INTRODUCTION TO STATISTICAL MACHINE TRANSLATION

Hervé Blanchon

Laboratoire LIG
Équipe GETALP
herve.blanchon@univ-grenoble-alpes.fr

Foreword

- The content of this presentation is based on
  - Philipp Koehn
    - Statistical Machine Translation, Philipp Koehn
      Cambridge University Press, 2010, 433 p.
    - several other P. Koehn's tutorials on SMT
  - Laurent Besacier & others
    - language model
- Topics

Addressed	Not addressed
word-based models phrase-based models	integrating linguistic information tree-based models (hierarchical models)

- Further readings:
  - http://www.statmt.org

Outline

- Introduction
- Language models
- Word-Based Models
- Phrase-Based Models
- Decoding
- Evaluation
  - at the end of the module

INTRODUCTION

What is SMT about?

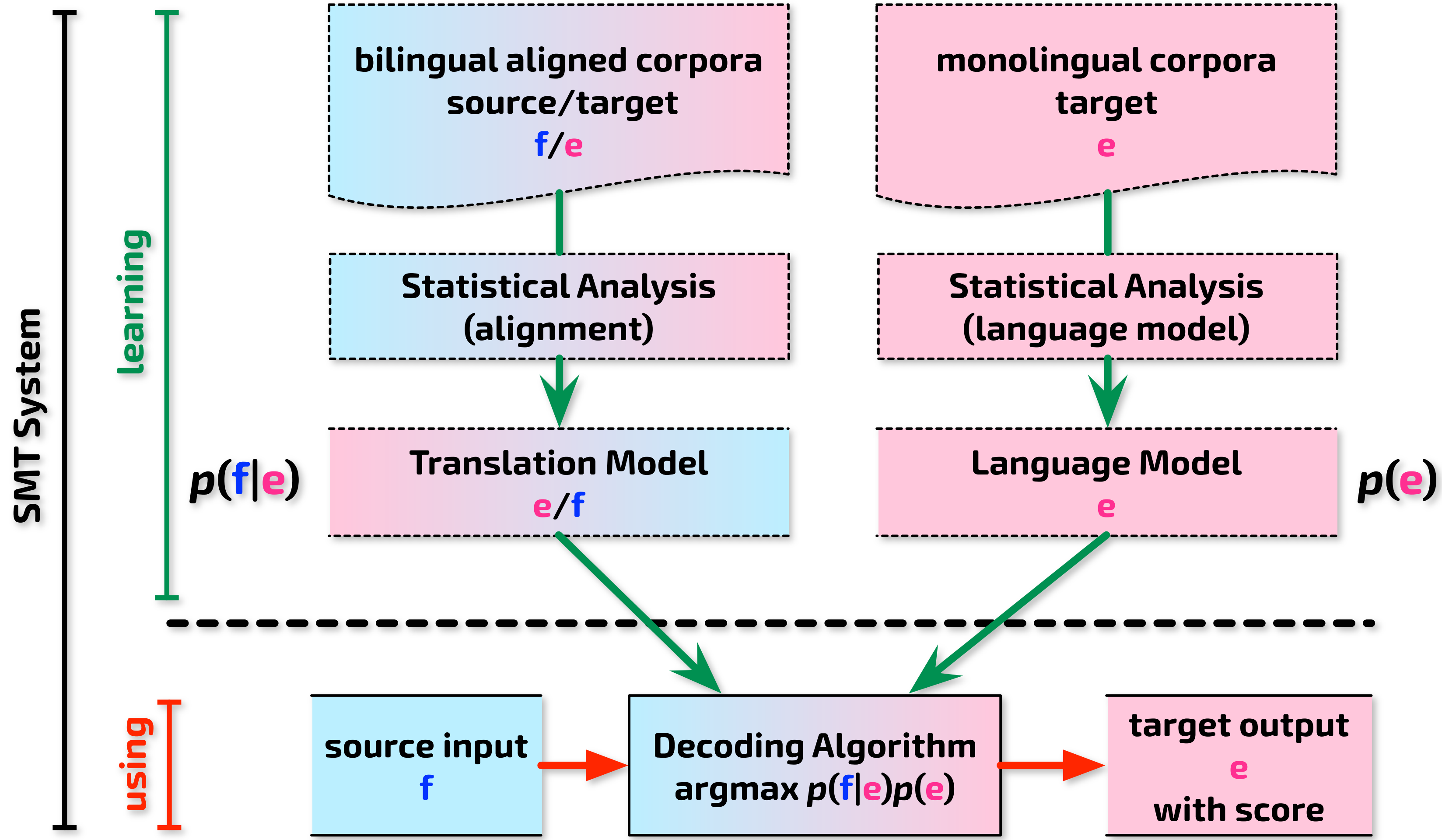
- Learning an MT system to translate from $\boldsymbol{f}$ (source) to $\boldsymbol{e}$ (target) data using statistics
  - $p(\boldsymbol{e}|\boldsymbol{f})$

or *(Bayes)*

  - $p(\boldsymbol{e}|\boldsymbol{f}) \propto p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e})$
- What kind of data?
  - parallel bilingual corpora (source, target)
    - usually $\boldsymbol{f}$ (Foreign) for the source, $\boldsymbol{e}$ (English) for the target
  - monolingual corpora for the target
    - usually $\boldsymbol{e}$ (English) for the target
- For what purpose?
  - parallel bilingual corpora: learn how $\boldsymbol{e}$ translate into $\boldsymbol{f}$
    - a **translation model**
  - monolingual corpora: learn if an utterance u is acceptable in $\boldsymbol{e}$
    - a **language model**

How is a SMT from f to e built and used?

- Step 1: learning the models
- Step 2: using the models to translate



What Kind of Alignment?

- Word alignments
- Phrase alignments
  - a phrase is a set of consecutive words
    - a segment, a chunk

Let's see!

LANGUAGE MODELS

Language Models

- Can answer the question
  - What is the probability that this string of words is correct?
    - "The cat is dead" => very good ($\approx$ 1.0)
    - "The cat is talkative" => quite poor ($\approx$?)
    - "Is the crowned cat" => very poor ($\approx$ 0.0)
- Use
  - Automatic Speech Recognition
  - Machine translation
  - Language recognition
  - Optical Character Recognition
  - ....

Language Models

- Given a string of words $W = w_1 w_2 w_3 w_4 \ldots w_n$
  - *chain rule*

$$p(w_1, w_2, w_3, \ldots, w_n) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \ldots p(w_n|w_1, w_2, \ldots, w_{n-1})$$

- Markov assumption (use history of limited length)
  - Only the k preceding words belong to the history
  - Model of order k
- Example: a model of order 1 (bigram)

$$p(w_1, w_2) = p(w_1)p(w_2|w_1)$$

Estimate the n-grams Probabilities

- Collect frequencies of words and word sequences in very large corpus
  - Several million words
- Using “chain rule”:

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

Model size?

