

# Statistical Machine Translation: Decoding

Alexander Fraser

(Many slides from Aleš Tamchyna, Philipp Koehn)

`fraser@cis.uni-muenchen.de`

SS 2023

# Outline

- ▶ Which features are used in PBMT?
- ▶ How to compute the score of a translation?
- ▶ Search for the best translation: decoding.
  - ▶ Overview of the translation process.
  - ▶ Making decoding tractable: beam search.
- ▶ Other decoding algorithms.

# Log-Linear Model

We know how to score a full translation hypothesis:

$$P(e, a|f) \propto \exp \sum_i \lambda_i f_i(e, a, f)$$

$\lambda_i$  ... feature weights

$f_i$  ... feature functions

# Log-Linear Model: Features

Typical baseline feature set for PBMT:

- ▶ Phrase translation probability, both direct and inverse:
  - ▶  $P_{TM}(e|f)$
  - ▶  $P_{TM_{inv}}(f|e)$
- ▶ Lexical translation probability (direct and inverse):
  - ▶  $P_{lex}(e|f)$
  - ▶  $P_{lex_{inv}}(f|e)$
- ▶ Language model probability:
  - ▶  $P_{LM}(e)$
- ▶ Phrase penalty.
- ▶ Word penalty.
- ▶ Distortion penalty.

## Lexical Weights ( $P_{lex}$ )

The problem: many extracted phrases are rare.

(Esp. long phrases might only be seen once in the parallel corpus.)

## Lexical Weights ( $P_{lex}$ )

The problem: many extracted phrases are rare.

(Esp. long phrases might only be seen once in the parallel corpus.)

$P(\text{"modrý autobus přistál na Marsu"} | \text{"a blue bus lands on Mars"}) = 1$

$P(\text{"a blue bus lands on Mars"} | \text{"modrý autobus přistál na Marsu"}) = 1$

Is that a reliable probability estimate?

## Lexical Weights ( $P_{lex}$ )

The problem: many extracted phrases are rare.

(Esp. long phrases might only be seen once in the parallel corpus.)

$$P(" ; \text{distortion carried - over} | " ; \text{zkreslení} ) = 1$$

$$P(" ; \text{zkreslení} | " ; \text{distortion carried - over} ) = 1$$

Data from the “wild” are noisy. Word alignment contains errors.

“carried - over” is wrong.

This is a real phrase pair from a very good English-Czech SMT system.

Both  $P_{TM}(e|f)$  and  $P_{TM_{inv}}(f|e)$  say that this is a perfect translation.

## Lexical Weights ( $P_{lex}$ )

Decompose the phrase pair into word pairs. Look at the word-level translation probabilities.



## Lexical Weights ( $P_{lex}$ )

Decompose the phrase pair into word pairs. Look at the word-level translation probabilities.

Several possible definitions, e.g.:

## Lexical Weights ( $P_{lex}$ )

Decompose the phrase pair into word pairs. Look at the word-level translation probabilities.

Several possible definitions, e.g.:

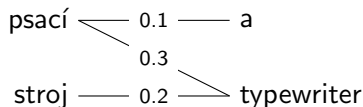
$$P_{lex}(\mathbf{e}|\mathbf{f}, a) = \prod_{j=1}^{l_e} \frac{1}{|\{(i,j) \in a\}|} \sum_{\forall (i,j) \in a} w(e_j, f_i)$$

## Lexical Weights ( $P_{lex}$ )

Decompose the phrase pair into word pairs. Look at the word-level translation probabilities.

Several possible definitions, e.g.:

$$P_{lex}(\mathbf{e}|\mathbf{f}, a) = \prod_{j=1}^{l_e} \frac{1}{|\{i|(i,j) \in a\}|} \sum_{\forall(i,j) \in a} w(e_j, f_i)$$

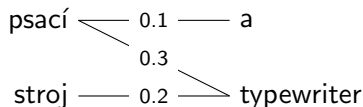


## Lexical Weights ( $P_{lex}$ )

Decompose the phrase pair into word pairs. Look at the word-level translation probabilities.

