Phrase-Based Models

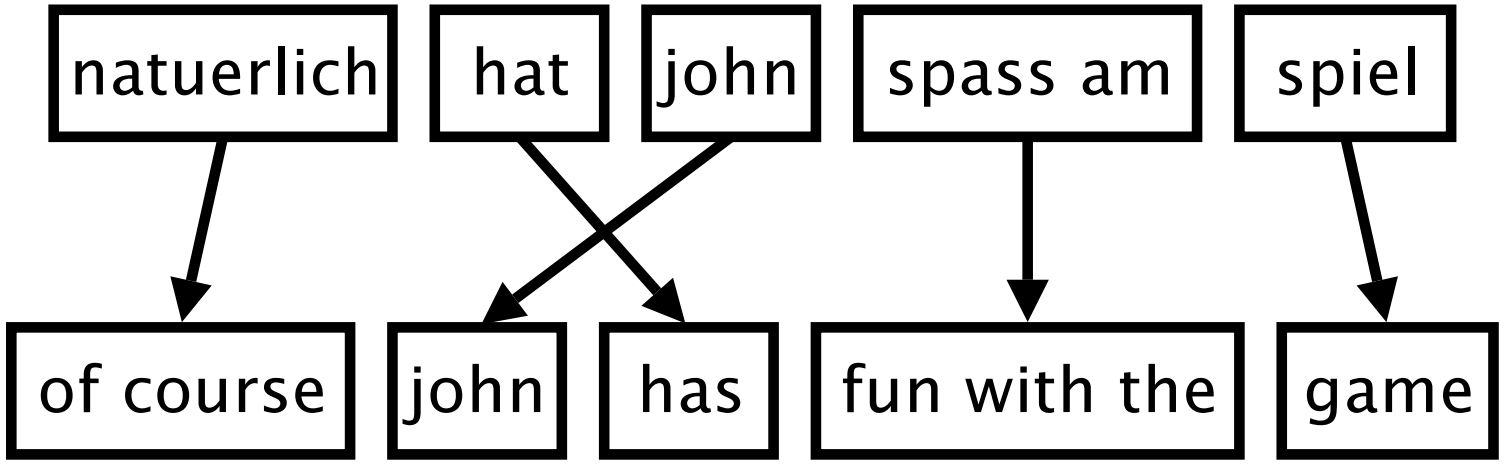
Philipp Koehn

9 September 2025

Motivation

- Word-Based Models translate *words* as atomic units
- Phrase-Based Models translate *phrases* as atomic units
- Advantages:
  - many-to-many translation can handle non-compositional phrases
  - use of local context in translation
  - the more data, the longer phrases can be learned
- "Standard Model", used by Google Translate and others until about 2017

Phrase-Based Model



- Foreign input is segmented in phrases
- Each phrase is translated into English
- Phrases are reordered

Phrase Translation Table

- Main knowledge source: table with phrase translations and their probabilities
- Example: phrase translations for natuerlich

Translation	Probability $\phi(\bar{e}\|\bar{f})$
of course	0.5
naturally	0.3
of course ,	0.15
, of course ,	0.05

Real Example

- Phrase translations for den Vorschlag learned from the Europarl corpus:

English	$\phi(\bar{e}\|\bar{f})$	English	$\phi(\bar{e}\|\bar{f})$
the proposal	0.6227	the suggestions	0.0114
's proposal	0.1068	the proposed	0.0114
a proposal	0.0341	the motion	0.0091
the idea	0.0250	the idea of	0.0091
this proposal	0.0227	the proposal ,	0.0068
proposal	0.0205	its proposal	0.0068
of the proposal	0.0159	it	0.0068
the proposals	0.0159	...	...

  - lexical variation (proposal vs suggestions)
  - morphological variation (proposal vs proposals)
  - included function words (the, a, ...)
  - noise (it)

Linguistic Phrases?

- Model is not limited to linguistic phrases
  (noun phrases, verb phrases, prepositional phrases, ...)
- Example non-linguistic phrase pair

  spass am → fun with the
- Prior noun often helps with translation of preposition
- Experiments show that limitation to linguistic phrases hurts quality

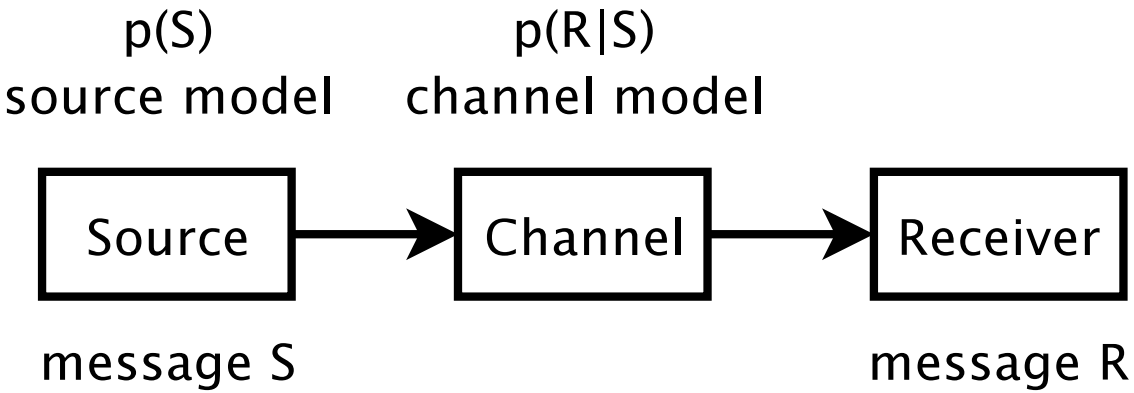
modeling

Noisy Channel Model

- We would like to integrate a language model
- Bayes rule

$$\begin{aligned} \text{argmax}_{\mathbf{e}}\, p(\mathbf{e}|\mathbf{f}) &= \text{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e})}{p(\mathbf{f})} \\ &= \text{argmax}_{\mathbf{e}}\, p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e}) \end{aligned}$$

Noisy Channel Model



- Applying Bayes rule also called noisy channel model
  - we observe a distorted message R (here: a foreign string **f**)
  - we have a model on how the message is distorted (here: translation model)
  - we have a model on what messages are probably (here: language model)
  - we want to recover the original message S (here: an English string **e**)

More Detail

- Bayes rule

$$\begin{aligned} \mathbf{e}_{\text{best}} &= \text{argmax}_{\mathbf{e}}\ p(\mathbf{e}|\mathbf{f}) \\ &= \text{argmax}_{\mathbf{e}}\ p(\mathbf{f}|\mathbf{e})\ p_{\text{LM}}(\mathbf{e}) \end{aligned}$$

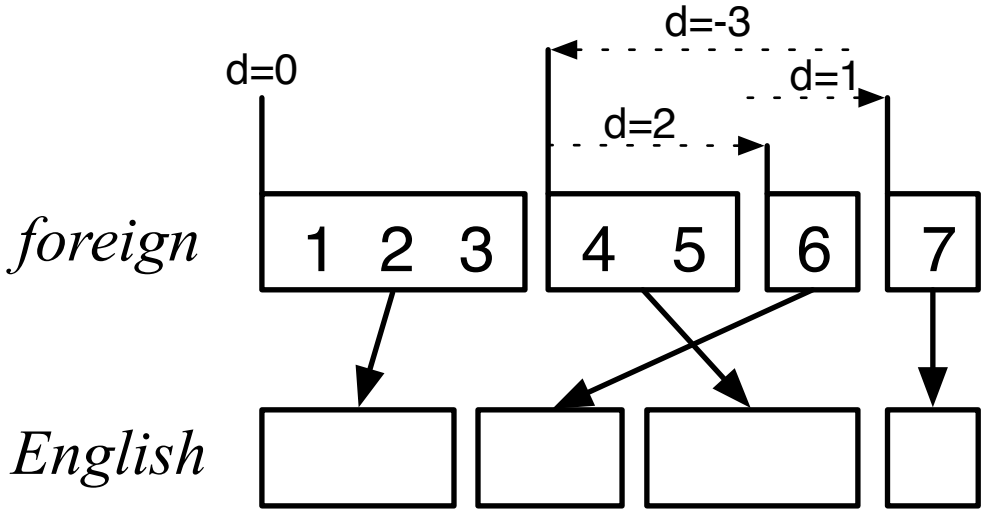
  - translation model $p(\mathbf{f}|\mathbf{e})$
  - language model $p_{\text{LM}}(\mathbf{e})$

- Decomposition of the translation model

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)\ d(start_i - end_{i-1} - 1)$$

  - phrase translation probability ϕ
  - reordering probability d

Distance-Based Reordering



phrase	translates	movement	distance
1	1–3	start at beginning	0
2	6	skip over 4–5	+2
3	4–5	move back over 4–6	-3
4	7	skip over 6	+1