- For each n-gram, one must store a probability
- If we assume a vocabulary of 20,000 words

Model	Max number of parameters
0th order (unigram)	$20{,}000$
1st order (bigram)	$20{,}000^2 = 400$ million
2nd order (trigram)	$20{,}000^3 = 8$ trillion
3rd order (4-gram)	$20{,}000^4 = 160$ quadrillion

- Trigram LM are mostly used

Practical example

- From a corpus of 275 million words written in English
  - newspapers such as “Wall Street Journal”

Model	Number of n-grams
1-gram	716,706
2-gram	12,537,755
3-gram	22,174,483

Quality of a language model

- Entropy of a sequence $w_1, w_2, \dots, w_n$

$$H(w_1, w_2, \dots, w_n) = -\sum_{w_1 \dots w_n \in \Sigma^n} p(w_1 \dots w_n) \log_2 p(w_1 \dots w_n)$$

- per word Entropy of a sequence $w_1, w_2, \dots, w_n$

$$\frac{1}{n} H(w_1, w_2, \dots, w_n) = -\frac{1}{n} \sum_{w_1 \dots w_n \in \Sigma^n} p(w_1 \dots w_n) \log_2 p(w_1 \dots w_n)$$

- Entropy of a language $L = \{w_1, w_2, \dots, w_n | 0 < n < \infty\}$

$$H(L) = -\lim_{n \to \infty} \frac{1}{n} H(w_1 \dots w_n)$$

- Perplexity

$$perplexity(L) = 2^{H(L)}$$

  - A language model m is better than m' if it assign lower perplexity to the test corpus $w_1 \dots w_n$

Example: 1-gram

- Training set [14 tokens ($^1/_{14} \cong 0.0714$)]
  - *there is a big house*
  - *i buy a house*
  - *they buy the new house*
- Model

$p(there) = 0.0714$	$p(is) = 0.0714$	$p(a) = 0.1429$
$p(big) = 0.0714$	$p(house) = 0.2143$	$p(i) = 0.0714$
$p(buy) = 0.1429$	$p(they) = 0.0714$	$p(the) = 0.0714$
$p(new) = 0.0714$		

- Test sentence S: *they buy a big house*
  - $p(S) = \underbrace{0.0714}_{they} \times \underbrace{0.1429}_{buy} \times \underbrace{0.0714}_{a} \times \underbrace{0.1429}_{big} \times \underbrace{0.2143}_{house} = 0.0000231$

Example: 2-gram

- Training set
  - *there is **a** big house*
  - ***i** buy **a** house*
  - *they buy **the** new house*
- Model

$p(big \mid a) = 0.5$	$p(is \mid there) = 1$	$p(buy \mid they) = 1$
$p(house \mid a) = 0.5$	$p(buy \mid i) = 1$	$p(a \mid buy) = 0.5$
$p(new \mid the) = 1$	$p(house \mid big) = 1$	$p(the \mid buy) = 0.5$
$p(a \mid is) = 1$	$p(house \mid new) = 1$	$p(they \mid <s>) = 0.333$
$p(there \mid <s>) = 0.333$	$p(i \mid <s>) = 0.333$	

- Test sentence $S1$: *they buy a big house*
  - $p(S) = \underbrace{0.333}_{\text{<s> they}} \times \underbrace{1}_{\text{they buy}} \times \underbrace{0.5}_{\text{buy a}} \times \underbrace{0.5}_{\text{a big}} \times \underbrace{1}_{\text{big house}} = 0.0833$

Problem of Unknown Events

- Training set
  - *there is **a** big house*
  - ***i** buy **a** house*
  - *they buy **the** new house*

- Let sentence $S2$
  - *they buy a new house*
- The bigram "$a\ new$" has never been seen
  - $p(S2) = 0$?!
  - But the sentence is correct

Problem of Unknown Events

- Two types of "zeroes"
  - Unknown words
    - Problem dealt with a label "$UNKNOWN$"
    - The probability $p(UNKNOWN)$ is estimated
      - Tend to over-estimate the probability
        - Smoothing mechanisms
  - Unknown N-grams
    - Smoothing by giving them a low probability (but not zero!)
    - Fall back (backoff) to a lower-order n-gram
  - Give a non-zero probability to un-observed events
    - This is not a maximum likelihood estimate

More on that topic with Laurent Besacier

WORD-BASED MODELS

Lexical Translation

- How to translate a word → look up in dictionary
  - Haus — house, building, home, household, shell
- Multiple translations
  - some more frequent than others
  - for instance: house, and building most common
  - special cases: Haus of a snail is its shell

Note: In all lectures, we translate from a foreign language into English

Collect Statistics

- Look at a parallel corpus
  - here German text along with English translation

Translations of haus	Count
house	8000
building	1600
home	200
household	150
shell	50

Estimate Translation Probabilities

- Lexical translation probability distribution p_f given a foreign word f for each English translation e
  - $\sum_e p_f(e) = 1$
  - $\forall e : 0 \leq p_f(e) \leq 1$
- Maximum likelihood estimation
  - in our case divide each *count* by *sum of counts*
  - $$p_f(e) = \begin{cases} 0.8 & \text{if } e = \text{house} \\ 0.16 & \text{if } e = \text{building} \\ 0.02 & \text{if } e = \text{home} \\ 0.015 & \text{if } e = \text{household} \\ 0.005 & \text{if } e = \text{shell} \end{cases}$$

Alignment

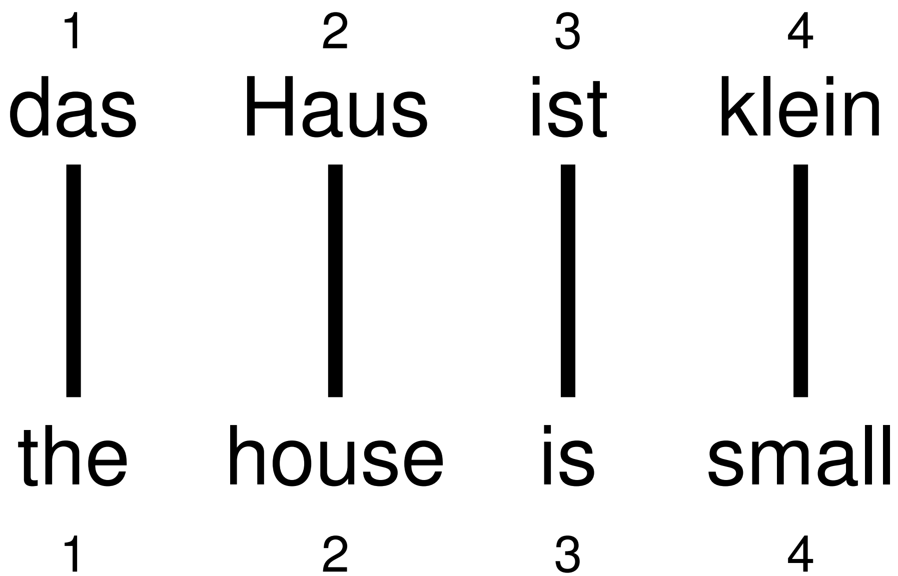
- In a parallel text (or when we translate), we align words in one language with the words in the other

