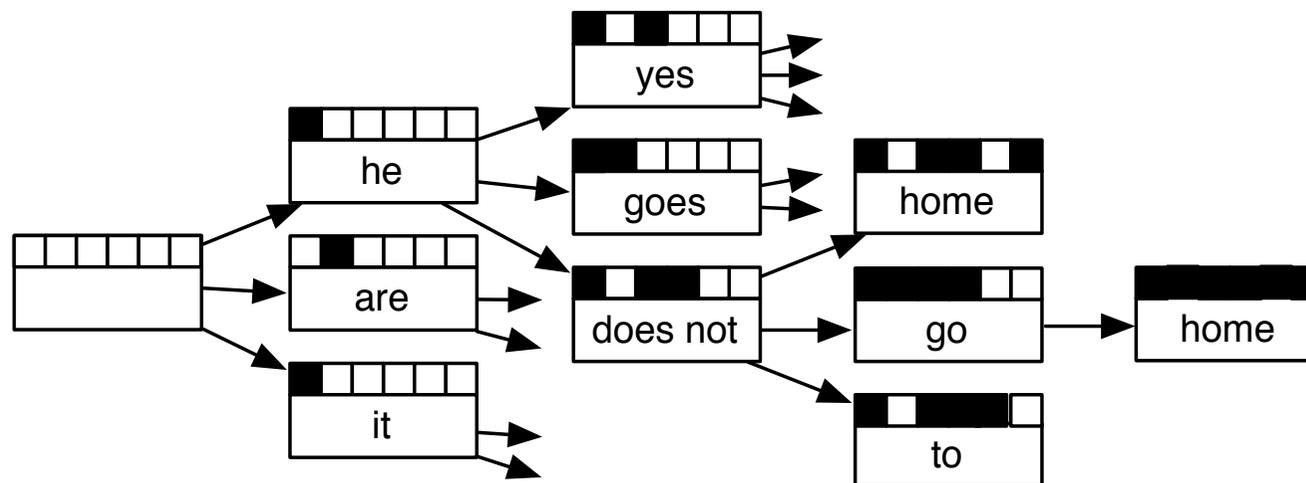pick any translation option, create new hypothesis

Decoding: Hypothesis Expansion

er geht ja nicht nach hause

he

are

it
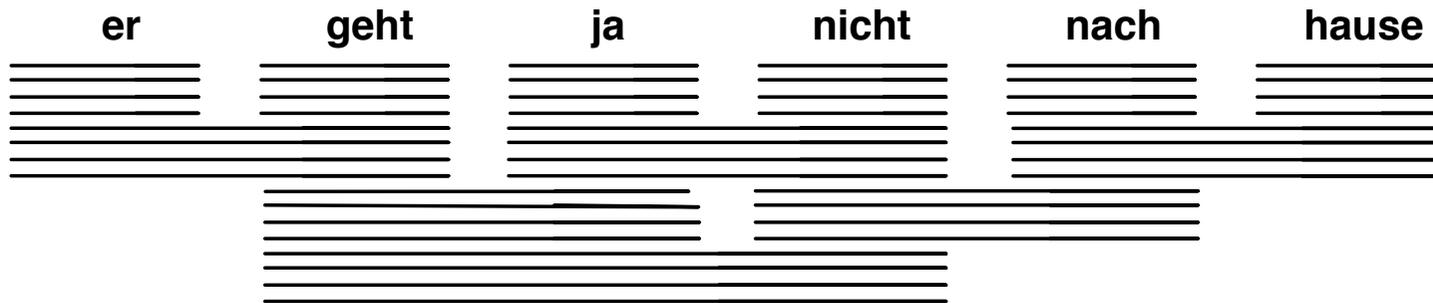
create hypotheses for all other translation options

er   geht   ja   nicht   nach   hause
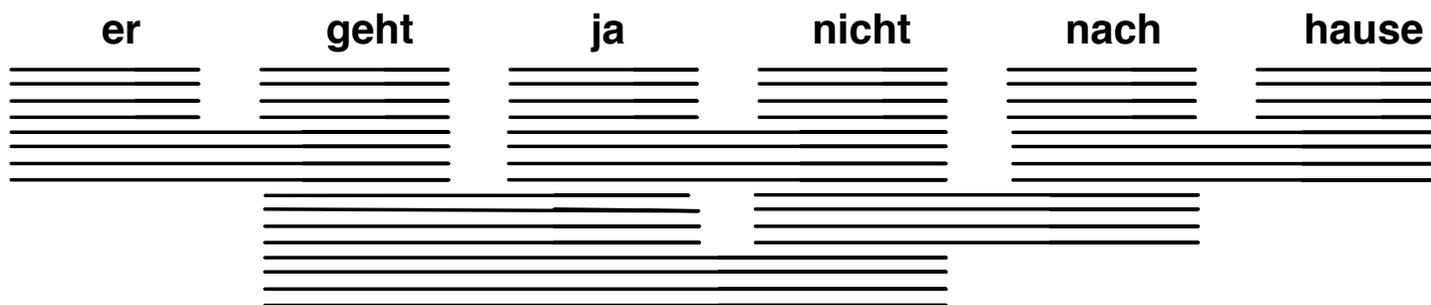
also create hypotheses from created partial hypothesis

Decoding: Find Best Path

er   geht   ja   nicht   nach   hause
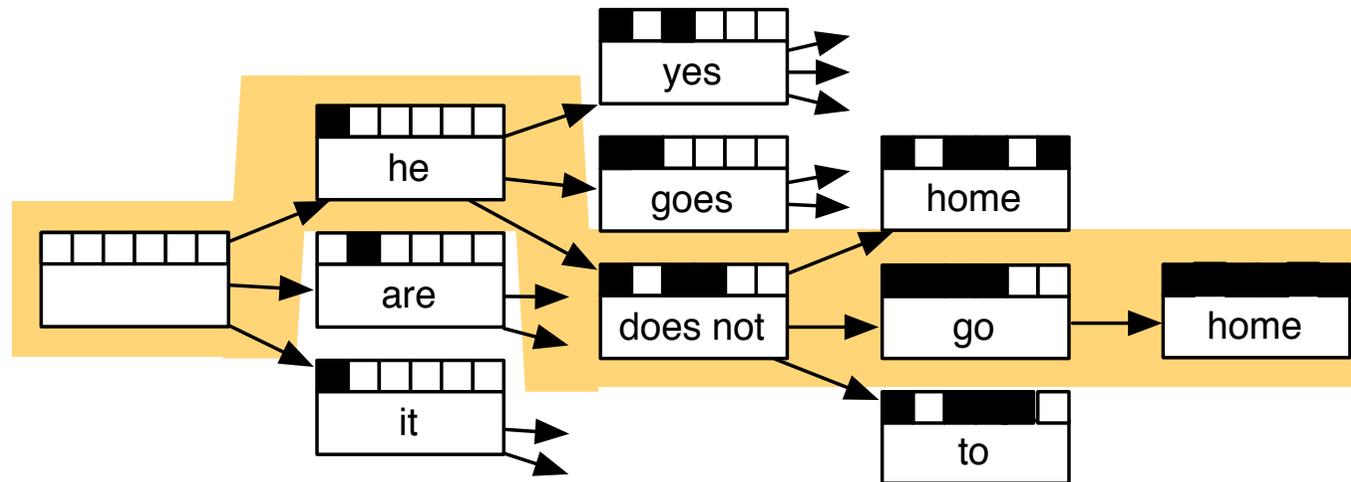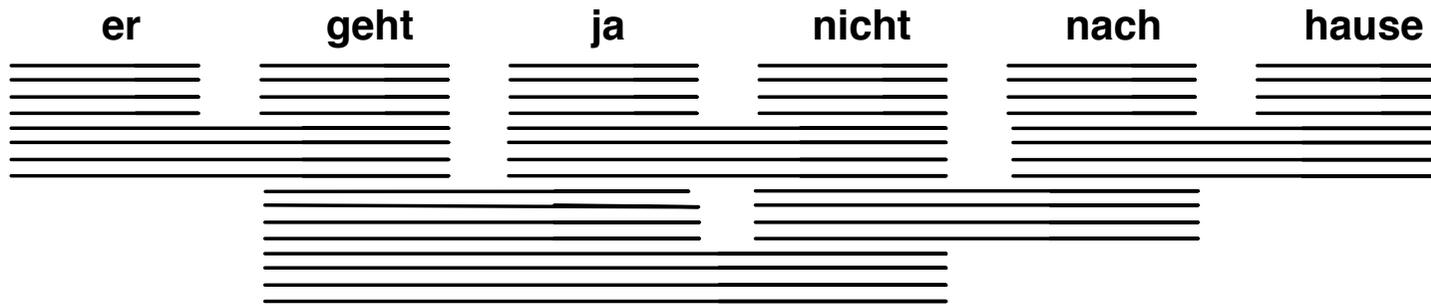
backtrack from highest scoring complete hypothesis

dynamic programming

- The suggested process creates exponential number of hypothesis

- Machine translation decoding is NP-complete

- Reduction of search space:

  - recombination (risk-free)
  - pruning (risky)

Recombination

- Two hypothesis paths lead to two matching hypotheses

  - same number of foreign words translated
  - same English words in the output
  - different scores
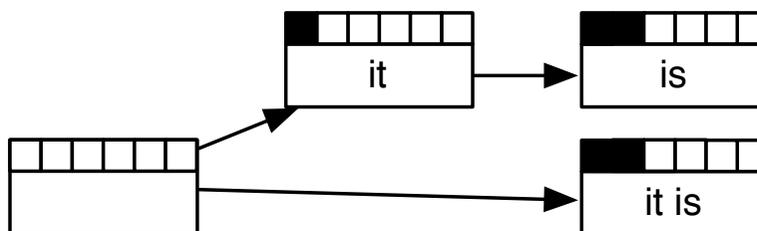
- Worse hypothesis is dropped
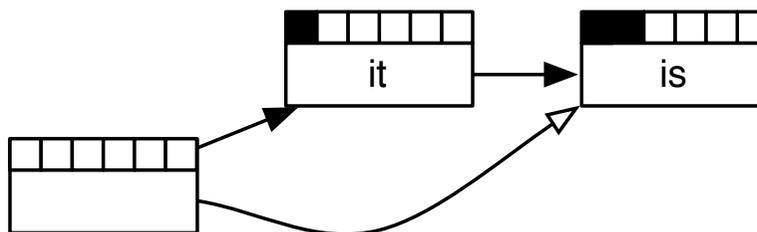
Recombination

- Two hypothesis paths lead to hypotheses indistinguishable in subsequent search

  – same number of foreign words translated
  – same last two English words in output (assuming trigram language model)
  – same last foreign word translated
  – different scores
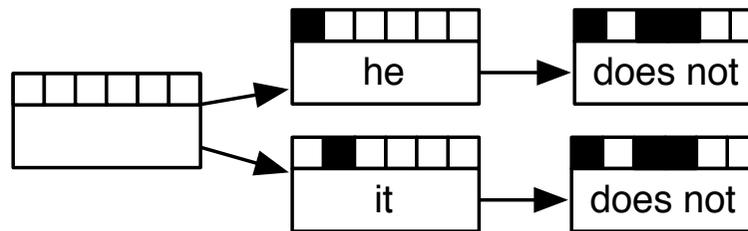
- Worse hypothesis is dropped
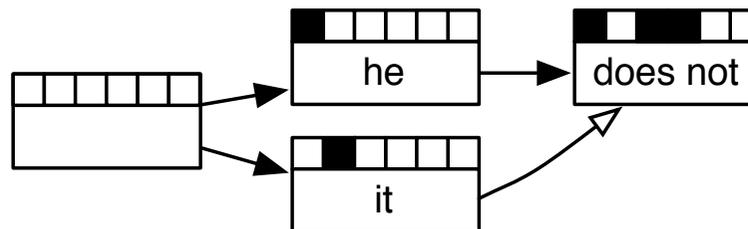
Restrictions on Recombination

- **Translation model:** Phrase translation independent from each other

  $\rightarrow$ no restriction to hypothesis recombination

- **Language model:** Last $n - 1$ words used as history in n-gram language model

  $\rightarrow$ recombined hypotheses must match in their last $n - 1$ words

- **Reordering model:** Distance-based reordering model based on distance to end position of previous input phrase

  $\rightarrow$ recombined hypotheses must have that same end position

- Other feature function may introduce additional restrictions

pruning

Pruning

- Recombination reduces search space, but not enough

  (we still have a NP complete problem on our hands)

- Pruning: remove bad hypotheses early

  – put comparable hypothesis into stacks
    (hypotheses that have translated same number of input words)
  – limit number of hypotheses in each stack

no word translated   one word translated   two words translated   three words translated
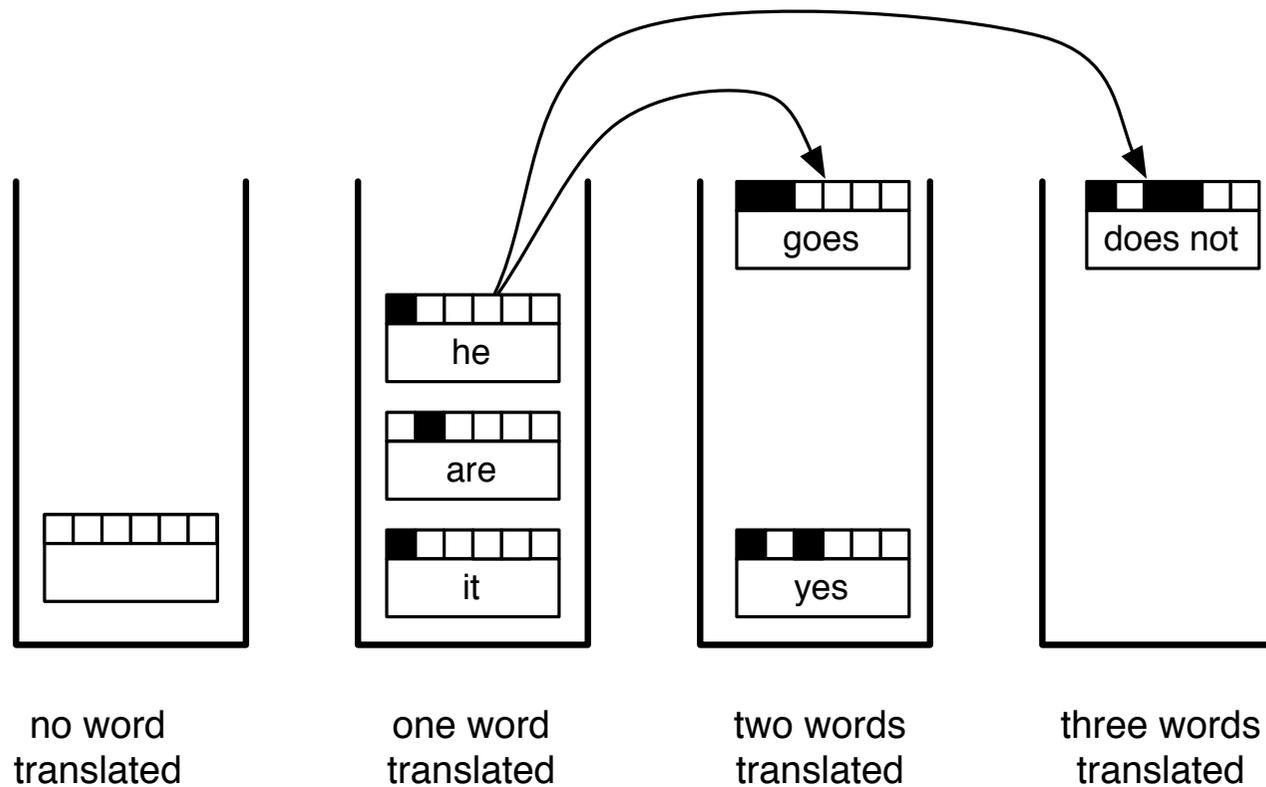
- Hypothesis expansion in a stack decoder
  - translation option is applied to hypothesis
  - new hypothesis is dropped into a stack further down

Stack Decoding Algorithm

1: place empty hypothesis into stack 0
2: **for all** stacks $0...n-1$ **do**
3:     **for all** hypotheses in stack **do**
4:         **for all** translation options **do**
5:             **if** applicable **then**
6:                 create new hypothesis
7:                 place in stack
8:                 recombine with existing hypothesis **if** possible
9:                 prune stack **if** too big
10:             **end if**
11:         **end for**
12:     **end for**
13: **end for**

Pruning

- Pruning strategies

  - histogram pruning: keep at most k hypotheses in each stack
  - stack pruning: keep hypothesis with score $\alpha \times$ best score ($\alpha < 1$)

- Computational time complexity of decoding with histogram pruning

$$O(\text{max stack size} \times \text{translation options} \times \text{sentence length})$$

- Number of translation options is linear with sentence length, hence:

$$O(\text{max stack size} \times \text{sentence length}^2)$$

- Quadratic complexity

Reordering Limits

- Limiting reordering to maximum reordering distance

- Typical reordering distance 5–8 words

  - depending on language pair
  - larger reordering limit hurts translation quality