Several possible definitions, e.g.:

$$P_{lex}(\mathbf{e}|\mathbf{f}, a) = \prod_{j=1}^{l_e} \frac{1}{|\{i|(i,j) \in a\}|} \sum_{\forall(i,j) \in a} w(e_j, f_i)$$



$$P_{lex}(\text{"a typewriter"}|\text{"psací stroj"}) = \left[ \frac{1}{1} \cdot 0.1 \right] \cdot \left[ \frac{1}{2} \cdot (0.3 + 0.2) \right] = 0.025$$

## Word Penalty

Not all languages use the same number of words on average.

vidím problém ||| I can see a problem

## Word Penalty

Not all languages use the same number of words on average.

vidím problém ||| I can see a problem

- ▶ We want to control how many words are generated.

## Word Penalty

Not all languages use the same number of words on average.

vidím problém ||| I can see a problem

- ▶ We want to control how many words are generated.
- ▶ Word penalty simply adds 1 for each produced word in the translation.

# Word Penalty

Not all languages use the same number of words on average.

vidím problém ||| I can see a problem

- ▶ We want to control how many words are generated.
- ▶ Word penalty simply adds 1 for each produced word in the translation.
- ▶ Depending on the  $\lambda$  for word penalty, we will either generate shorter or longer outputs.



# Word Penalty

Not all languages use the same number of words on average.

vidím problém ||| I can see a problem

- ▶ We want to control how many words are generated.
- ▶ Word penalty simply adds 1 for each produced word in the translation.
- ▶ Depending on the  $\lambda$  for word penalty, we will either generate shorter or longer outputs.

$$\hat{e} = \arg \max_{e,a} \sum_i \lambda_i f_i(e, a, f)$$

## Phrase Penalty

- ▶ Add 1 for each produced *phrase* in the translation.

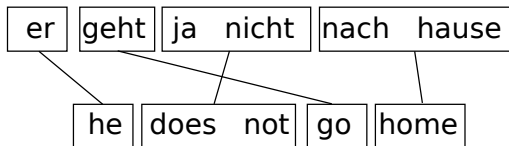
## Phrase Penalty

- ▶ Add 1 for each produced *phrase* in the translation.
- ▶ Varying the  $\lambda$  for phrase penalty can lead to more literal (word-by-word) translations (made from a lot of short phrases) or to more idiomatic outputs (use fewer, longer phrases – if available).

# Distortion Penalty

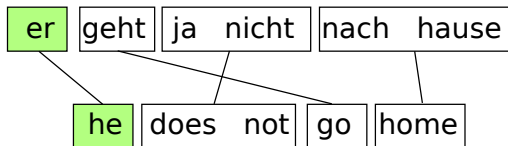
- ▶ The simplest way to capture **phrase reordering**.
- ▶ Can be sufficient for some language pairs
- ▶ Several possible definitions!
- ▶ Definition I tend to use:
  - ▶ Distance between the end of the previous phrase (on the source side) and the beginning of the current phrase.

## How to Score a Translation?



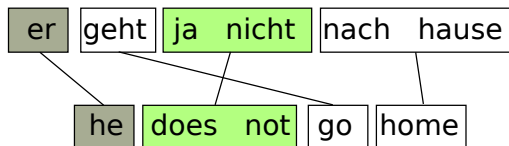
$$\text{score}(e|f) = 0$$

## How to Score a Translation?



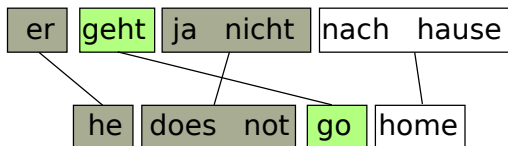
$$\begin{aligned} \text{score}(e|f)_+ &= \lambda_{TM} \cdot \log P_{TM}(\text{"he"} | \text{"er"}) \\ &+ \lambda_{TM_{inv}} \cdot \log P_{TM_{inv}}(\text{"er"} | \text{"he"}) \\ &+ \lambda_{lex} \cdot \log P_{lex}(\text{"he"} | \text{"er"}) \\ &+ \lambda_{lex_{inv}} \cdot \log P_{lex_{inv}}(\text{"er"} | \text{"he"}) \\ &+ \lambda_D \cdot 0 \\ &+ \lambda_{WP} \cdot 1 \\ &+ \lambda_{PP} \cdot 1 \\ &+ \lambda_{LM} \cdot \log P_{LM}(\text{"he"} | \text{"<S>"}) \end{aligned}$$