Scoring function: $d(x) = \alpha^{|x|}$ — exponential with distance

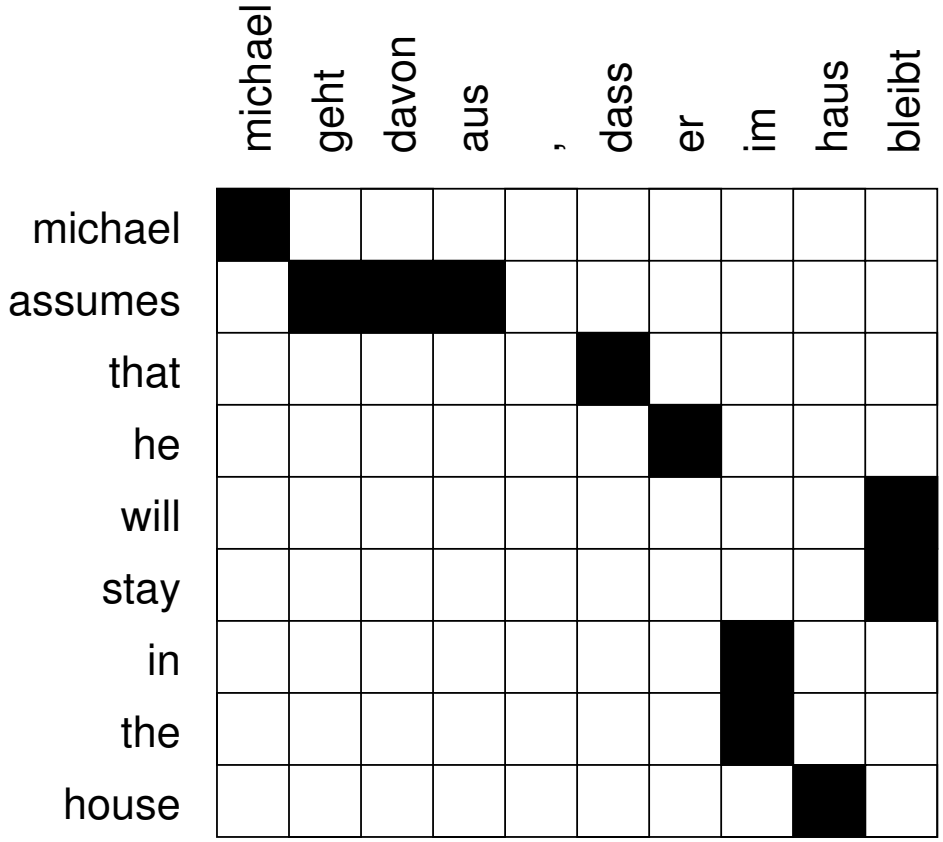
training

Learning a Phrase Translation Table

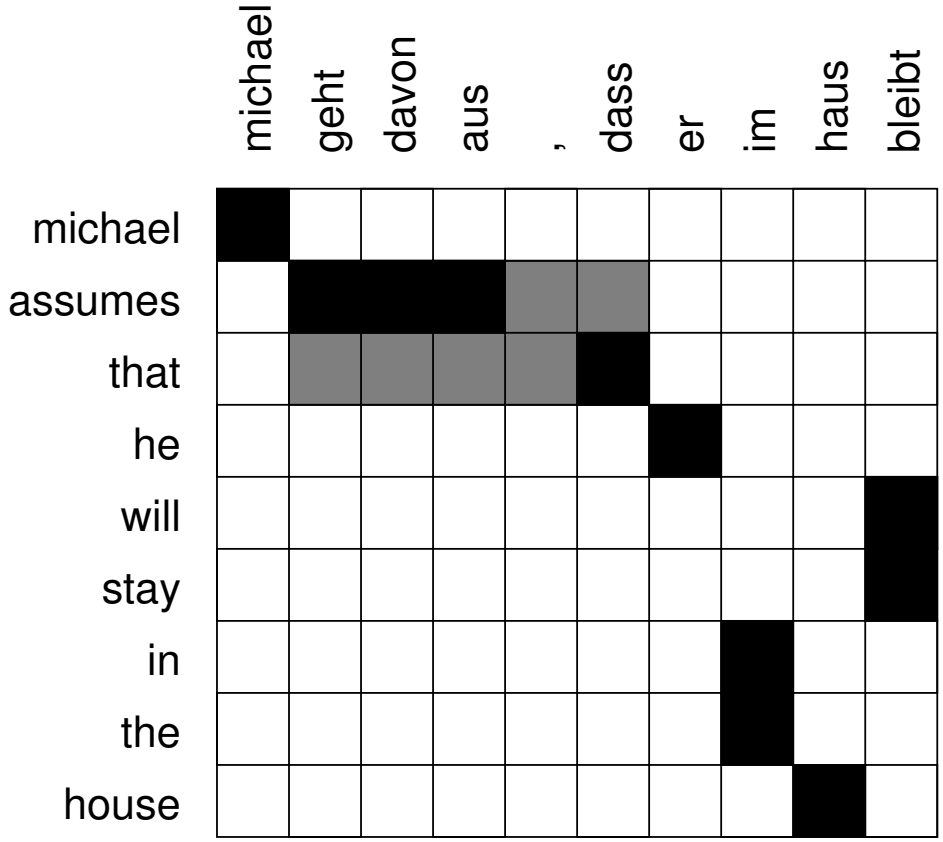
- Task: learn the model from a parallel corpus

- Three stages:
  - word alignment: using IBM models or other method
  - extraction of phrase pairs
  - scoring phrase pairs

Word Alignment



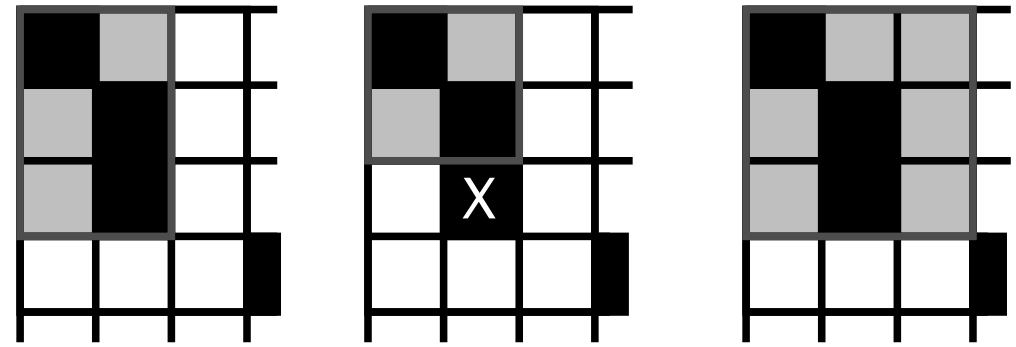
Extracting Phrase Pairs



extract phrase pair consistent with word alignment:

assumes that / geht davon aus , dass

Consistent



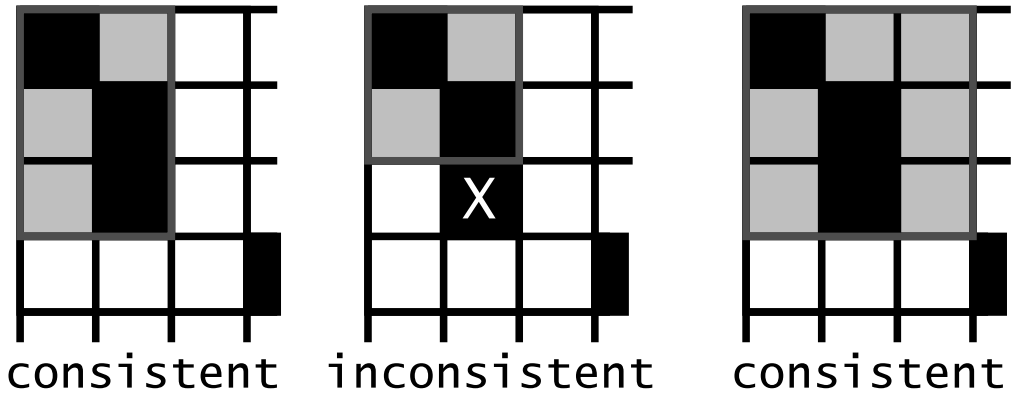
ok

violated
one
alignment
point outside

ok
unaligned
word is fine

All words of the phrase pair have to align to each other.

Consistent



Phrase pair $(\bar{e}, \bar{f})$ consistent with an alignment A, if all words $f_1, ..., f_n$ in $\bar{f}$ that have alignment points in A have these with words $e_1, ..., e_n$ in $\bar{e}$ and vice versa:

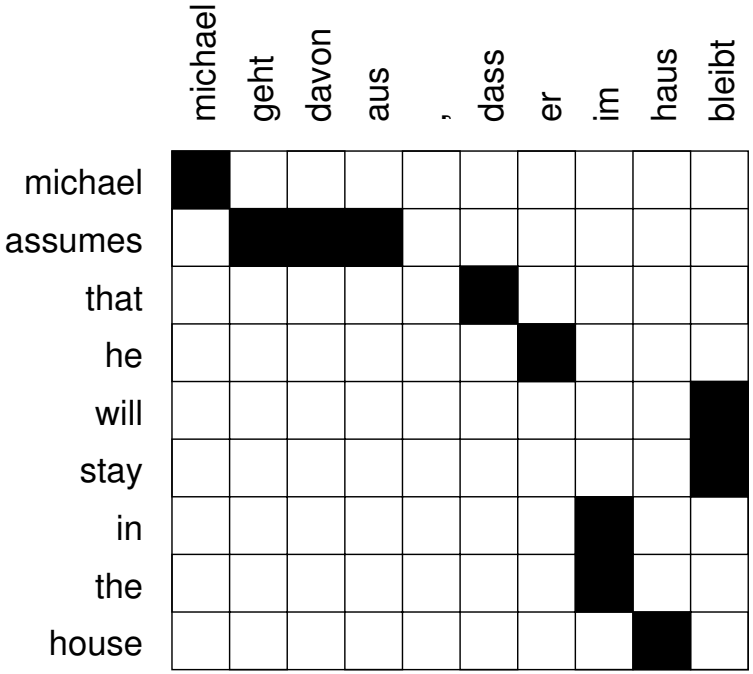
$$(\bar{e}, \bar{f}) \text{ consistent with } A \Leftrightarrow$$

$$\forall e_i \in \bar{e} : (e_i, f_j) \in A \rightarrow f_j \in \bar{f}$$

$$\text{AND } \forall f_j \in \bar{f} : (e_i, f_j) \in A \rightarrow e_i \in \bar{e}$$

$$\text{AND } \exists e_i \in \bar{e}, f_j \in \bar{f} : (e_i, f_j) \in A$$

Phrase Pair Extraction



Smallest phrase pairs:

michael — michael

assumes — geht davon aus / geht davon aus ,

that — dass / , dass

he — er

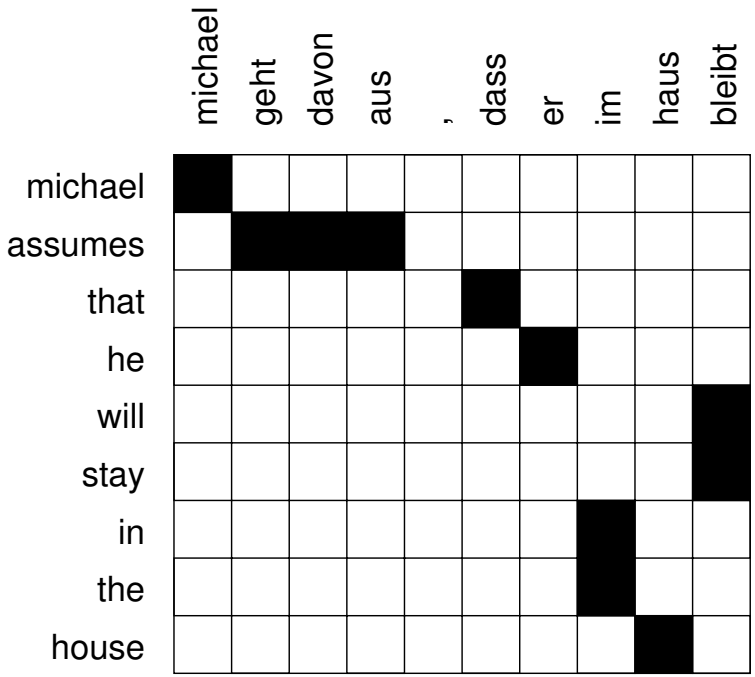
will stay — bleibt

in the — im

house — haus

unaligned words (here: German comma) lead to multiple translations

Larger Phrase Pairs



michael assumes — michael geht davon aus / michael geht davon aus ,
assumes that — geht davon aus , dass ; assumes that he — geht davon aus , dass er
that he — dass er / , dass er ; in the house — im haus
michael assumes that — michael geht davon aus , dass
michael assumes that he — michael geht davon aus , dass er
michael assumes that he will stay in the house — michael geht davon aus , dass er im haus bleibt
assumes that he will stay in the house — geht davon aus , dass er im haus bleibt
that he will stay in the house — dass er im haus bleibt ; dass er im haus bleibt ,
he will stay in the house — er im haus bleibt ; will stay in the house — im haus bleibt

Scoring Phrase Translations

- Phrase pair extraction: collect all phrase pairs from the data
- Phrase pair scoring: assign probabilities to phrase translations
- Score by relative frequency:

$$\phi(\bar{f}|\bar{e}) = \frac{\text{count}(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} \text{count}(\bar{e}, \bar{f}_i)}$$

EM Training of the Phrase Model

- We presented a heuristic set-up to build phrase translation table
(word alignment, phrase extraction, phrase scoring)

- Alternative: align phrase pairs directly with EM algorithm

  - initialization: uniform model, all $\phi(\bar{e}, \bar{f})$ are the same
  - expectation step:
    - estimate likelihood of all possible phrase alignments for all sentence pairs
  - maximization step:
    - collect counts for phrase pairs $(\bar{e}, \bar{f})$, weighted by alignment probability
    - update phrase translation probabilties $p(\bar{e}, \bar{f})$

- However: method easily overfits
(learns very large phrase pairs, spanning entire sentences)

Size of the Phrase Table

- Phrase translation table typically bigger than corpus

  ... even with limits on phrase lengths (e.g., max 7 words)

→ Too big to store in memory?

- Solution for training
  - extract to disk, sort, construct for one source phrase at a time
- Solutions for decoding
  - on-disk data structures with index for quick look-ups
  - suffix arrays to create phrase pairs on demand

advanced modeling

Weighted Model

- Described standard model consists of three sub-models
  - phrase translation model $\phi(\bar{f}|\bar{e})$
  - reordering model d
  - language model $p_{LM}(e)$

$$e_{\text{best}} = \text{argmax}_e \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)\ d(start_i - end_{i-1} - 1) \prod_{i=1}^{|\mathbf{e}|} p_{LM}(e_i|e_1...e_{i-1})$$

- Some sub-models may be more important than others

- Add weights λ_ϕ, λ_d, λ_{LM}

$$e_{\text{best}} = \text{argmax}_e \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)^{\lambda_\phi}\ d(start_i - end_{i-1} - 1)^{\lambda_d} \prod_{i=1}^{|\mathbf{e}|} p_{LM}(e_i|e_1...e_{i-1})^{\lambda_{LM}}$$

Log-Linear Model

- Such a weighted model is a log-linear model:

$$p(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x)$$

- Our feature functions
  - number of feature function $n = 3$
  - random variable $x = (e, f, start, end)$
  - feature function $h_1 = \log \phi$
  - feature function $h_2 = \log d$
  - feature function $h_3 = \log p_{\text{LM}}$

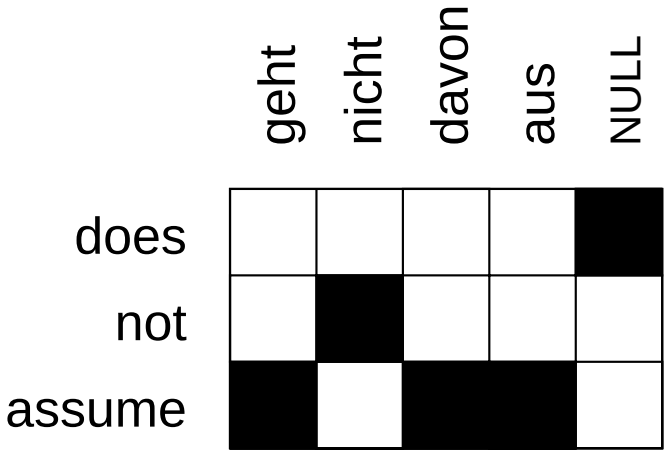
Weighted Model as Log-Linear Model

$$p(e,a|f) = \exp(\lambda_\phi \sum_{i=1}^{I} \log \phi(\bar{f}_i|\bar{e}_i) +$$
$$\lambda_d \sum_{i=1}^{I} \log d(a_i - b_{i-1} - 1) +$$
$$\lambda_{LM} \sum_{i=1}^{|\mathrm{e}|} \log p_{LM}(e_i|e_1 ... e_{i-1}))$$

More Feature Functions

- Bidirectional alignment probabilities: $\phi(\bar{e}|\bar{f})$ and $\phi(\bar{f}|\bar{e})$
- Rare phrase pairs have unreliable phrase translation probability estimates

  $\rightarrow$ lexical weighting with word translation probabilities



$$\text{lex}(\bar{e}|\bar{f},a) = \prod_{i=1}^{\text{length}(\bar{e})} \frac{1}{|\{j|(i,j)\in a\}|} \sum_{\forall(i,j)\in a} w(e_i|f_j)$$

More Feature Functions

- Language model has a bias towards short translations

  $\rightarrow$ word count: $\text{wc}(e) = \log |\mathbf{e}|^{\omega}$

- We may prefer finer or coarser segmentation

  $\rightarrow$ phrase count $\text{pc}(e) = \log |I|^{\rho}$

- Multiple language models

- Multiple translation models

- Other knowledge sources

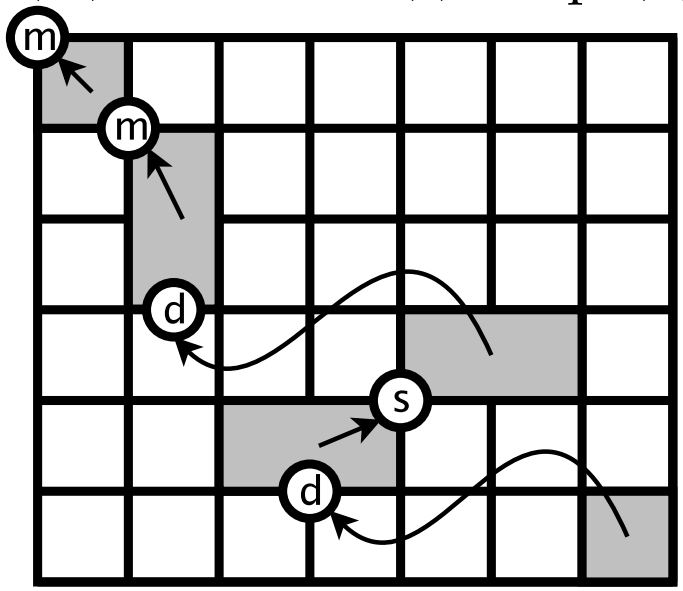
reordering

Lexicalized Reordering

- Distance-based reordering model is weak

  $\rightarrow$ learn reordering preference for each phrase pair

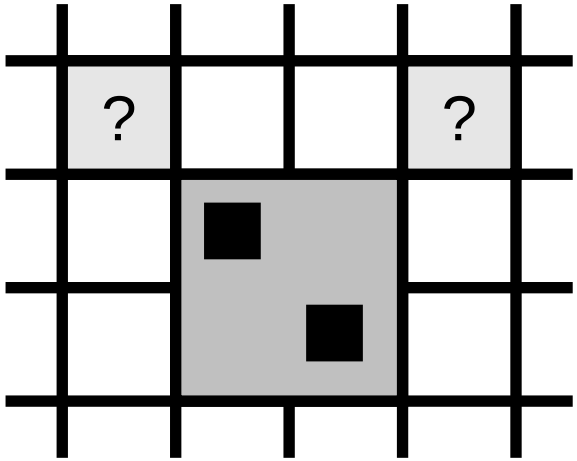
- Three orientations types: (m) monotone, (s) swap, (d) discontinuous



$$\text{orientation} \in \{m, s, d\}$$

$$p_o(\text{orientation}|\bar{f}, \bar{e})$$

Learning Lexicalized Reordering



- Collect orientation information during phrase pair extraction
  - if word alignment point to the top left exists → **monotone**
  - if a word alignment point to the top right exists→ **swap**
  - if neither a word alignment point to top left nor to the top right exists → neither monotone nor swap → **discontinuous**

Learning Lexicalized Reordering

- Estimation by relative frequency

$$p_o(\text{orientation}) = \frac{\sum_{\bar{f}} \sum_{\bar{e}} count(\text{orientation}, \bar{e}, \bar{f})}{\sum_o \sum_{\bar{f}} \sum_{\bar{e}} count(o, \bar{e}, \bar{f})}$$