- Reduces complexity to linear

$$O(\text{max stack size} \times \text{sentence length})$$

- Speed / quality trade-off by setting maximum stack size

future cost estimation

Translating the Easy Part First?

the tourism initiative addresses this for **the first time**
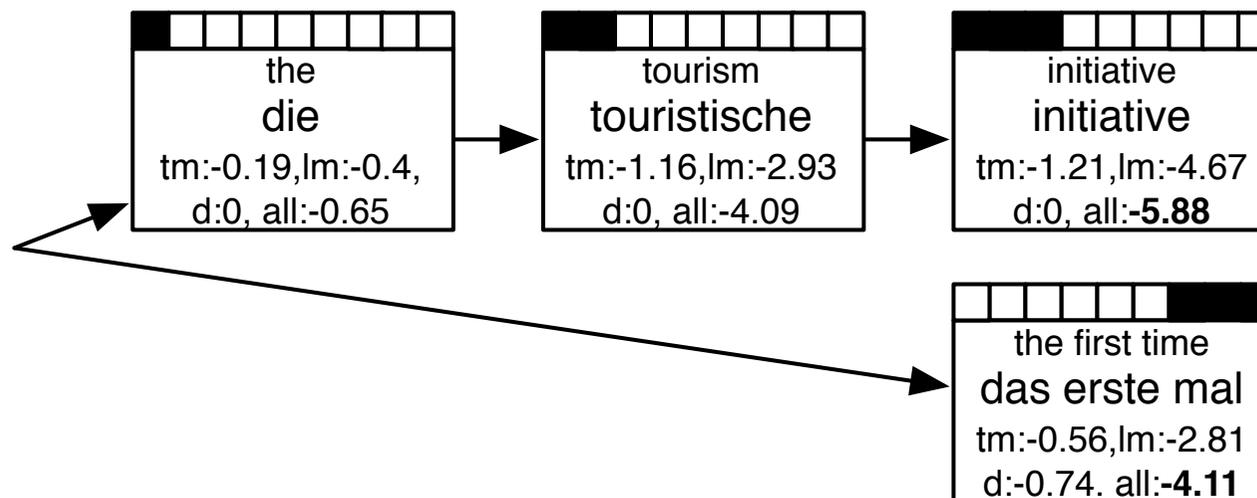
both hypotheses translate 3 words
worse hypothesis has better score

- Future cost estimate: how expensive is translation of rest of sentence?

- Optimistic: choose cheapest translation options

- Cost for each translation option

  - **translation model**: cost known

  - **language model:** output words known, but not context
    $\rightarrow$ estimate without context

  - **reordering model:** unknown, ignored for future cost estimation

the   tourism initiative addresses this   for   the   first   time

-1.0	-2.0	-1.5	-2.4	-1.4	-1.0	-1.0	-1.9	-1.6

-4.0	-2.5	-2.2

-1.3   -2.4

-2.7

-2.3

-2.3

-2.3
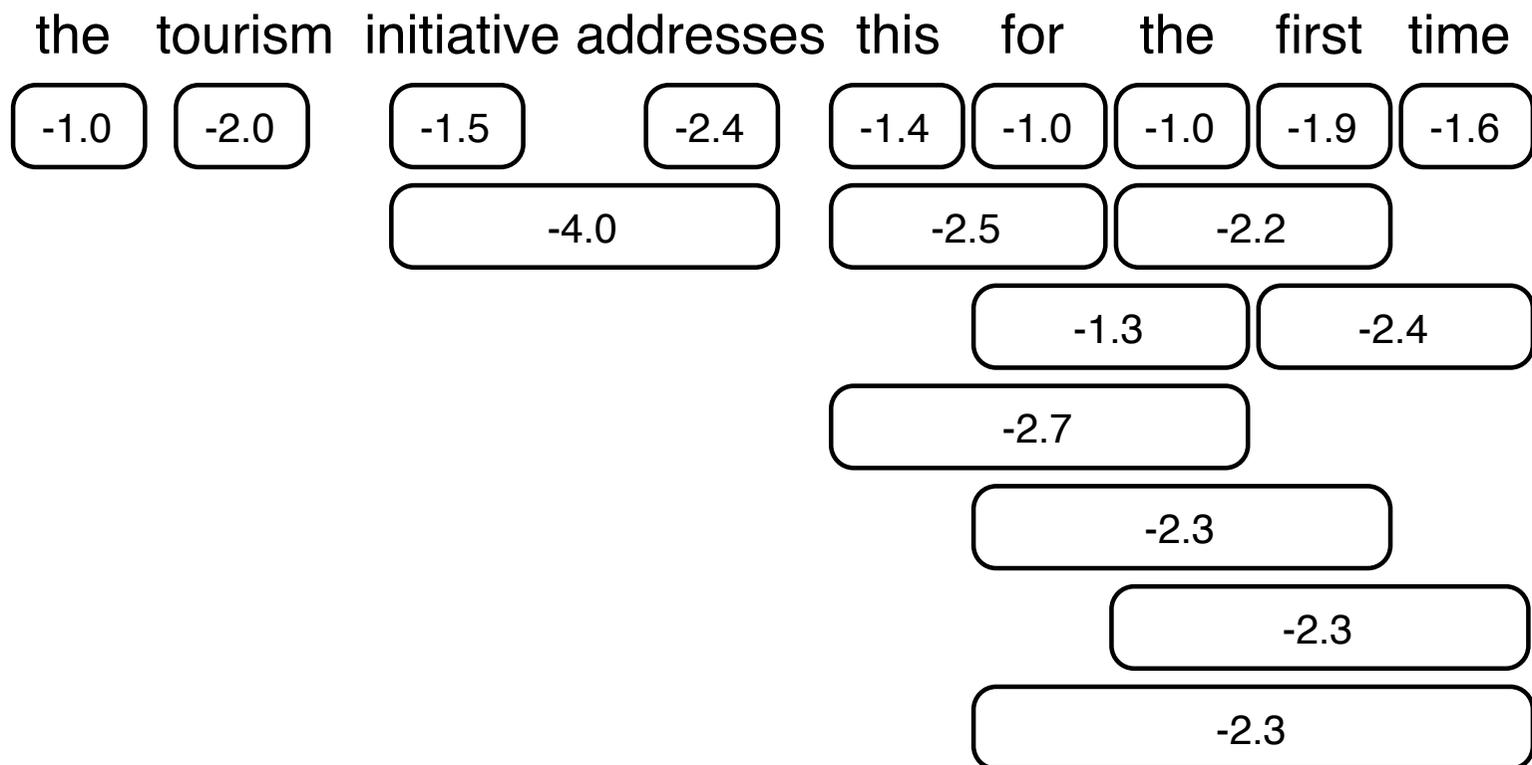
cost of cheapest translation options for each input span (log-probabilities)

- Compute cost estimate for all contiguous spans by combining cheapest options

first word	future cost estimate for n words (from first)								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
the	-1.0	-3.0	-4.5	-6.9	-8.3	-9.3	-9.6	-10.6	-10.6
tourism	-2.0	-3.5	-5.9	-7.3	-8.3	-8.6	-9.6	-9.6	
initiative	-1.5	-3.9	-5.3	-6.3	-6.6	-7.6	-7.6		
addresses	-2.4	-3.8	-4.8	-5.1	-6.1	-6.1			
this	-1.4	-2.4	-2.7	-3.7	-3.7				
for	-1.0	-1.3	-2.3	-2.3					
the	-1.0	-2.2	-2.3						
first	-1.9	-2.4							
time	-1.6								

- Function words cheaper (the: -1.0) than content words (tourism -2.0)
- Common phrases cheaper (for the first time: -2.3)
  than unusual ones (tourism initiative addresses: -5.9)

- Hypothesis score and future cost estimate are combined for pruning

  – left hypothesis starts with hard part: the tourism initiative
  score: -5.88, future cost: -6.1 → total cost -11.98

  – middle hypothesis starts with easiest part: the first time
  score: -4.11, future cost: -9.3 → total cost -13.41

  – right hypothesis picks easy parts: this for ... time
  score: -4.86, future cost: -9.1 → total cost -13.96

cube pruning

- Exhaustive matching of hypotheses to applicable translations options
  $\rightarrow$ too much computation

```
1:  place empty hypothesis into stack 0
2:  for all stacks 0...n − 1 do
3:      for all hypotheses in stack do
4:          for all translation options do
5:              if applicable then
6:                  create new hypothesis
7:                  place in stack
8:                  recombine with existing hypothesis if possible
9:                  prune stack if too big
10:             end if
11:         end for
12:     end for
13: end for
```

Group Hypotheses and Options

- Group hypotheses by coverage vector

  - ■ ■ ■ ☐ ☐ ☐
  - ■ ■ ☐ ■ ☐ ☐
  - ■ ☐ ■ ■ ☐ ☐
  - ...