1	2	3	4
das	Haus	ist	klein
|	|	|	|
the	house	is	small
1	2	3	4

- Word positions are numbered 1–4

Alignment function

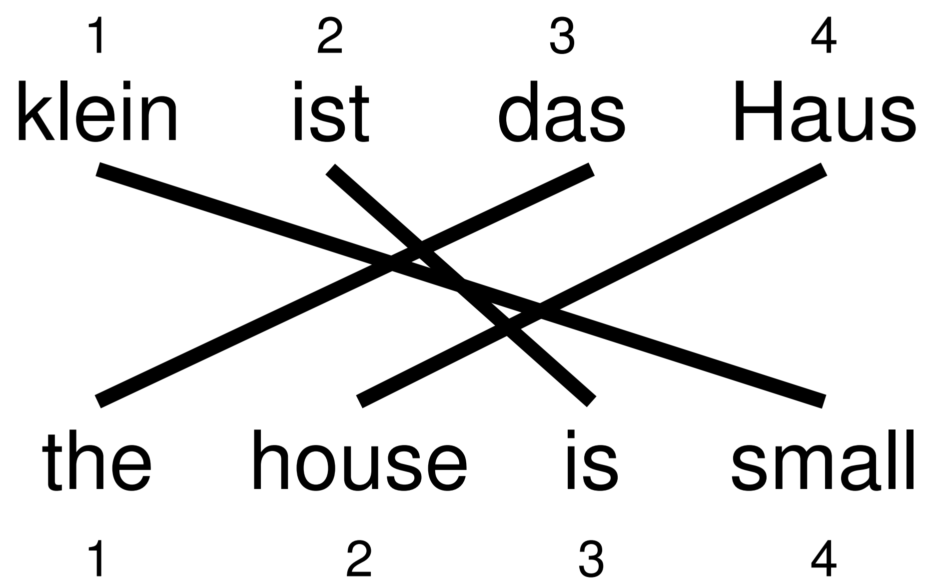
- Formalizing alignment with an alignment function
  - Mapping an English target word at position i to a German source word at position j
  - with a function $a: i \rightarrow j$
- Example



  - a: {1 → 1, 2 → 2, 3 → 3, 4 → 4}

Reordering

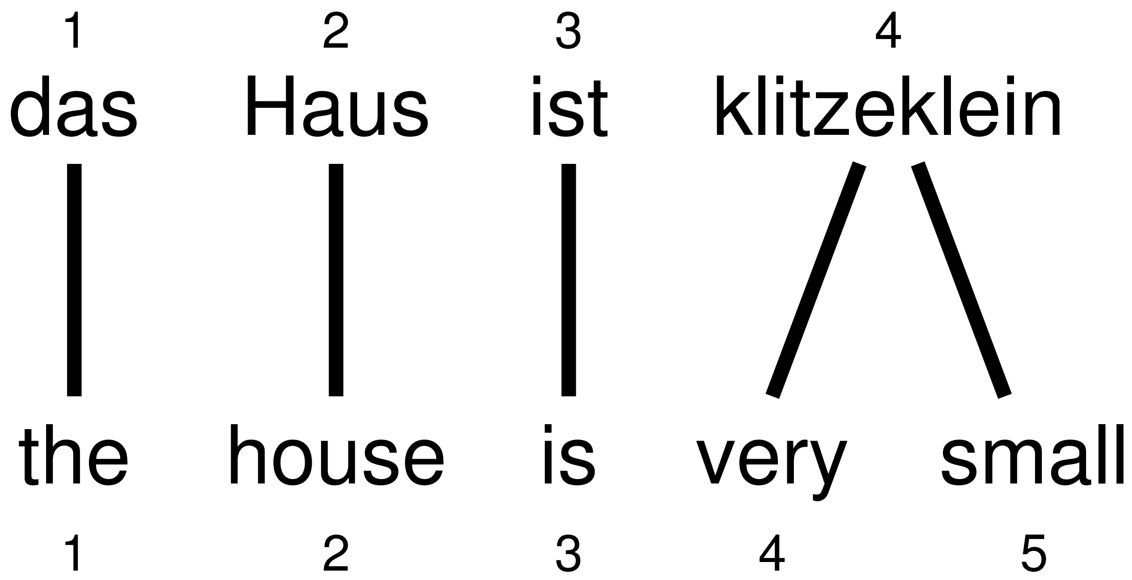
- Words may be reordered during translation
- Example



- $a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$

One-to-Many Translation

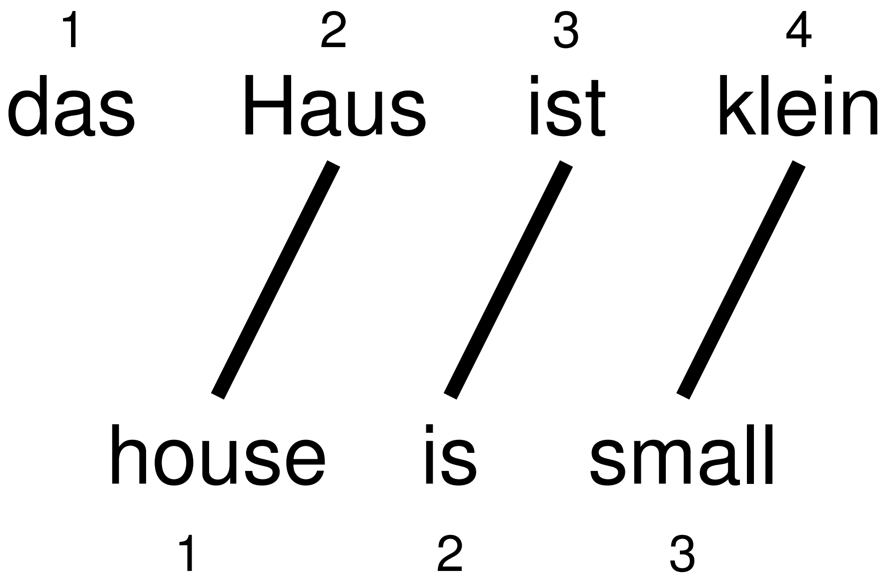
- A source word may translate into multiple target words
- Example