- Smoothing with unlexicalized orientation model $p(\text{orientation})$ to avoid zero probabilities for unseen orientations

$$p_o(\text{orientation}|\bar{f}, \bar{e}) = \frac{\sigma\ p(\text{orientation}) + count(\text{orientation}, \bar{e}, \bar{f})}{\sigma + \sum_o count(o, \bar{e}, \bar{f})}$$

translation process

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

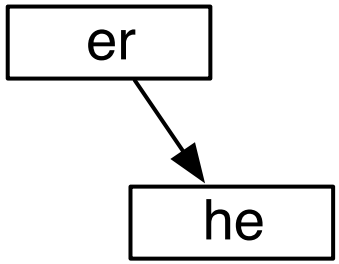
- Task: translate this sentence from German into English

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

Translation Process

- Task: translate this sentence from German into English

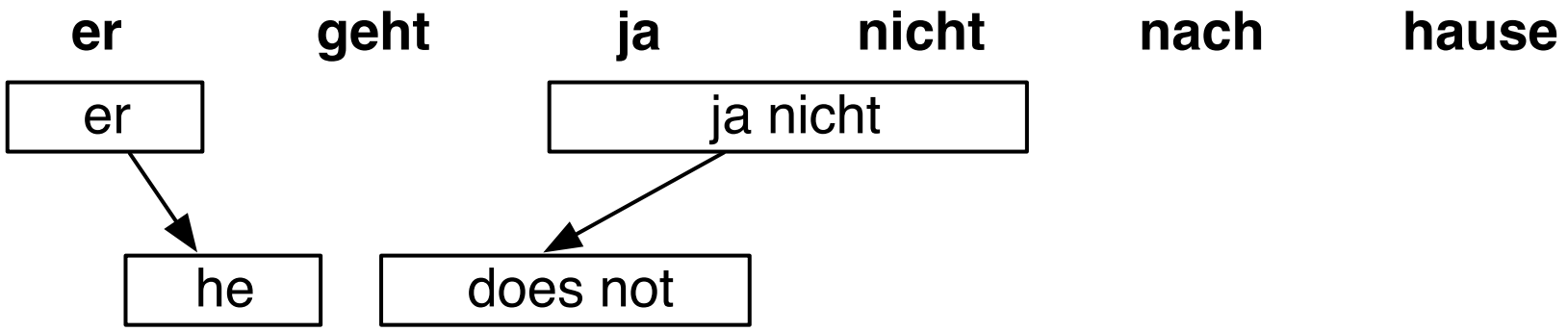
er **geht** **ja** **nicht** **nach** **hause**



- Pick phrase in input, translate

Translation Process

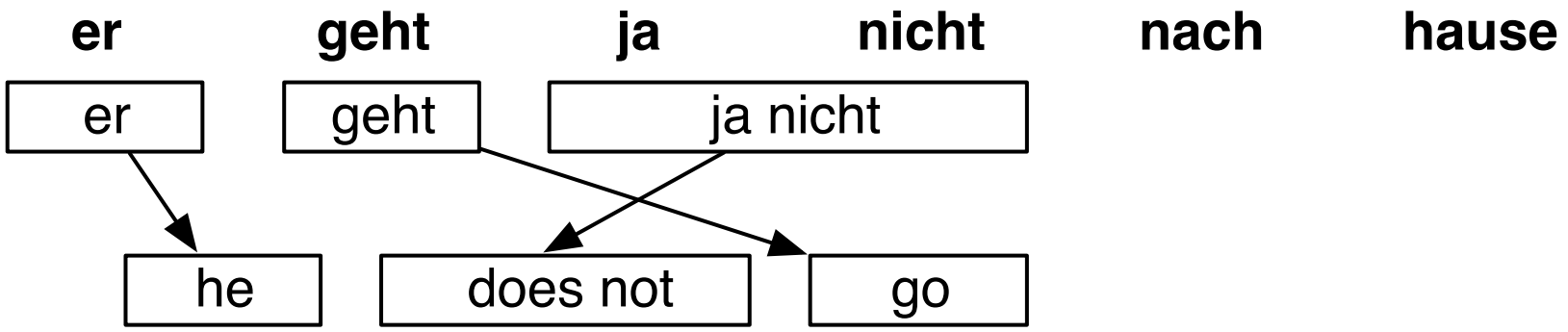
- Task: translate this sentence from German into English



- Pick phrase in input, translate
  - it is allowed to pick words out of sequence reordering
  - phrases may have multiple words: many-to-many translation

Translation Process

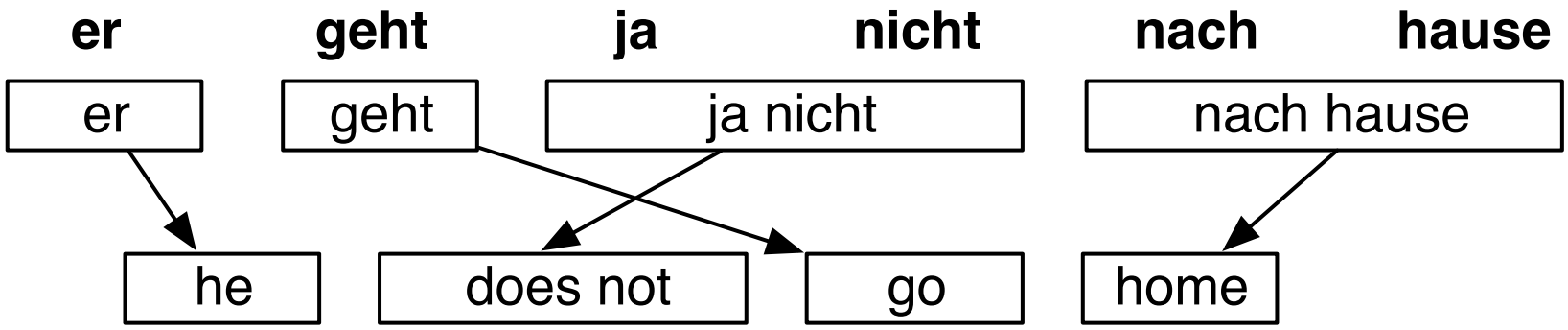
- Task: translate this sentence from German into English



- Pick phrase in input, translate

Translation Process

- Task: translate this sentence from German into English



- Pick phrase in input, translate

Computing Translation Probability

- 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

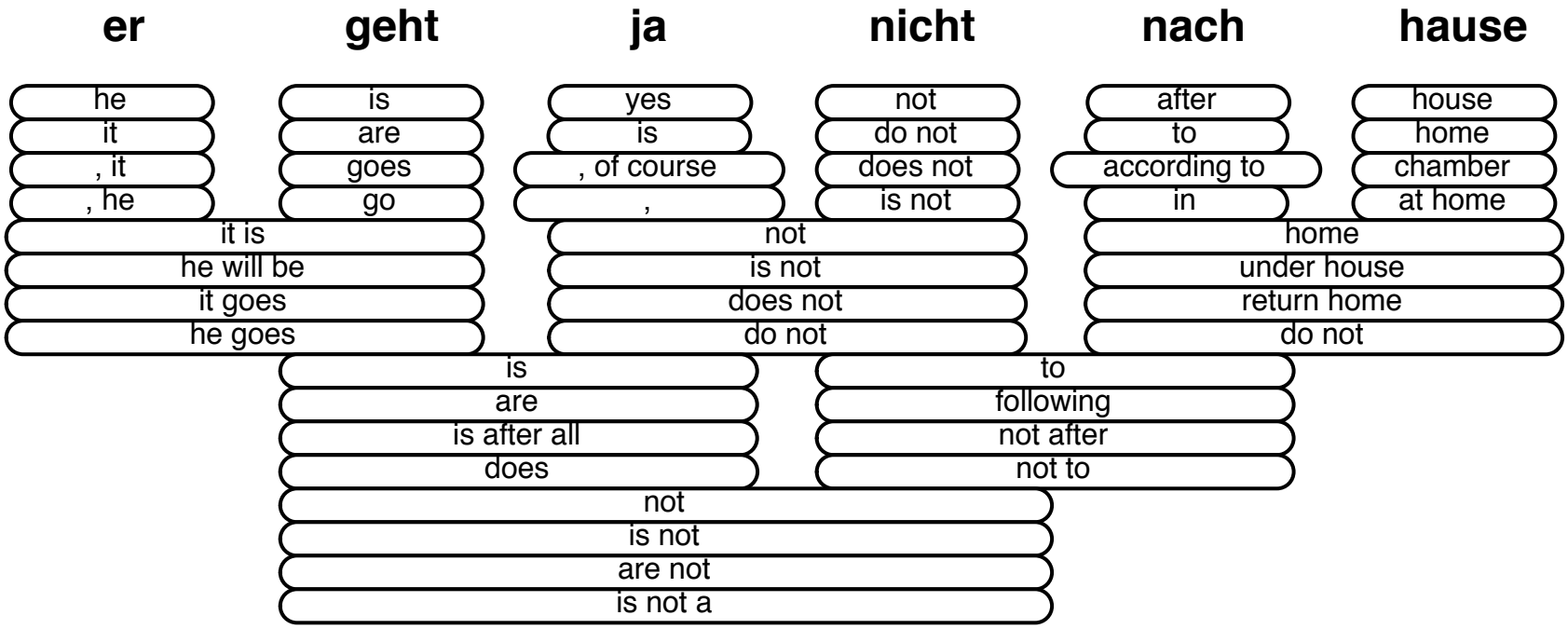
Phrase translation Picking phrase $\bar{f}_i$ to be translated as a phrase $\bar{e}_i$
→ look up score $\phi(\bar{f}_i|\bar{e}_i)$ from phrase translation table

Reordering Previous phrase ended in end_{i-1}, current phrase starts at $start_i$
→ compute $d(start_i - end_{i-1} - 1)$