- Group translation options by span

  - ☐ ☐ ☐ ■ ☐ ☐
  - ☐ ☐ ☐ ☐ ■ ☐
  - ☐ ☐ ☐ ■ ■ ☐
  - ...

⇒ Loop over groups, check for applicability once for each pair of groups (not much gained so far)
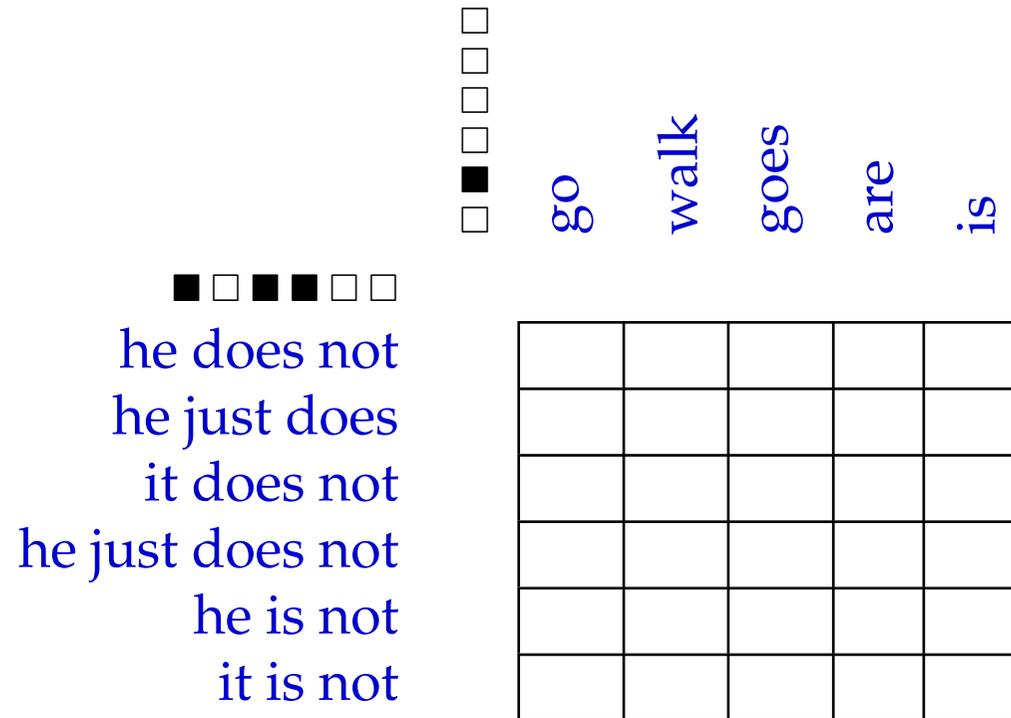
- Example: group with 6 hypotheses, group with 5 translation options

- Should we really create all 6×5 of them?

Rank by Score

□
□
□
□
■
□

-1.1 go   -1.2 walk   -1.4 goes   -1.7 are   -2.1 is

■ □ ■ ■ □ □

he does not -3.2
he just does -3.5
it does not -4.1
he just does not -4.3
he is not -4.7
it is not -5.1
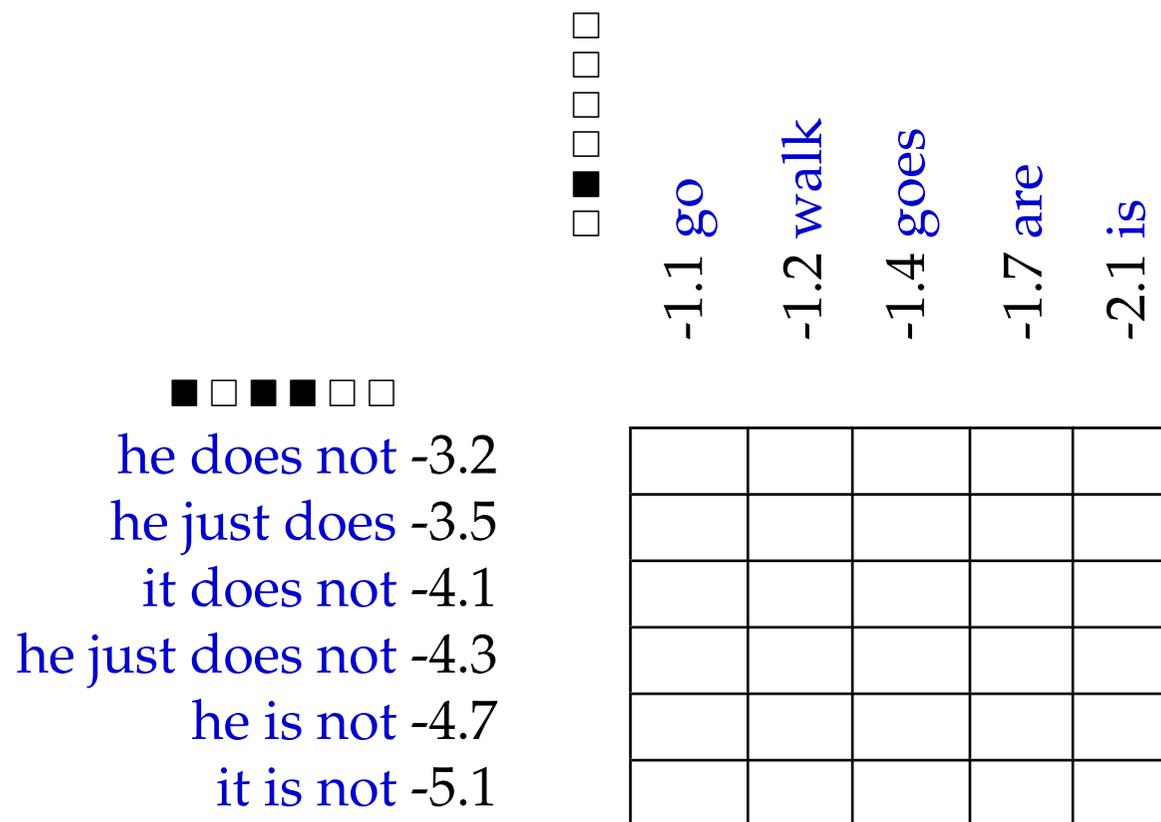
- Rank hypotheses by score so far