- $a : \{1 \rightarrow 1, 2 \rightarrow 2, 3 \rightarrow 3, 4 \rightarrow 4, 5 \rightarrow 4\}$

Dropping Words

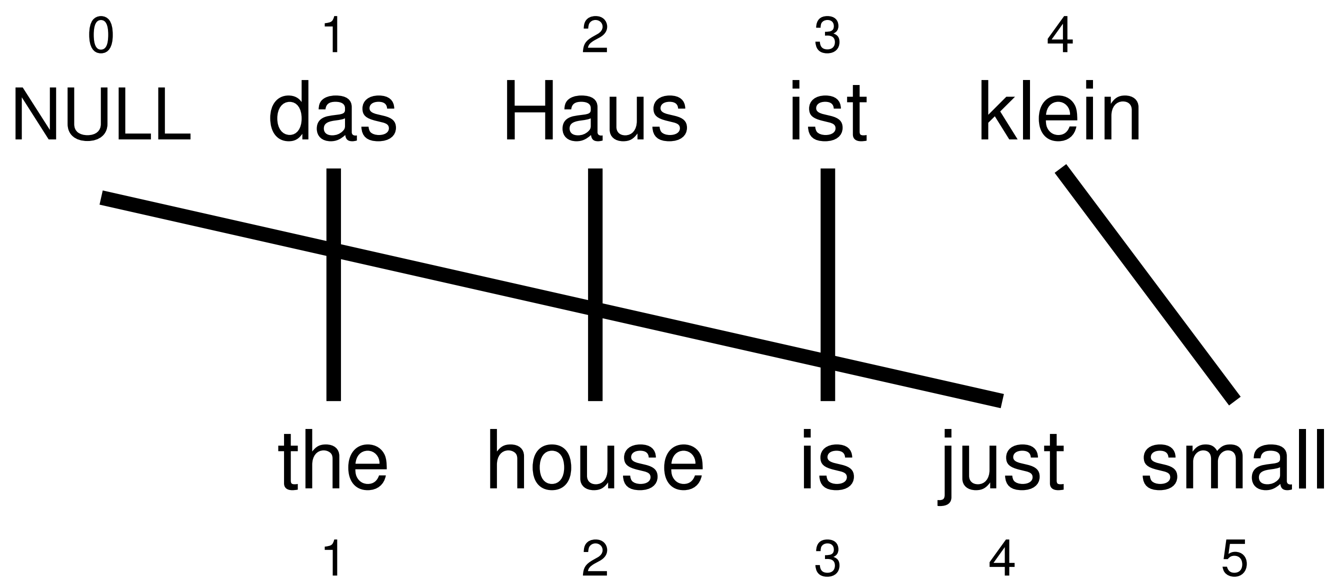
- Words may be dropped when translated
- Example
  - dropping the German article das



  - a : $\{1 \rightarrow 2, 2 \rightarrow 3, 3 \rightarrow 4\}$

Inserting Words

- Words may be added during translation
- Example
  - The English just does not have an equivalent in German
  - We still need to map it to something: special null token



  - a : {1 → 1, 2 → 2, 3 → 3, 4 → 0, 5 → 4}

IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation
- Translation probability (*a joint probability*)
  - from a foreign sentence $\boldsymbol{f} = (f_1, \dots, f_{l_f})$ of length l_f
  - to an English sentence $\boldsymbol{e} = (e_1, \dots, e_{l_e})$ of length l_e
  - with $a\colon j \to i$ an alignment function of each English word e_j to a foreign word f_i
  - $p(\boldsymbol{e}, a|\boldsymbol{f}) = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t\left(e_j | f_{a(j)}\right)$
- The parameter ϵ is a normalization constant

IBM Model 1 – Breakdown of the formula

- The formula
  - $p(\boldsymbol{e}, a|\boldsymbol{f}) = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$
- can be broken down as follow:
  - core: product over the lexical probability for all l_e generated output words e_j
  - fraction before product: normalization
    - adding the NULL token: $l_f + 1$ input words
    - $(l_f + 1)^{l_e}$ alignments that map $l_f + 1$ input words into l_e output words
    - ϵ: normalization constant so that $p(\boldsymbol{e}, a|\boldsymbol{f})$ is a proper probability distribution (the probability of all possible translations $\boldsymbol{e}$ and alignments a sum up to 1: $\sum_{\boldsymbol{e},a} p(\boldsymbol{e}, a|\boldsymbol{f}) = 1$)

IBM Model 1 – Example

- The probability of f = das Haus ist klein being translated into e = the house is small given:

a

1	2	3	4
das	Haus	ist	klein
the	house	is	small
1	2	3	4

das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

- is
  - $p(e,a|f) = \frac{\epsilon}{5^4} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})$
  - $p(e,a|f) = \frac{\epsilon}{5^4} \times 0.7 \times 0.8 \times 0.8 \times 0.4$
  - $p(e,a|f) = 0.0029\epsilon$

Learning Lexical Tanslation Models

- Need to estimate the lexical translation probabilities $t(e|f)$ from a parallel corpus
  - ... but we do not have the alignments
- Chicken and egg problem
  - if we had the alignments
    - $\longrightarrow$ we could estimate the parameters of our generative model
  - if we had the parameters
    - $\longrightarrow$ we could estimate the alignments

Learning Lexical Translation Model (EM)

- Solution : Expectation Maximization (EM)
- Expectation Maximization in a nutshell
  - an iterative learning method

1. initialize model parameters (e.g. uniform)
2. assign probabilities to the missing data
3. estimate model parameters from completed data
4. iterate steps 2–3 until convergence

Expectation

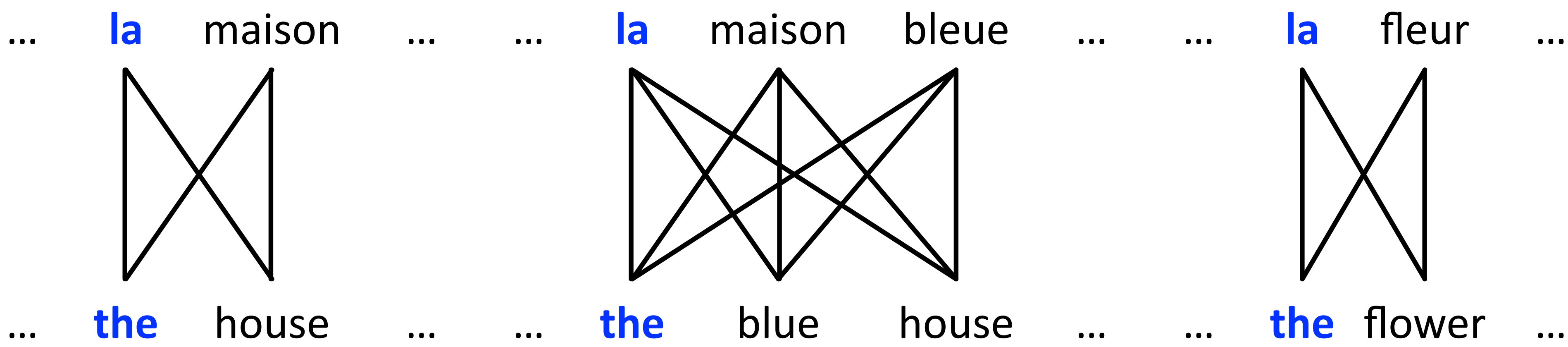
- Expectation of a random variables X
  - a set of values $x_1, x_2, \ldots, x_n$
  - a probability $p(x_i), \forall i \in [1..n]$