## How to Score a Translation?



$$\begin{aligned} \text{score}(e|f)_+ &= \lambda_{TM} \cdot \log P_{TM}(\text{"does not"} | \text{"ja nicht"}) \\ &+ \lambda_{TM_{inv}} \cdot \log P_{TM_{inv}}(\text{"ja nicht"} | \text{"does not"}) \\ &+ \lambda_{lex} \cdot \log P_{lex}(\text{"does not"} | \text{"ja nicht"}) \\ &+ \lambda_{lex_{inv}} \cdot \log P_{lex_{inv}}(\text{"ja nicht"} | \text{"does not"}) \\ &+ \lambda_D \cdot 1 \\ &+ \lambda_{WP} \cdot 2 \\ &+ \lambda_{PP} \cdot 1 \\ &+ \lambda_{LM} \cdot \log P_{LM}(\text{"does not"} | \text{"<S>he"}) \end{aligned}$$

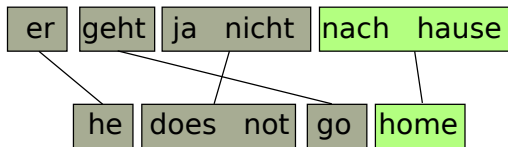
## How to Score a Translation?



$$\begin{aligned} \text{score}(e|f)_+ &= \lambda_{TM} \cdot \log P_{TM}(\text{"go"} | \text{"geht"}) \\ &\quad + \lambda_{TM_{inv}} \cdot \log P_{TM_{inv}}(\text{"geht"} | \text{"go"}) \\ &\quad + \lambda_{lex} \cdot \log P_{lex}(\text{"go"} | \text{"geht"}) \\ &\quad + \lambda_{lex_{inv}} \cdot \log P_{lex_{inv}}(\text{"geht"} | \text{"go"}) \\ &\quad + \lambda_D \cdot 3 \\ &\quad + \lambda_{WP} \cdot 1 \\ &\quad + \lambda_{PP} \cdot 1 \\ &\quad + \lambda_{LM} \cdot \log P_{LM}(\text{"go"} | \text{"does not"}) \end{aligned}$$

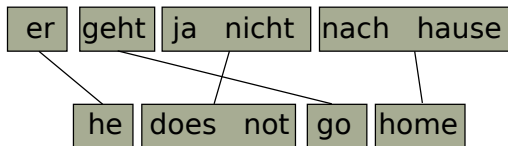


## How to Score a Translation?



$$\text{score}(e|f)_+ = \dots$$

## How to Score a Translation?



$$\text{score}(e|f)_+ = \dots$$

# 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

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

- Two types of error
  - the most probable translation is bad  $\rightarrow$  fix the model
  - search does not find the most probably translation  $\rightarrow$  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

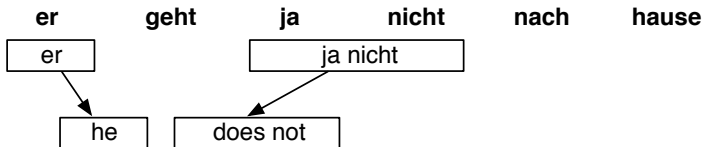
- Task: translate this sentence from German into English



- Pick phrase in input, translate

# Translation Process

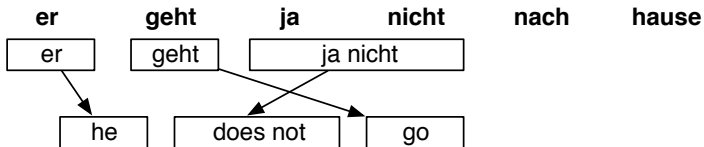
- Task: translate this sentence from German into English



- 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

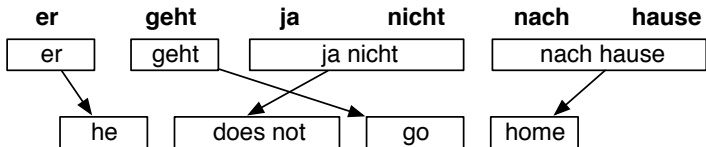
- Task: translate this sentence from German into English