- Rank translation options by score estimate

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
■□■■□□					
he does not -3.2	-4.2	-4.4	-4.6	-4.9	-5.3
he just does -3.5	-4.5	-4.7	-4.9	-5.2	-5.6
it does not -4.1	-5.1	-5.3	-5.5	-5.8	-6.2
he just does not -4.3	-5.3	-5.5	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2
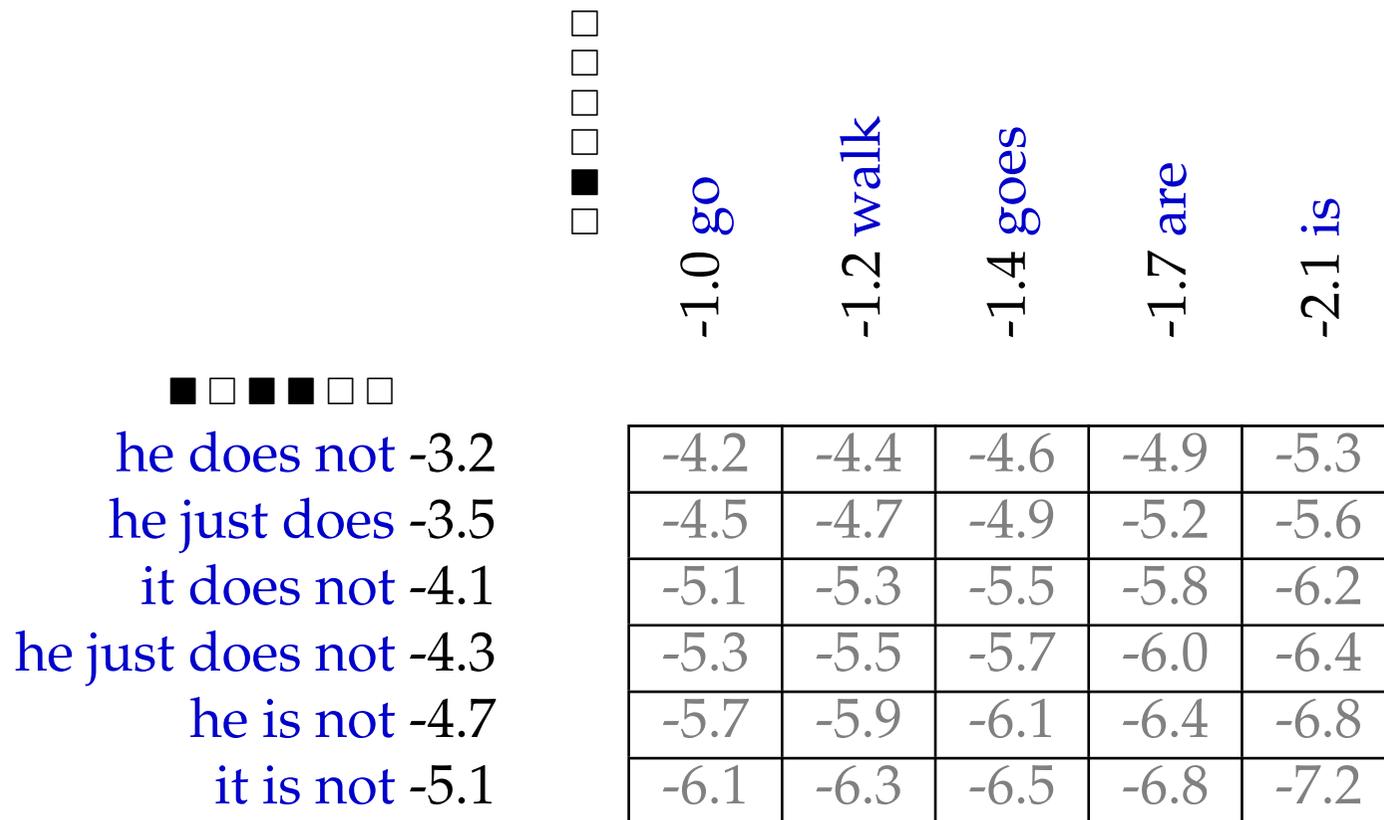
- Expected score: hypothesis score + translation option score

- Real score will be different, since language model score depends on context

Only Compute Half

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
he does not -3.2	**-4.2**	**-4.4**	**-4.6**	**-4.9**	**-5.3**
he just does -3.5	**-4.5**	**-4.7**	**-4.9**	**-5.2**	-5.6
it does not -4.1	**-5.1**	**-5.3**	**-5.5**	-5.8	-6.2
he just does not -4.3	**-5.3**	**-5.5**	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2
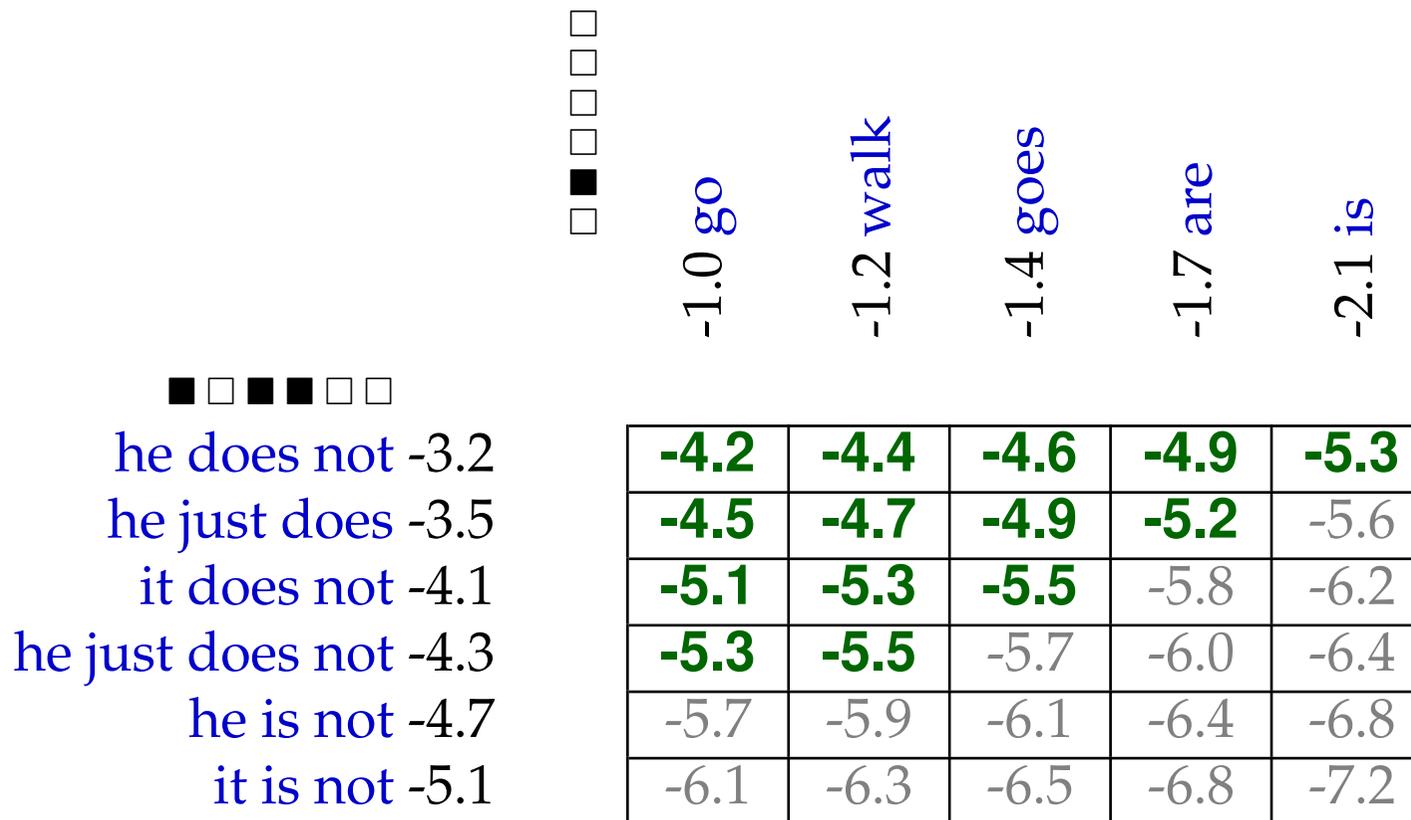
- If we want to save computational cost, we could decide to only compute some

- One way to do this: based on expected score

Cube Pruning

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
he does not -3.2	**-3.9**	-4.4	-4.6	-4.9	-5.3
he just does -3.5	-4.5	-4.7	-4.9	-5.2	-5.6
it does not -4.1	-5.1	-5.3	-5.5	-5.8	-6.2
he just does not -4.3	-5.3	-5.5	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2
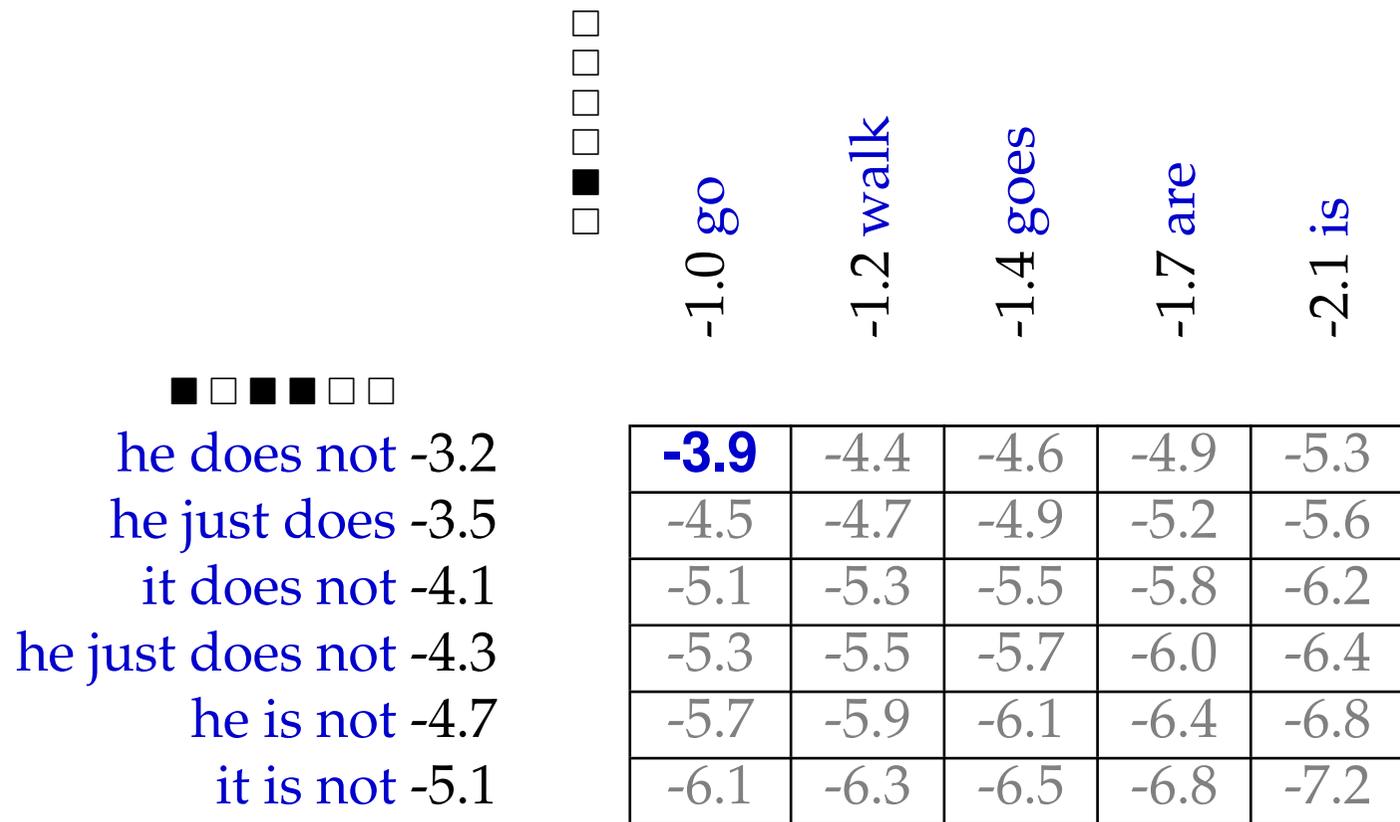
- Start with best hypothesis, best translation option