$$E(X) = \sum_{i=1}^{n} p(x_i) x_i$$

- Informal definition
  - the expected value of a random variable is intuitively the long-run average value of repetitions of the experiment it represents
- Example: a dice
  - 6 equiprobable (1/6) resting positions (1, 2, … 6)
  - $E(dice) = \frac{1}{6}\times 1+\frac{1}{6}\times 2+\frac{1}{6}\times 3\frac{1}{+6}\times 4+\frac{1}{6}\times 5+\frac{1}{6}\times 6=3.5$

EM Algorithm

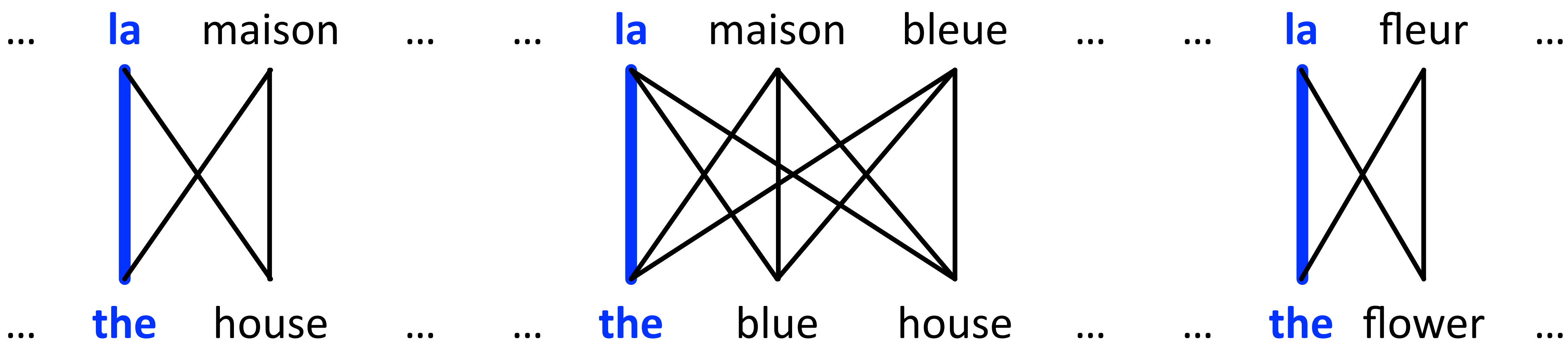
- Initial step:
  - all alignments are equally likely
- Example of data



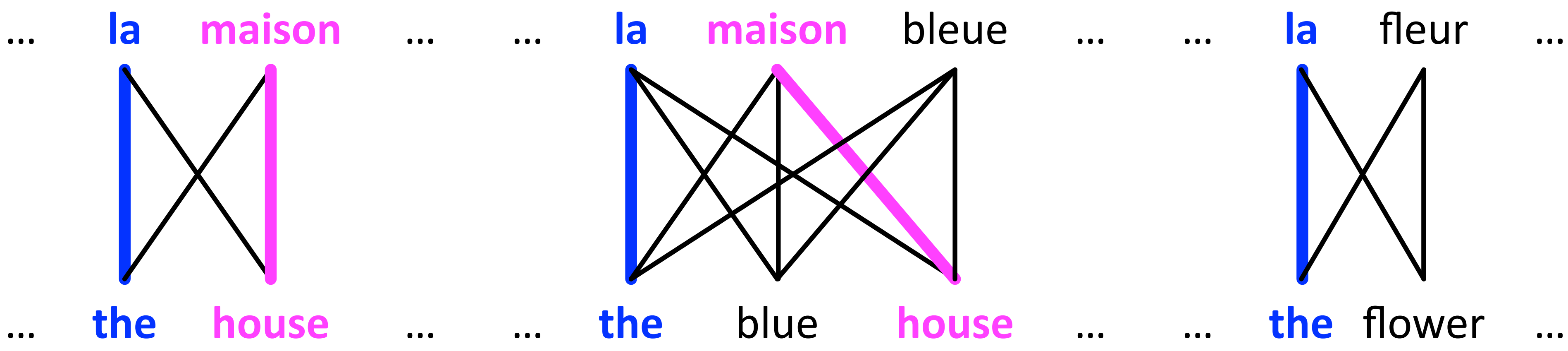
- Model learns
  - e.g., **la** is often aligned with **the**