Language model For n-gram model, need to keep track of last $n-1$ words
→ compute score $p_{\text{LM}}(w_i|w_{i-(n-1)}, ..., w_{i-1})$ for added words w_i

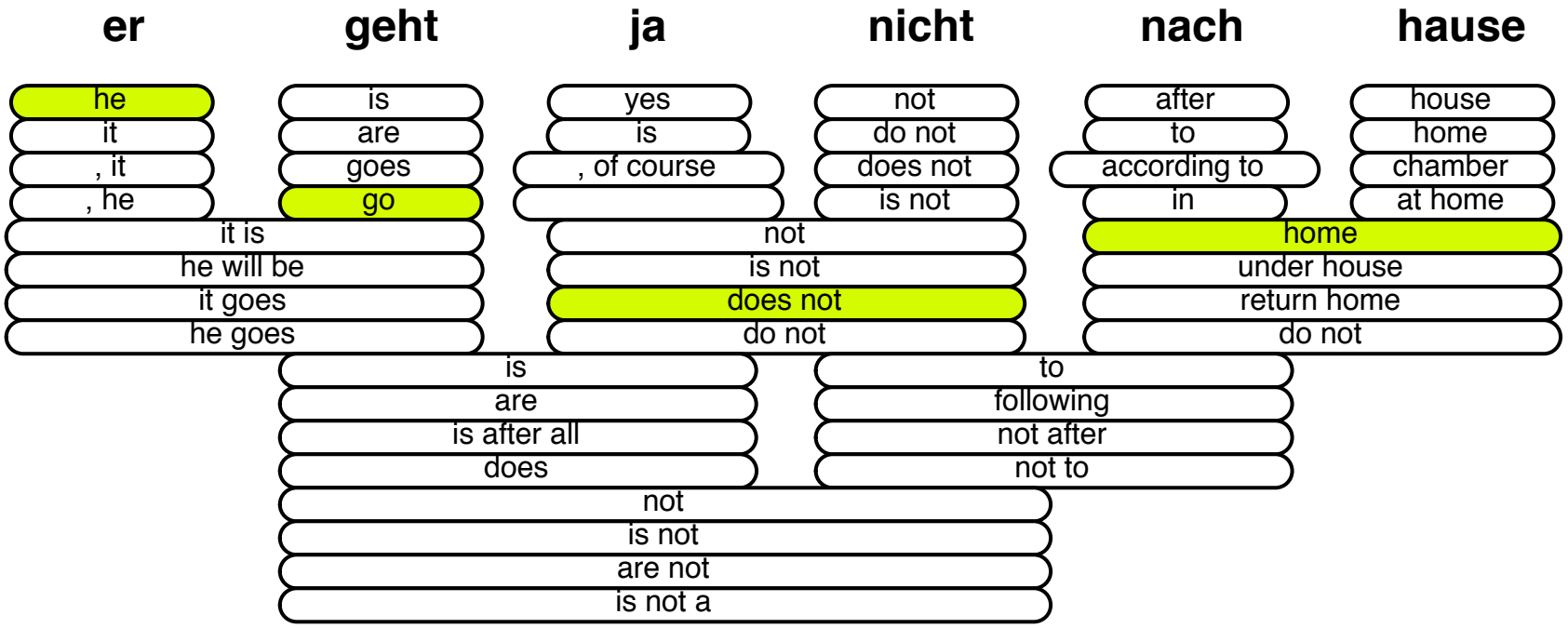
decoding process

Translation Options



- 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

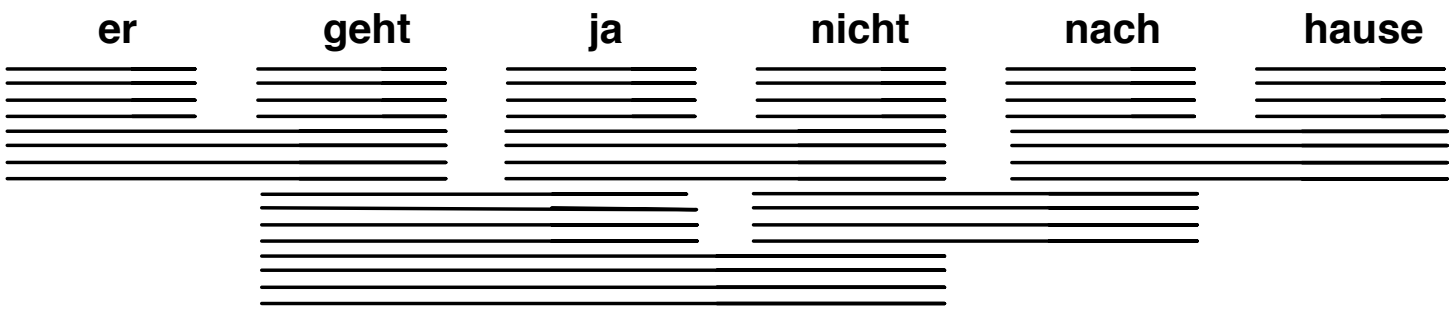
Translation Options



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

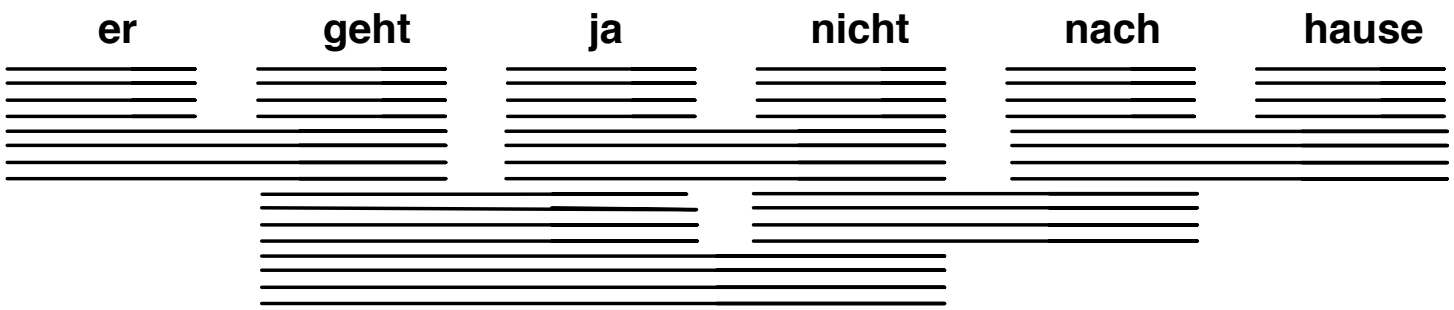
→ Search problem solved by heuristic beam search

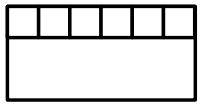
Decoding: Precompute Translation Options