- Create new hypothesis (actual score becomes available)

Cube Pruning (2)

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
he does not -3.2	**-3.9**	**-4.1**	-4.6	-4.9	-5.3
he just does -3.5	**-4.3**	-4.7	-4.9	-5.2	-5.6
it does not -4.1	-5.1	-5.3	-5.5	-5.8	-6.2
he just does not -4.3	-5.3	-5.5	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2

- Commit it to the stack

- Create its neighbors

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
he does not -3.2	**-3.9**	**-4.1**	**-4.7**	-4.9	-5.3
he just does -3.5	**-4.3**	**-4.4**	-4.9	-5.2	-5.6
it does not -4.1	-5.1	-5.3	-5.5	-5.8	-6.2
he just does not -4.3	-5.3	-5.5	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2

- Commit best neighbor to the stack

- Create its neighbors in turn

Cube Pruning (4)

	-1.0 go	-1.2 walk	-1.4 goes	-1.7 are	-2.1 is
he does not -3.2	**-3.9**	**-4.1**	**-4.7**	-4.9	-5.3
he just does -3.5	**-4.3**	**-4.4**	-4.9	-5.2	-5.6
it does not -4.1	**-4.0**	-5.3	-5.5	-5.8	-6.2
he just does not -4.3	-5.3	-5.5	-5.7	-6.0	-6.4
he is not -4.7	-5.7	-5.9	-6.1	-6.4	-6.8
it is not -5.1	-6.1	-6.3	-6.5	-6.8	-7.2

- Keep doing this for a specific number of hypothesis

- Different hypothesis / translation options groups compete as well

heafield pruning

- Main idea

  - a lot of hypotheses share suffixes
  - a lot of translation options share prefixes
  - combining
    * the last word of a hypothesis
    * the first word of a translation options
    may already indicate if we should pursue further

- Method

  - organize hypotheses by suffix tree
  - organize translation options by prefix tree
  - process priority queue based on pairs of nodes in these trees

Example

Hypotheses with 2 words translated

- -2.1 a big country
- -2.2 large countries
- -2.7 the big countries
- -2.8 a large country
- -2.9 the big country
- -3.1 a big nation

Translation options for a source span

- -1.1 does not waver
- -1.5 do not waver
- -1.7 wavers not
- -1.9 does not hesitate
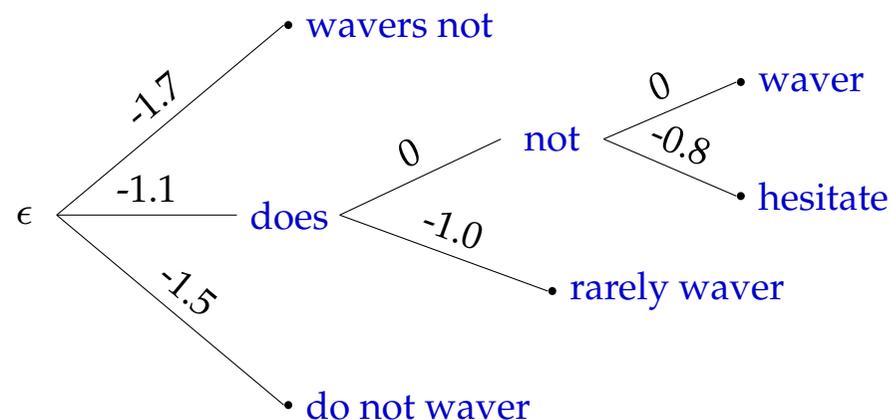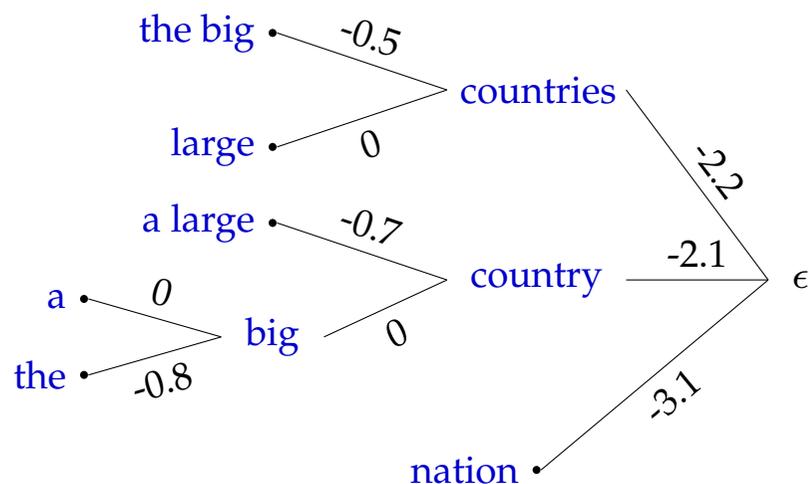- -2.1 does rarely waver

Encode in Suffix and Prefix Trees

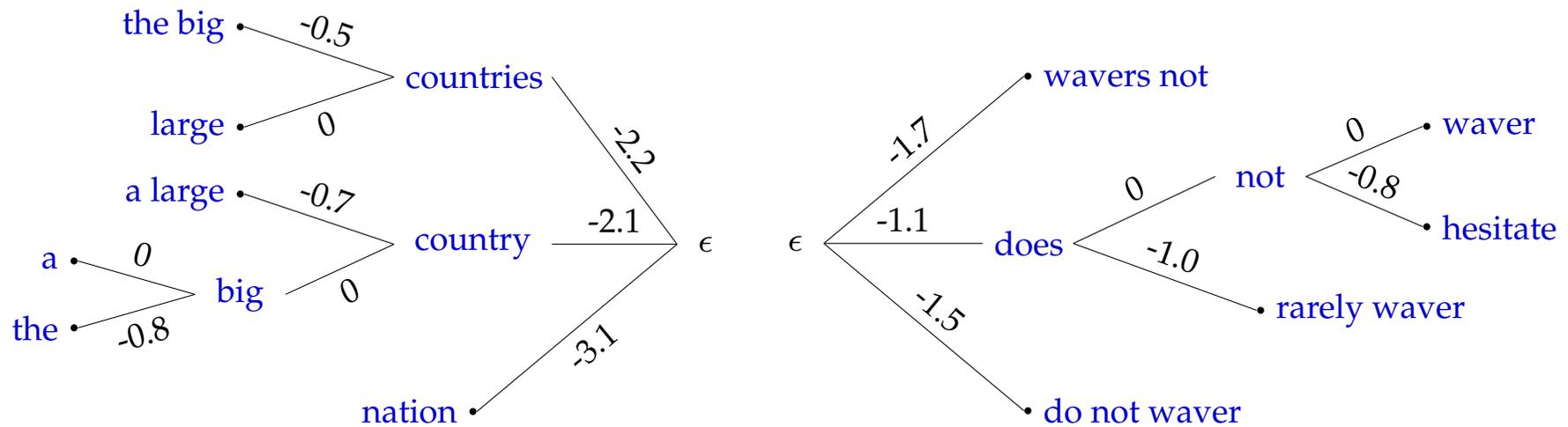**Hypotheses with 2 words translated**

- -2.1 a big country
- -2.2 large countries
- -2.7 the big countries
- -2.8 a large country
- -2.9 the big country
- -3.1 a big nation