EM Algorithm

- After initial step : la and the more likely

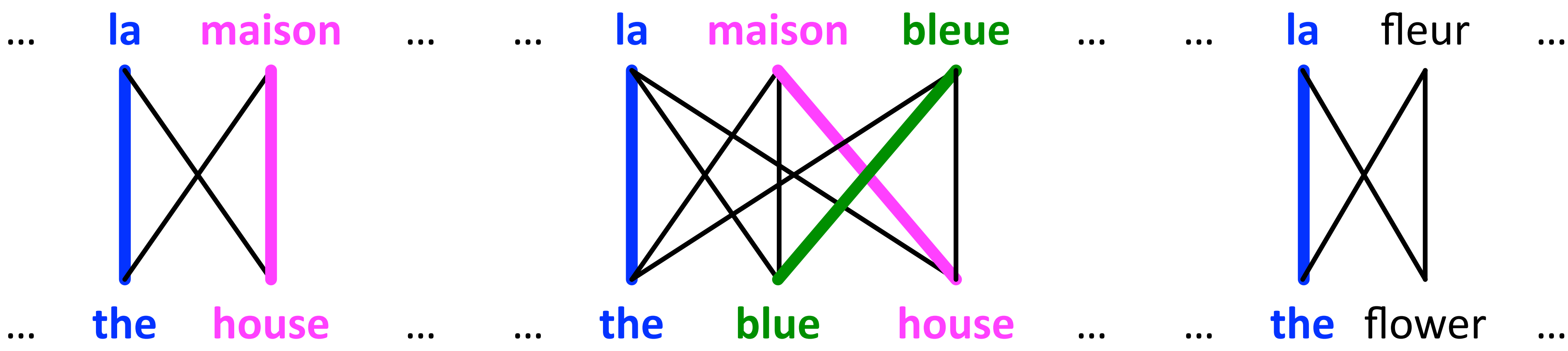


- After next step : maison and house more likely

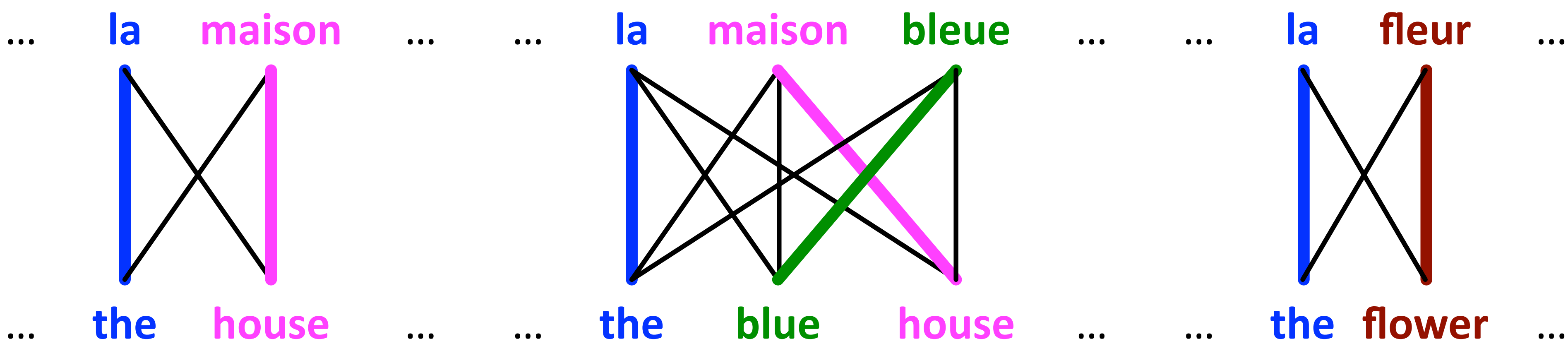


EM Algorithm

- After next step : bleue and blue more likely

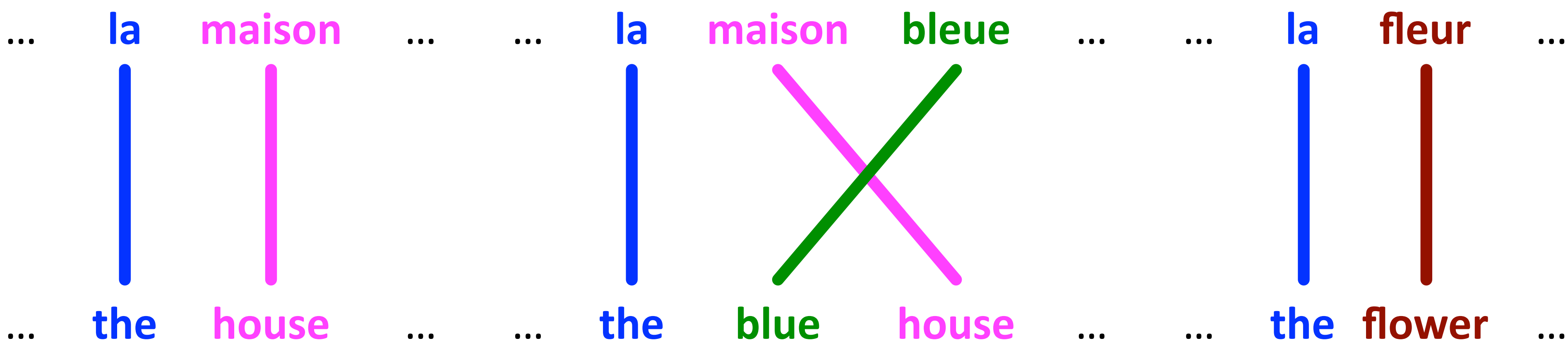


- After next step : fleur and flower more likely



EM Algorithm

- Final step: convergence



- Hidden inherent structure revealed by EM
  - parameter estimation from the aligned corpus
- Probabilities
  - p(la|the) = 0.453   p(fleur|flower) = 0.334
  - p(maison|house) = 0.876   p(bleu|blue) = 0.563

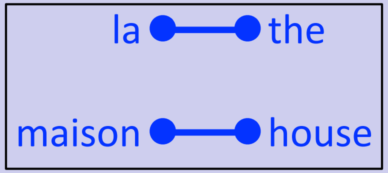
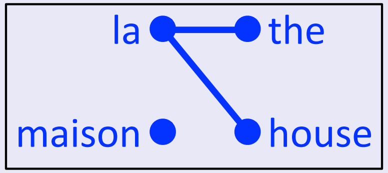
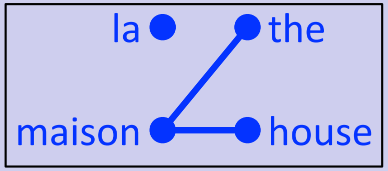
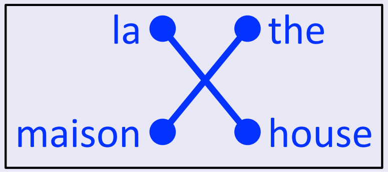
IBM Model 1 and EM

- EM Algorithm consists of two steps
- Expectation-Step: Apply model to the data
  - parts of the model are hidden (here: alignments)
  - using the model, assign probabilities to possible values
    - = probabilities of alignments
- Maximization-Step: Estimate model from data
  - take assigned values as fact
  - collect counts (weighted by probabilities)
  - estimate model from counts
    - = count collection
- Iterate these steps until convergence

IBM Model 1 and EM

Probabilities

- $p(\text{the}|\text{la}) = 0.7, p(\text{house}|\text{la}) = 0.05, p(\text{the}|\text{maison}) = 0.1, p(\text{house}|\text{maison}) = 0.8$

Alignments	$\boldsymbol{p(e,f\|a)}$	$p(a\|e,f)$
	$= p(\text{the}\|\text{la})\times p(\text{house}\|\text{maison})$ $= 0.7\times 0.8$ $= 0.56$	$= 0.56/0.68$ $= 0.824$
	$= p(\text{the}\|\text{la})\times p(\text{house}\|\text{la})$ $= 0.7\times 0.05$ $= 0.035$	$= 0.035/0.68$ $= 0.052$
	$= p(\text{the}\|\text{maison})\times p(\text{house}\|\text{maison})$ $= 0.1\times 0.8$ $= 0.08$	$= 0.08/0.68$ $= 0.118$
	$= p(\text{house}\|\text{la})\times p(\text{the}\|\text{maison})$ $= 0.05\times 0.1$ $= 0.005$	$= 0.005/0.68$ $= 0.007$