consult phrase translation table for all input phrases

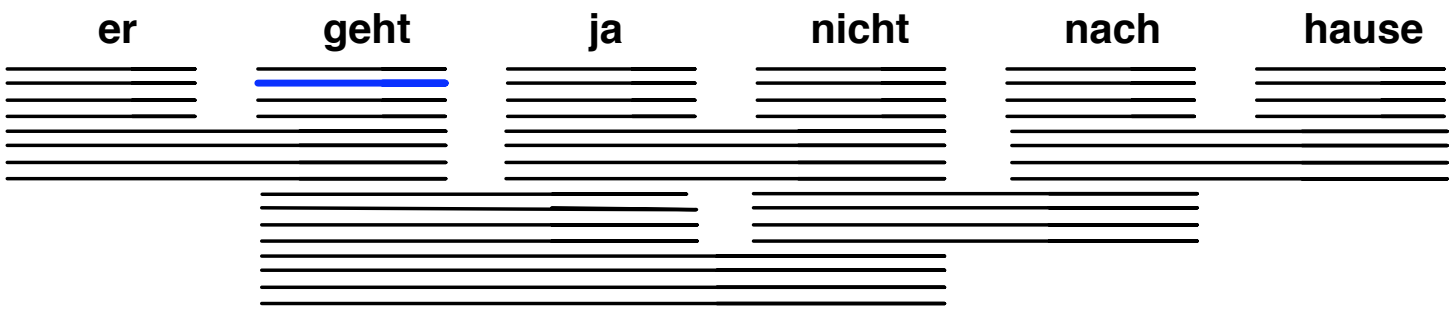
Decoding: Start with Initial Hypothesis

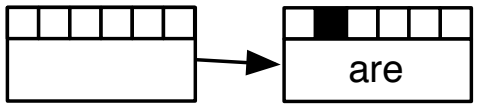




initial hypothesis: no input words covered, no output produced

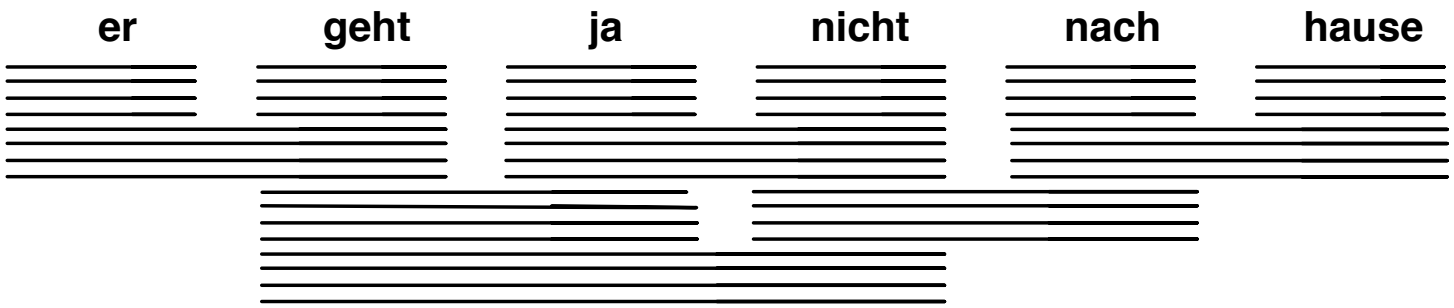
Decoding: Hypothesis Expansion

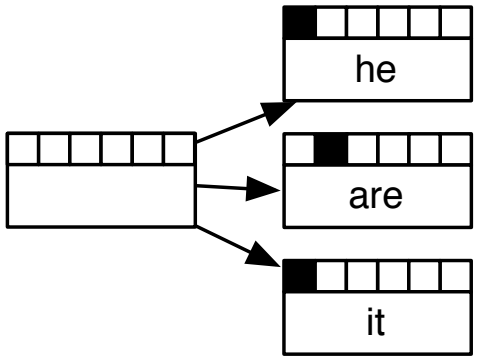




pick any translation option, create new hypothesis

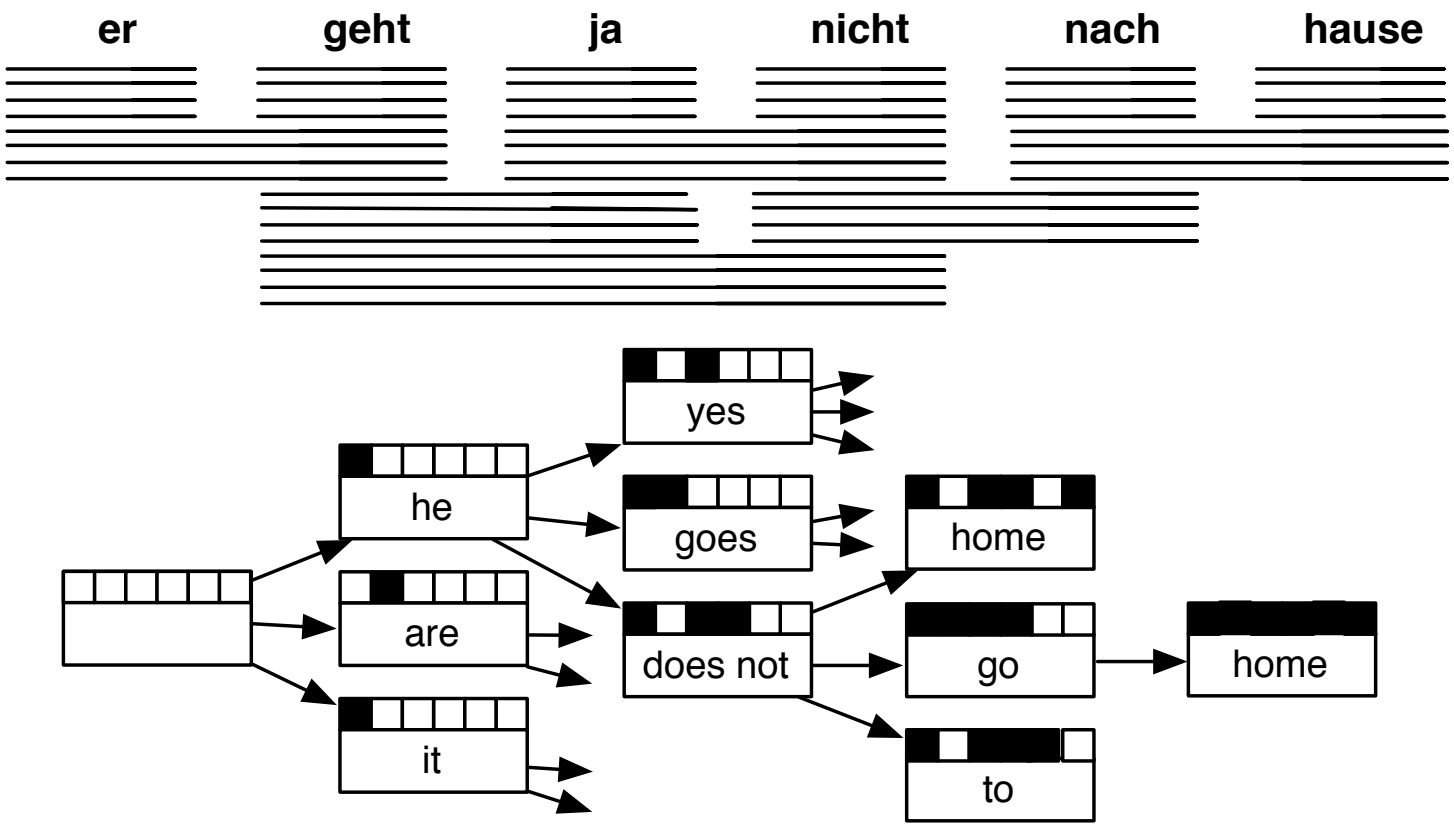
Decoding: Hypothesis Expansion





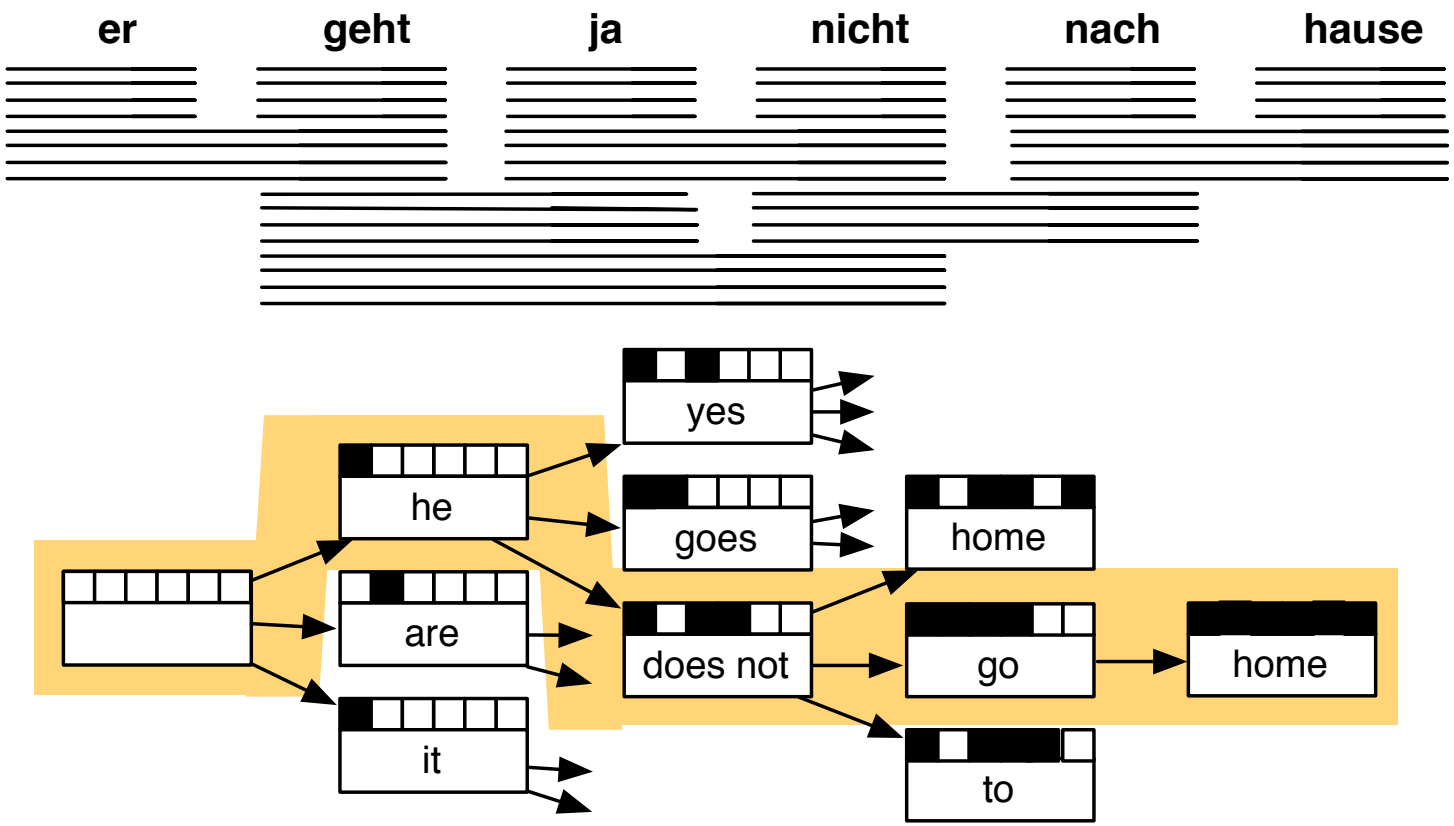
create hypotheses for all other translation options

Decoding: Hypothesis Expansion



also create hypotheses from created partial hypothesis

Decoding: Find Best Path



backtrack from highest scoring complete hypothesis

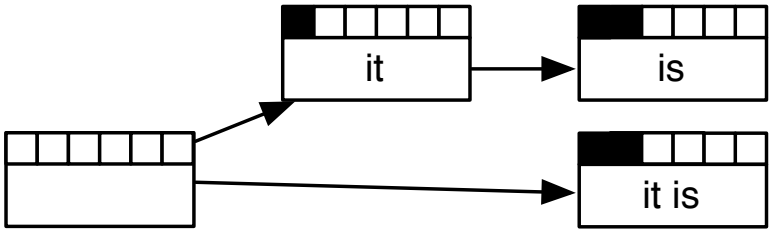
dynamic programming

Computational Complexity

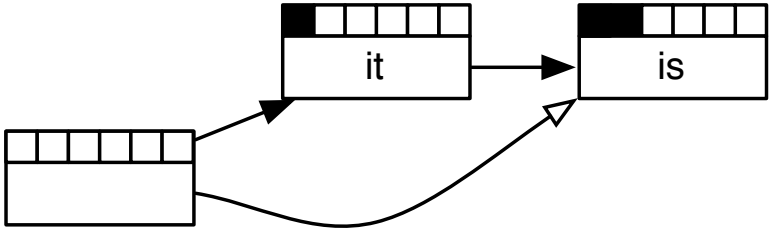
- The suggested process creates exponential number of hypothesis
- Machine translation decoding is NP-complete
- Reduction of search space:
  - recombination (risk-free)
  - pruning (risky)

Recombination

- Two hypothesis paths lead to two matching hypotheses
  - same foreign words translated
  - same English words in the output