Translation options for a source span

- -1.1 does not waver
- -1.5 do not waver
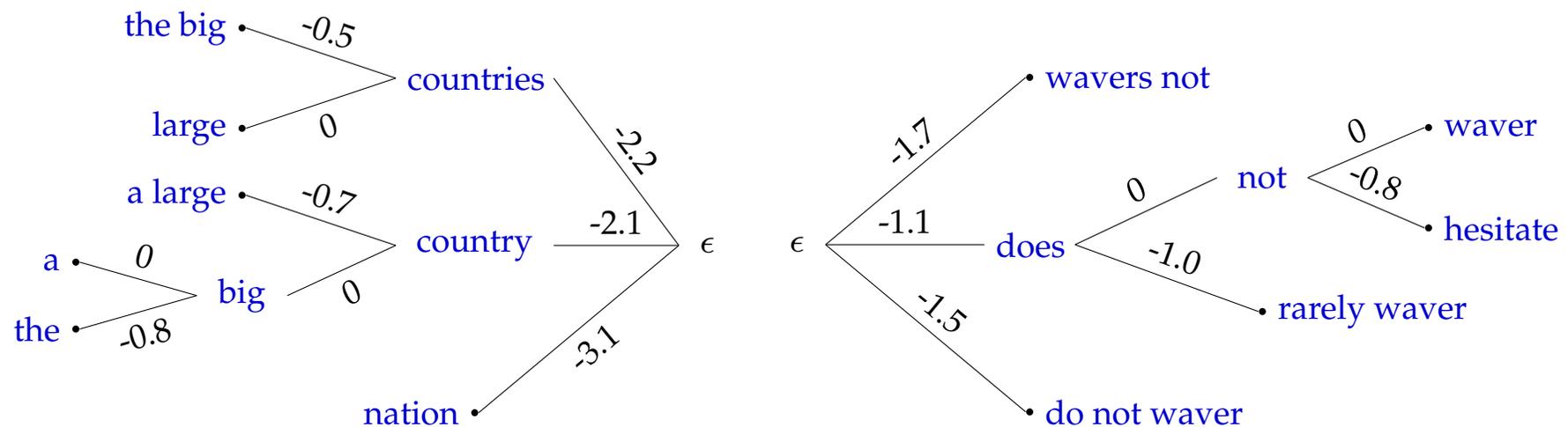- -1.7 wavers not
- -1.9 does not hesitate
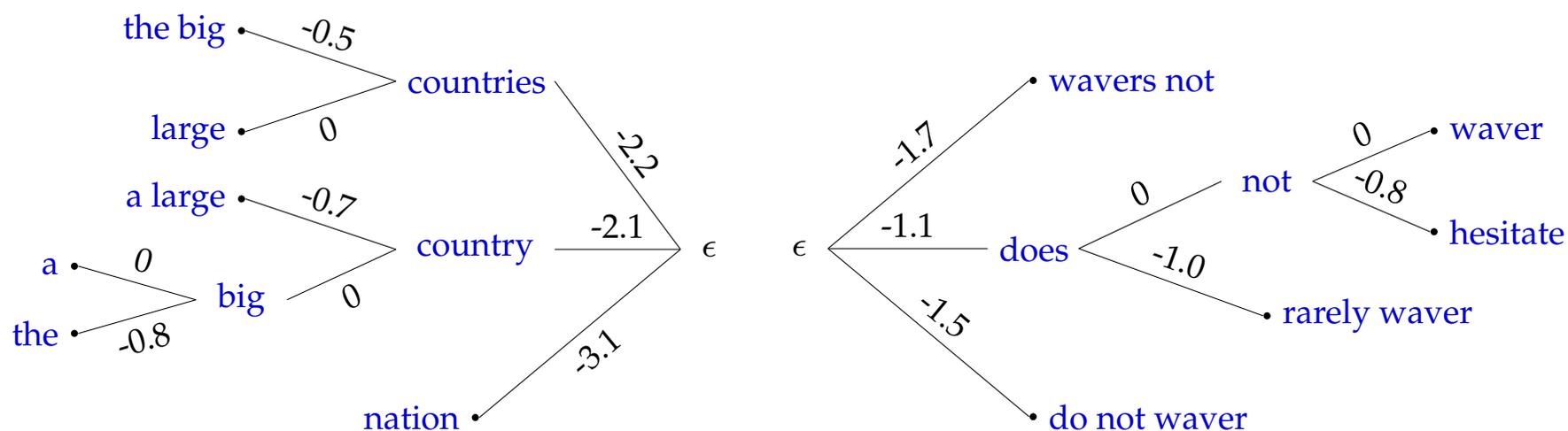- -2.1 does rarely waver

Set up Priority Queue

- Priority queue
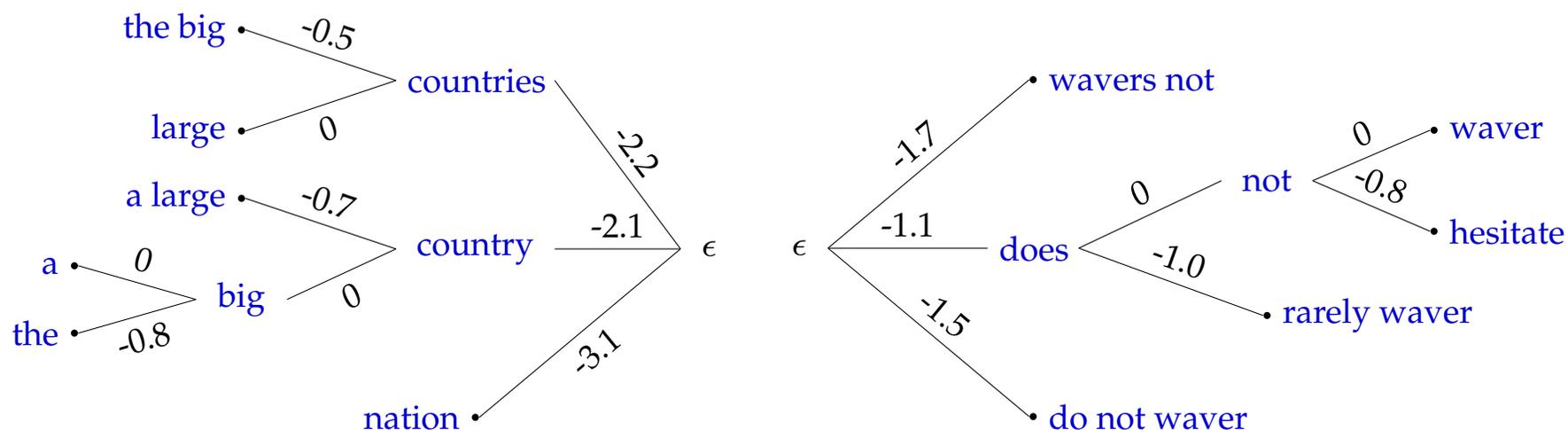 - (ϵ,ϵ), score: -3.2 (-2.1 + -1.1)

the big • —0.5
countries
large • 0

a large • —0.7
country —2.1
a • 0 ε
big 0
the • —0.8

nation • —3.1

wavers not •
—1.7
—1.1 ε does
—1.5 0 not 0 • waver
—1.0 —0.8
rarely waver • • hesitate
do not waver •

- Priority queue
  - (ϵ, ϵ), score: -3.2 (-2.1 + -1.1)
- Pop off: (ϵ, ϵ)
- Expand left (hypothesis): best is country
- Add new items
  - (country, ϵ), score: -3.2 (-2.1 + -1.1)
  - $(\epsilon[1+], \epsilon)$, score: -3.3 (-2.2 + -1.1)