- Pick phrase in input, translate

# Translation Process

- Task: translate this sentence from German into English



- Pick phrase in input, translate

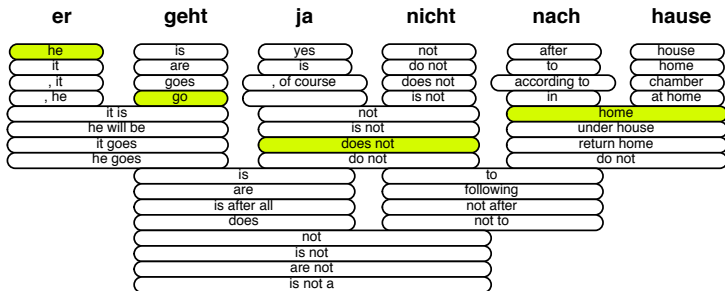


# Translation Options

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

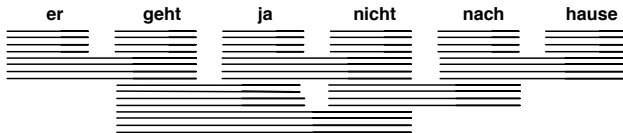
# Translation Options



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

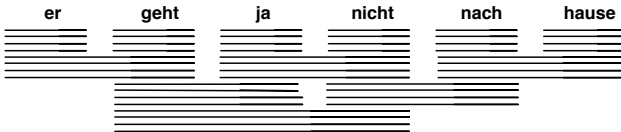
→ Search problem solved by heuristic beam search

# Decoding: Precompute Translation Options



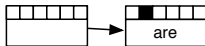
consult phrase translation table for all input phrases

## Decoding: Start with Initial Hypothesis



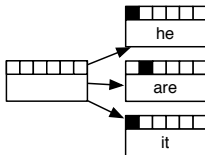
initial hypothesis: no input words covered, no output produced

# Decoding: Hypothesis Expansion



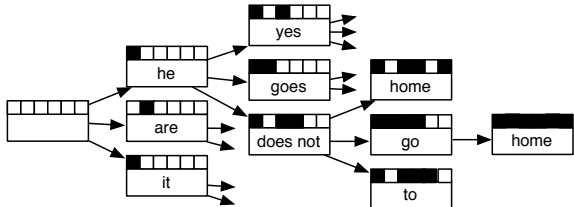
pick any translation option, create new hypothesis

# Decoding: Hypothesis Expansion



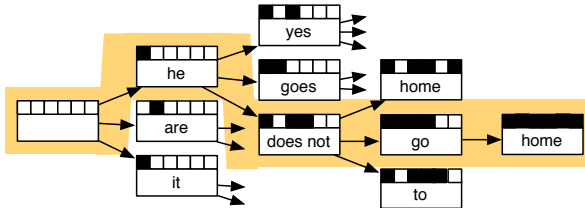
create hypotheses for all other translation options

# Decoding: Hypothesis Expansion



also create hypotheses from created partial hypothesis

# Decoding: Find Best Path



backtrack from highest scoring complete hypothesis

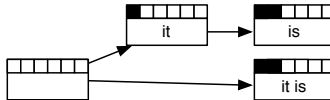


# Computational Complexity

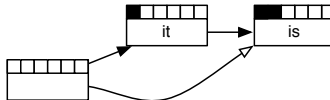
- The suggested process creates exponential number of hypothesis
- Machine translation decoding is NP-complete
- Reduction of search space:
  - recombination (risk-free)
  - pruning (risky)

# Recombination

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

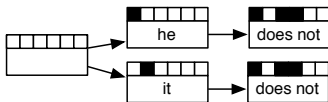


- Worse hypothesis is dropped

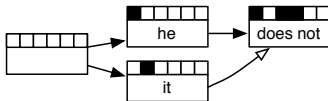


# Recombination

- 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



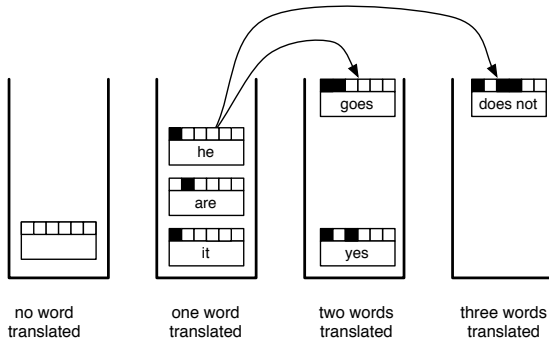
# Restrictions on Recombination

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

- 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  $0 \dots n - 1$  **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