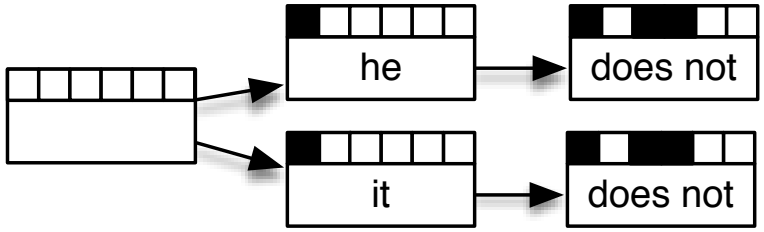


- Worse hypothesis is dropped

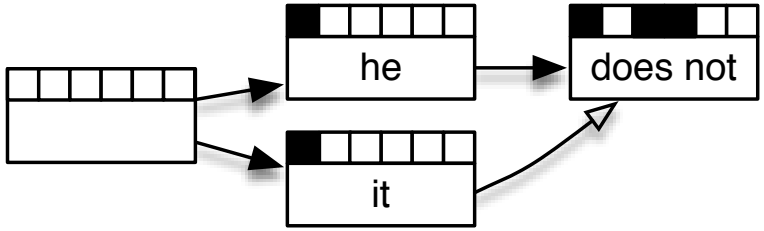


Recombination

- Two hypothesis paths lead to hypotheses indistinguishable in subsequent search
  - same foreign words translated
  - same last two English words in output (assuming trigram language model)
  - same last foreign word translated



- Worse hypothesis is dropped



Restrictions on Recombination

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

  → no restriction to hypothesis recombination

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

  → 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

  → recombined hypotheses must have that same end position

- Other feature function may introduce additional restrictions

pruning

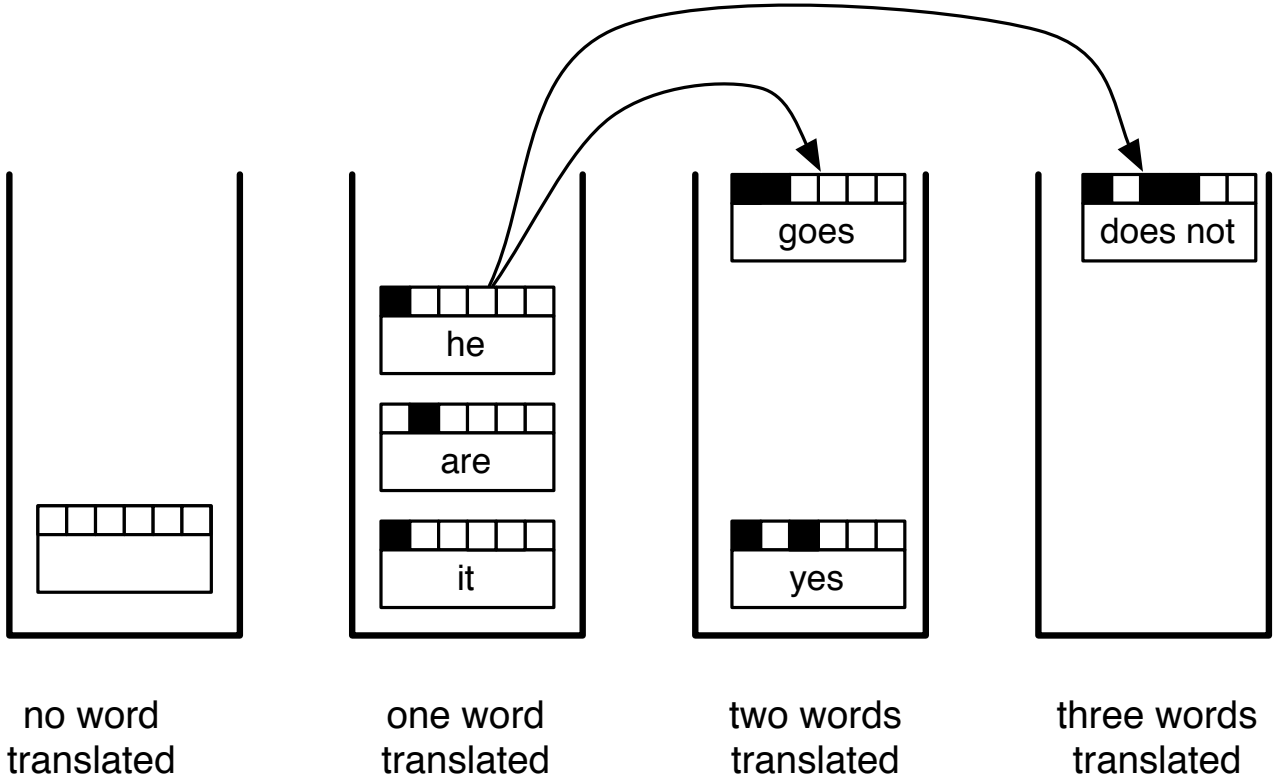
Pruning

- 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

Stacks



- 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

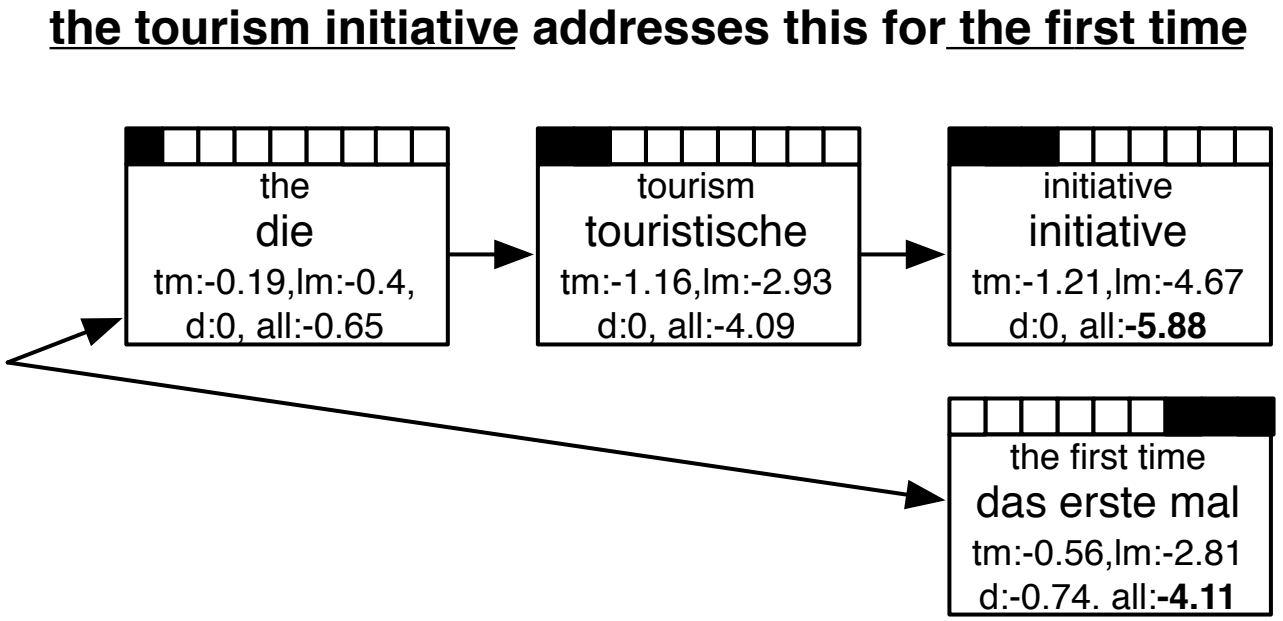
- Limiting reordering to maximum reordering distance
- Typical reordering distance 5–8 words
  - depending on language pair
  - larger reordering limit hurts translation quality
- Reduces complexity to linear

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

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

future cost estimation

Translating the Easy Part First?