- Priority queue
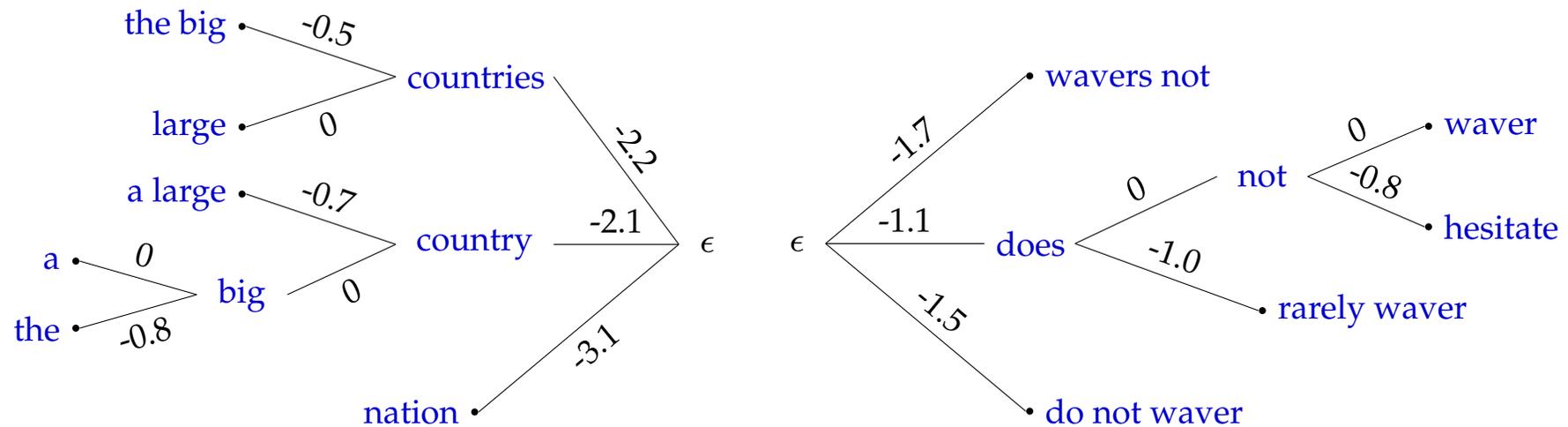 - (country,ϵ), score: -3.2 (-2.1 + -1.1)
  - (ϵ[1+],ϵ), score: -3.3 (-2.2 + -1.1)
- Pop off: (country,ϵ)
- Expand left (translation option): best is does
- Update language model probability estimate $\log\frac{p(\text{does}|\text{country})}{p(\text{does})} = +0.2$
- Add new items
  - (country,does), score: -3.0 (-2.1 + -1.1 + +0.2)
  - (country,ϵ[1+]), score: -3.6 (-2.1 + -1.5)

- Priority queue
  - (country,does), score: -3.0 (-2.1 + -1.1 + +0.2)
  - (ϵ[1+],ϵ), score: -3.3 (-2.2 + -1.1)
  - (country,ϵ[1+]), score: -3.6 (-2.1 + -1.5)
- Pop off: (country,does)
- Expand left (hypothesis): best is big
- Update language model probability estimate $\log\frac{p(\text{does}|\text{big country})}{p(\text{does}|\text{country})} = +0.1$
- Add new items
  - (big country,does), score: -2.9 (-2.1 + -1.1 + +0.2 + +0.1)
  - (country[1+],does), score: -3.7 (-2.1 + -1.1 + +0.2 + -0.7)

- Priority queue
  - (big country,does), score: -2.9 (-2.1 + -1.1 + +0.2 + +0.1)
  - (ϵ[1+],ϵ), score: -3.3 (-2.2 + -1.1)
  - (country,ϵ[1+]), score: -3.6 (-2.1 + -1.5)
  - (country[1+],does), score: -3.7 (-2.1 + -1.1 + +0.2 + -0.7)

- And so on...
  - once a full combination is completed (a big country,does not waver), add it to the stack
  - badly matching updates will push items down the priority queue
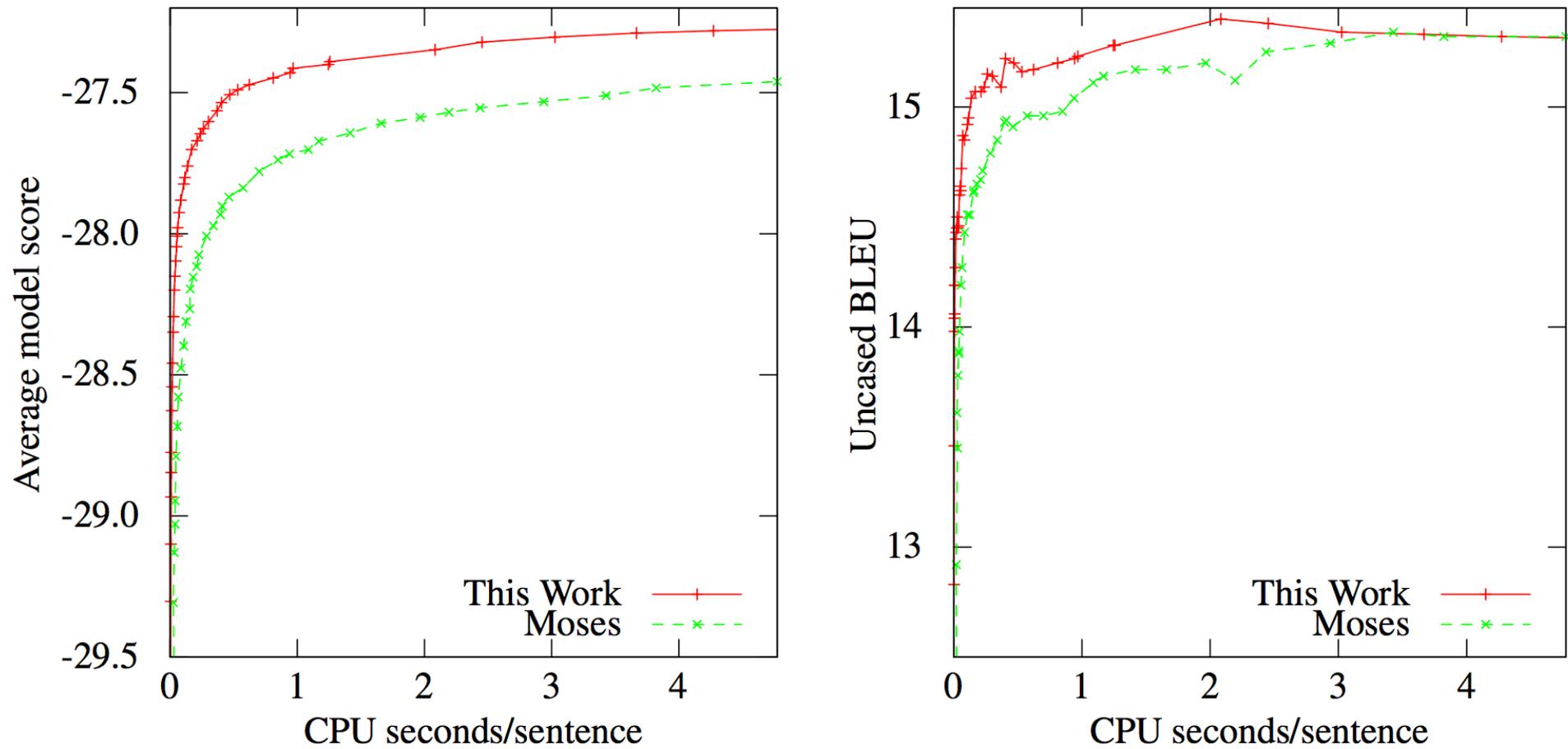   e.g., $\log \frac{p(\text{does}|\text{countries})}{p(\text{does})} = -2.1$

Figure 4: Performance of our decoder and Moses for various stack sizes k.

other decoding algorithms

- A* search

- Greedy hill-climbing

- Using finite state transducers (standard toolkits)

③ alternative path leading to hypothesis beyond threshold

cheapest score

① depth-first expansion to completed path

② recombination

probability + heuristic estimate

number of words covered

- Uses *admissible* future cost heuristic: never overestimates cost

- Translation agenda: create hypothesis with lowest score + heuristic cost

- Done, when complete hypothesis created

- Create one complete hypothesis with depth-first search (or other means)

- Search for better hypotheses by applying change operators

  – change the translation of a word or phrase
  – combine the translation of two words into a phrase
  – split up the translation of a phrase into two smaller phrase translations
  – move parts of the output into a different position
  – swap parts of the output with the output at a different part of the sentence

- Terminates if no operator application produces a better translation

Summary

- Translation process: produce output left to right

- Translation options

- Decoding by hypothesis expansion

- Reducing search space

  – recombination
  – pruning (requires future cost estimate)

- Other decoding algorithms