- with $\boldsymbol{p(a|e,f)} = \boldsymbol{p(e,f|a)}/\boldsymbol{p(e|f)}$ [next slide]
- and $\boldsymbol{p(e|f)} = \sum_a \boldsymbol{p(e,f|a)} = 0.56 + 0.035 + 0.08 + 0.005 = 0.68$

counts

- $c(\text{the}|\text{la}) = 0.824 + 0.052$ $\quad$ $c(\text{house}|\text{la}) = 0.052 + 0.007$
- $c(\text{the}|\text{maison}) = 0.118 + 0.007$ $\quad$ $c(\text{house}|\text{maison}) = 0.824 + 0.118$

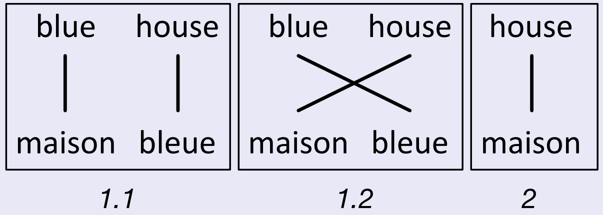
IBM Model 1 and EM – Expectation

- We need to compute $p(a|\boldsymbol{e}, \boldsymbol{f})$…
  - the probability of an alignment *a* given a pair of source (*e*) and foreign (*f*) sentences (a conditional probability)
- …Applying the chain rule:
  - $p(a|e,f) = \frac{p(e,f|a)}{p(e|f)}$
- …Given that we already have the formula for $p(e,f|a)$ (definition of Model 1)
- …We need to compute $p(e|f)$
  - [Koehn, 2010 – p. 89]

IBM Model 1 and EM – Example

corpus 2 sentences	1. blue house – maison bleue 2. house – maison

all possible alignments



Step 1. Set parameter values uniformly (2 words)

$t(\text{bleue}|\text{house}) = {}^1/_2$

$t(\text{maison}|\text{house}) = {}^1/_2$

$t(\text{bleue}|\text{blue}) = {}^1/_2$

$t(\text{maison}|\text{blue}) = {}^1/_2$

Step 2. Compute $p(a, f|e)$ for all alignments

1.1 $p(a, f|e) = {}^1/_2 \times {}^1/_2 = {}^1/_4$ (2 t=$^1/_2$ words)

1.2 $p(a, f|e) = {}^1/_2 \times {}^1/_2 = {}^1/_4$ (2 t=$^1/_2$ words)

2 $p(a, f|e) = {}^1/_2$ (1 t=$^1/_2$ word)

Step 3. Normalize $p(a, f|e)$ to yield $p(a|e, f)$ values

1.1 $p(a|e, f) = {}^1/_4 / {}^2/_4 = {}^1/_2$ (2 alignments p=$^1/_4$)

1.2 $p(a|e, f) = {}^1/_4 / {}^2/_4 = {}^1/_2$ (2 alignments p=$^1/_4$)

2 $p(a|e, f) = {}^1/_2 / {}^1/_2 = 1$ (1 alignments p=$^1/_2$)

Step 4. Collect fractional counts (**fc**)

$tc(\text{bleue}|\text{house}) = {}^1/_2$

$tc(\text{maison}|\text{house}) = {}^1/_2 + 1 = {}^3/_2$ (in 2 sentences)

$tc(\text{bleue}|\text{blue}) = {}^1/_2$

$tc(\text{maison}|\text{blue}) = {}^1/_2$

Step 1. Normalize **fc** to get revised parameter values

$t(\text{bleue}|\text{house}) = {}^1/_2 / {}^4/_2 = {}^1/_4 \; ({}^3/_2 + {}^1/_2)$

$t(\text{maison}|\text{house}) = {}^3/_2 / {}^4/_2 = {}^3/_4 \; ({}^3/_2 + {}^1/_2)$

$t(\text{bleue}|\text{blue}) = {}^1/_2 / 1 = {}^1/_2 \; ({}^1/_2 + {}^1/_2)$

$t(\text{maison}|\text{blue}) = {}^1/_2 / 1 = {}^1/_2 \; ({}^1/_2 + {}^1/_2)$

Repeat Step 2.

1.1 $p(a, f|e) = {}^1/_2 \times {}^1/_4 = {}^1/_8$

1.2 $p(a, f|e) = {}^1/_2 \times {}^3/_4 = {}^3/_8$

2 $p(a, f|e) = {}^3/_4$

Repeat Step 3.

1.1 $p(a|e, f) = {}^1/_8 / {}^2/_4 = {}^1/_4$

1.2 $p(a|e, f) = {}^3/_8 / {}^2/_4 = {}^3/_4$

2 $p(a|e, f) = {}^1/_2 / {}^1/_2 = 1$

Repeat Step 4.

$tc(\text{bleue}|\text{house}) = {}^1/_4$

$tc(\text{maison}|\text{house}) = {}^3/_4 + 1 = {}^7/_4$

$tc(\text{bleue}|\text{blue}) = {}^3/_4$

$tc(\text{maison}|\text{blue}) = {}^1/_4$

Repeat Step 4.

$t(\text{bleue}|\text{house}) = {}^1/_4 / {}^4/_2 = {}^1/_8$

$t(\text{maison}|\text{house}) = {}^7/_4 / {}^4/_2 = {}^7/_8$

$t(\text{bleue}|\text{blue}) = {}^3/_4 / 1 = {}^3/_4$

$t(\text{maison}|\text{blue}) = {}^1/_4 / 1 = {}^1/_4$

IBM Model 1 and EM – Example (cont.)

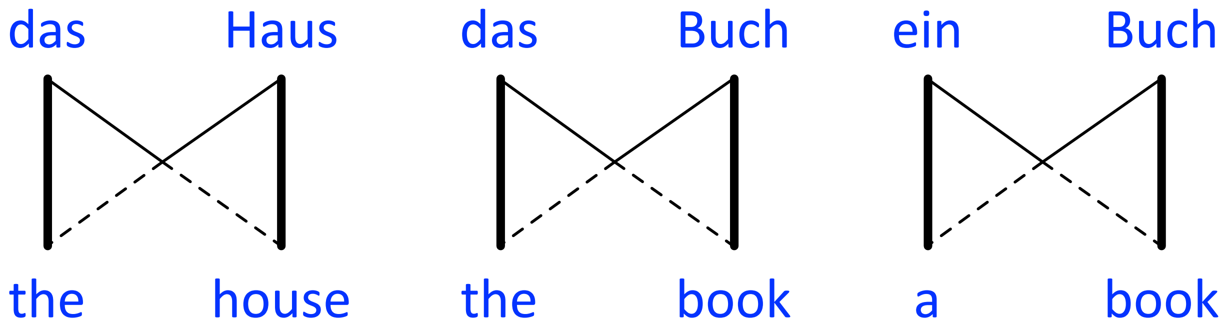
- Repeating step 2-5 many times yields:
  - $t(\text{bleue}|\text{house}) = 0.0001$
  - $t(\text{maison}|\text{house}) = 0.9999$
  - $t(\text{bleue}|\text{blue}) = 0.9999$
  - $t(\text{maison}|\text{blue}) = 0.0001$