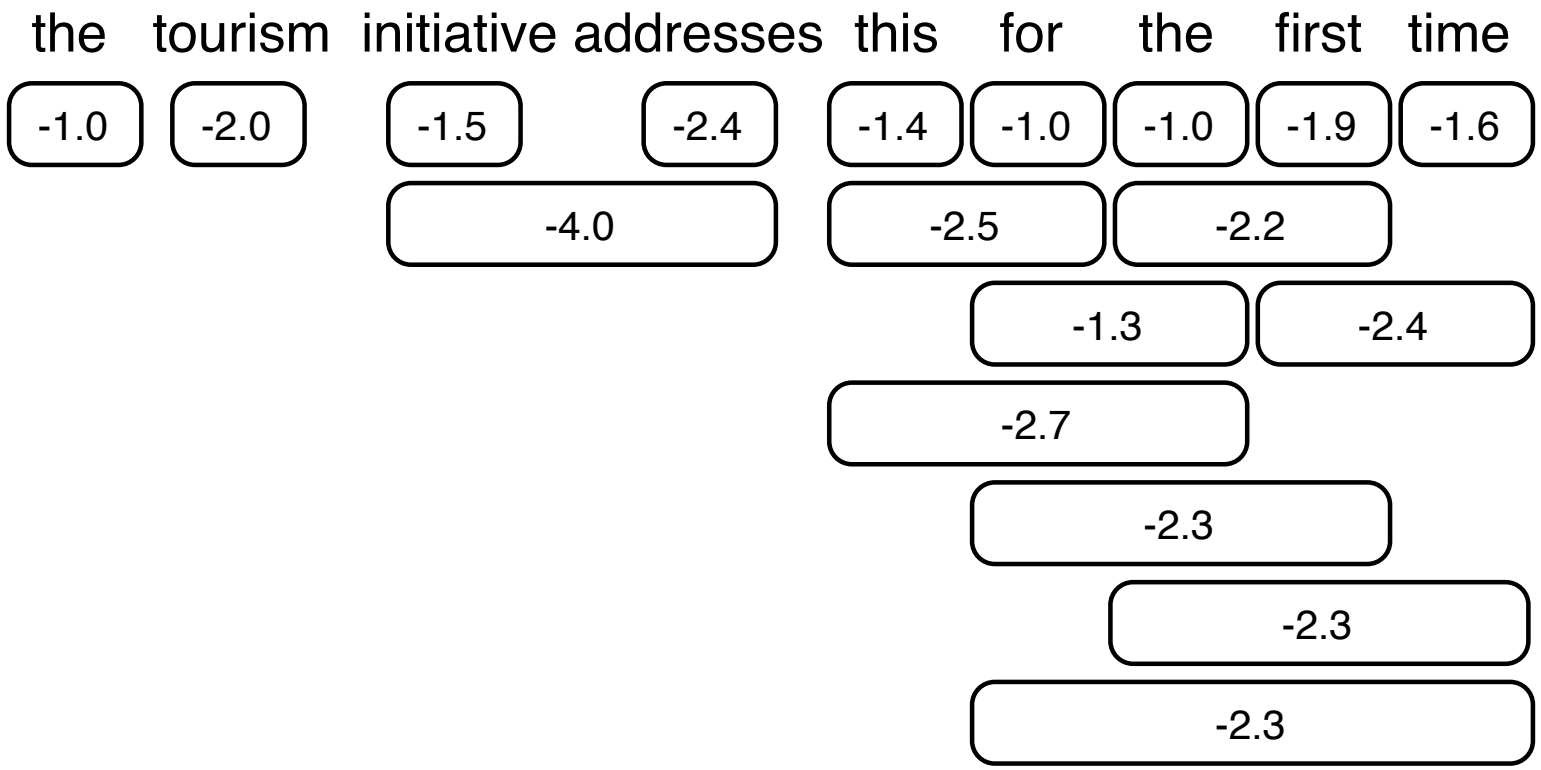


both hypotheses translate 3 words
worse hypothesis has better score

Estimating Future Cost

- 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

Cost Estimates from Translation Options



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

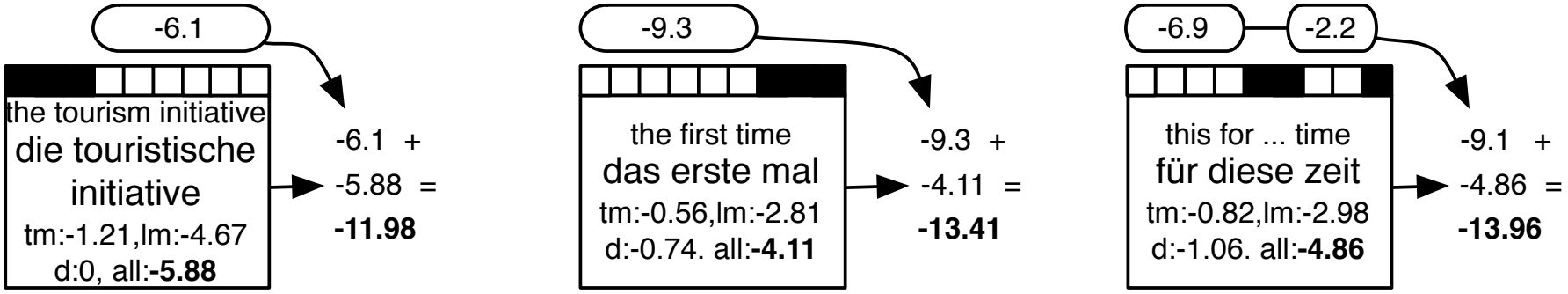
Cost Estimates for all Spans

- 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)

Combining Score and Future Cost



- 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

Stack Decoding Algorithm

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

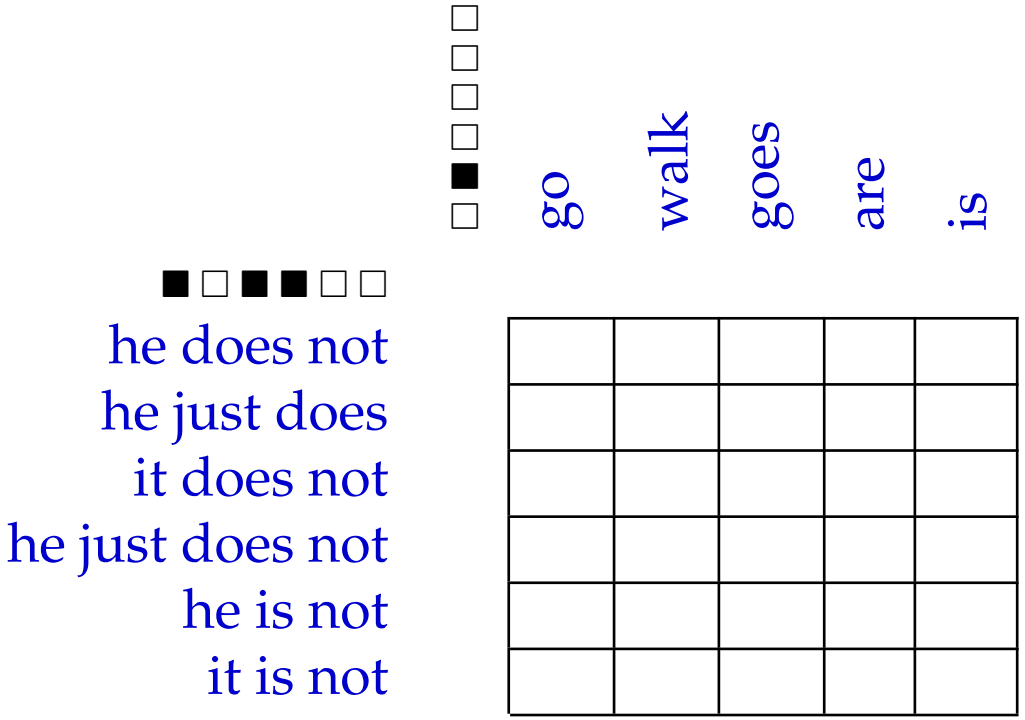
```
1:  place empty hypothesis into stack 0
2:  for all stacks 0...n − 1 do
3:    for all hypotheses in stack do
4:      for all translation options do
5:        if applicable then
6:          create new hypothesis
7:          place in stack
8:          recombine with existing hypothesis if possible
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
  - □ □ □ ■ □ □
  - □ □ □ □ ■ □
  - □ □ □ ■ ■ □
  - ...

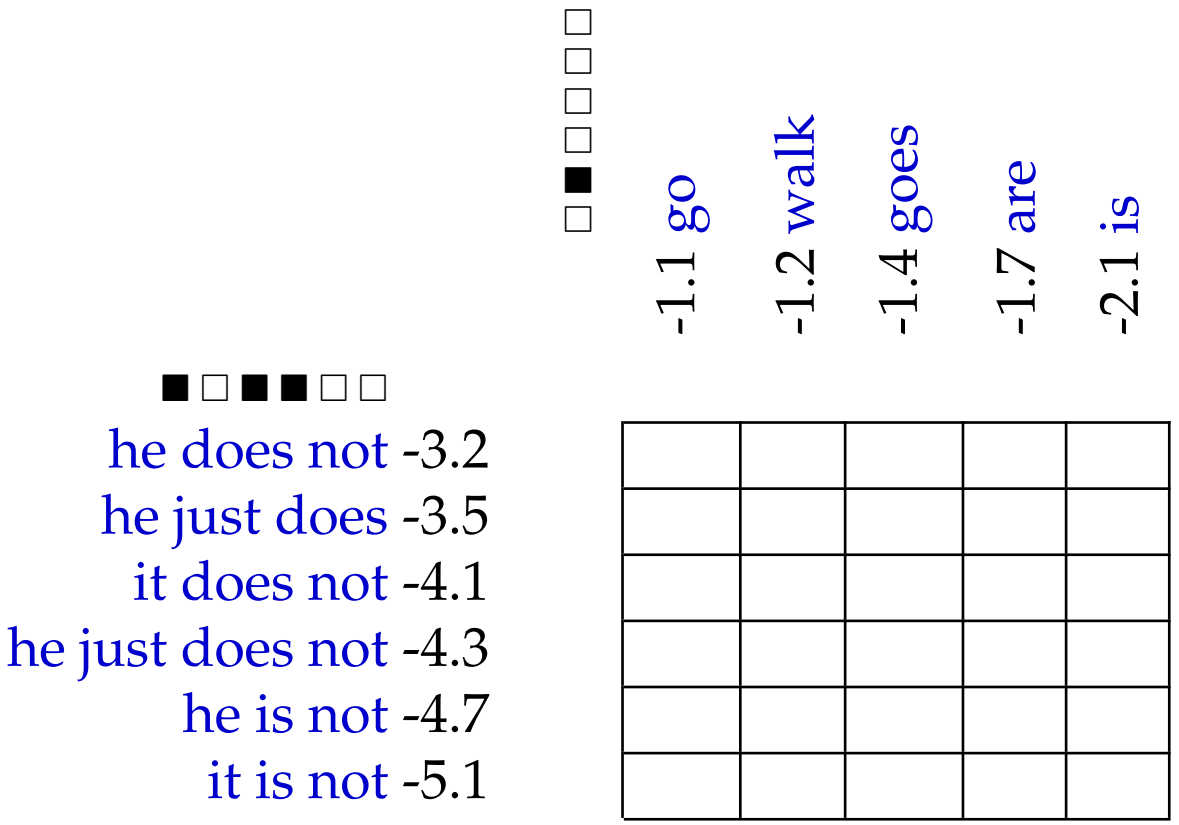
⇒ Loop over groups, check for applicability once for each pair of groups (not much gained so far)

All Hypotheses, All Options



- Example: group with 6 hypotheses, group with 5 translation options
- Should we really create all 6×5 of them?

Rank by Score



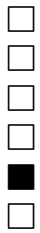
- Rank hypotheses by score so far
- Rank translation options by score estimate

Expected Score of New Hypothesis

	-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

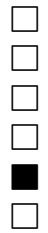
- If we want to save computational cost, we could decide to only compute some
- One way to do this: based on expected score

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

Cube Pruning (3)

	-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

questions?