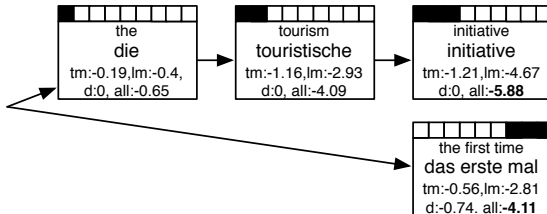
- Limiting reordering to maximum reordering distance
- Typical reordering distance 5–8 words
  - depending on language pair
  - larger reordering limit hurts translation quality
- Reduces complexity to linear

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

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

# Translating the Easy Part First?

the tourism initiative addresses this for the first time

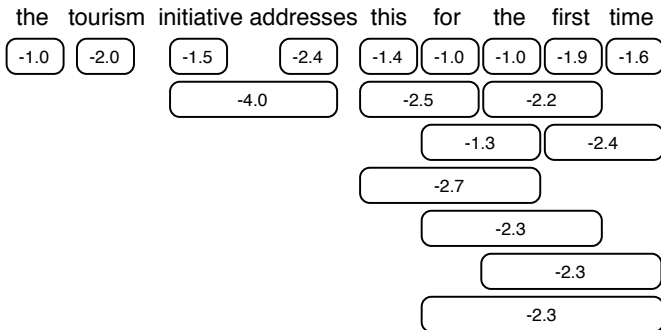


both hypotheses translate 3 words  
worse hypothesis has better score

# 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  
→ 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)

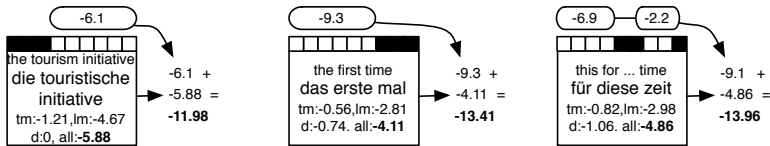
## Cost Estimates for all Spans

- 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

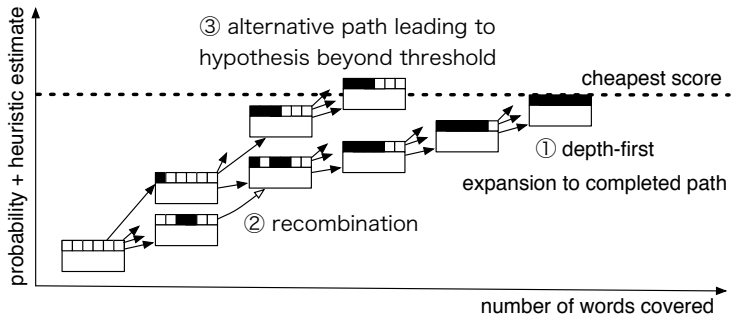


- Hypothesis score and future cost estimate are combined for pruning
  - left hypothesis starts with hard part: **the tourism initiative**  
score: -5.88, future cost: -6.1 → total cost -11.98
  - middle hypothesis starts with easiest part: **the first time**  
score: -4.11, future cost: -9.3 → total cost -13.41
  - right hypothesis picks easy parts: **this for ... time**  
score: -4.86, future cost: -9.1 → total cost -13.96

# Other Decoding Algorithms

- A\* search
- Greedy hill-climbing
- Using finite state transducers (standard toolkits)

# A\* Search



- Uses *admissible* future cost heuristic: never overestimates cost
- Translation agenda: create hypothesis with lowest score + heuristic cost
- Done, when complete hypothesis created



# Greedy Hill-Climbing

- 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

## Finite-state transducers

- ▶ It is also possible to output a pruned search graph to an external finite-state transducer package.
- ▶ This then carries out the search, but I will omit the details of this.
- ▶ Allows efficient search of this (pruned) graph
- ▶ Can be useful for rescoring the hypotheses using models that are difficult to implement directly in, e.g., Moses.

# Summary

- ▶ Log-linear model: standard features in PBMT.
- ▶ Computing the score of a translation.
- ▶ Overview of the translation process.
- ▶ Beam search algorithm.
  - ▶ Hypothesis recombination.
  - ▶ Pruning.
  - ▶ Limiting distortion.
  - ▶ Future cost.
- ▶ Other decoding algorithms.

Questions?

Thank you for your attention.