IBM Model 1 and EM – Pseudocode

```
Input:  set of sentence pairs (e,f)
Output: translation prob. t(e|f)
initialize t(e|f) uniformly
while not converged do
  // initialize
  count(e|f) = 0 for all e,f
  total(f) = 0 for all f
  for all sentence pairs (e,f) do
    // compute normalization
    for all words e in e do
      s-total(e) = 0
      for all words f in f do
        s-total(e) += t(e|f)
      end for
    end for
    // collect counts
    for all words e in e do
      for all words f in f do
        count(e|f)+= t(e|f) /s-total(e)
        total(f)+= t(e|f) /s-total(e)
      end for
    end for
  end for
  // estimate probabilities
  for all foreign words f do
    for all English words e do
      t(e|f) = count(e|f)/total(f)
    end for
  end for
end while
```

EM – Convergence

- The data



- The results

e	f	initial	1st it.	2nd it.	3rd it.	...	final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	0.3466	...	0
a	ein	0.25	0.5	0.5714	0.6534	...	1
the	haus	0.25	0.5	0.4286	0.3466	...	0
house	haus	0.25	0.5	0.5714	0.6534	...	1

Perplexity

- How well does the model fit the data?
- Perplexity: derived from probability of the training data according to the model
  - $\log_2 PP = -\sum_s \log_2 p(e_s|f_s)$
    - here e is the **translation** and f the **source** (classical notation)
- Example ($\epsilon = 1$)

	initial	1st it.	2nd it.	3rd it.	...	final
p(thehaus|dashaus)	0.0625	0.1875	0.1905	0.1913	...	0.1875
p(thebook|dasbuch)	0.0625	0.1406	0.1790	0.2075	...	0.25
p(abook|einbuch)	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8

- Δ perplexity is the convergence
  - because of the convergence behavior of EM

Ensuring fluent output

- The translation model cannot decide between small and little
  - sometime one is preferred over the other:
    - small step: 2,070,000 occurrences in the Google index
    - little step: 257,000 occurrences in the Google index
- who is here to help? The **language model**
  - estimate how likely a string is English based on n-gram statistics
    - $p(e) \quad = p(e_1, e_2, \ldots, e_n)$
    - $\qquad\quad = p(e_1)p(e_2|e_1) \ldots p(e_n|e_1, e_2, \ldots, e_{n-1})$
    - $\qquad\quad \approx p(e_1)p(e_2|e_1) \ldots p(e_n|e_{n-2}, e_{n-1})$

Noisy Channel Model

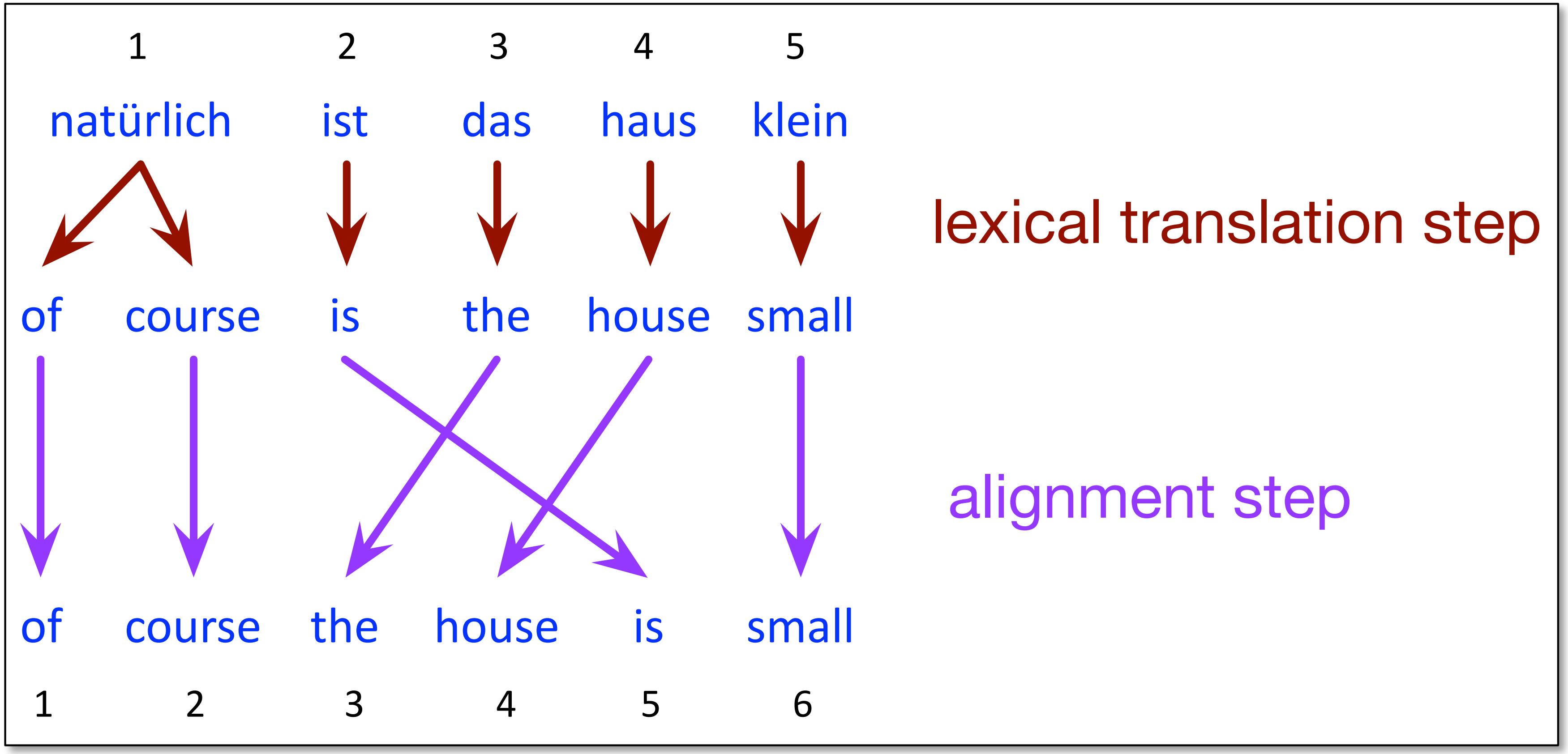
- In order to integrate a language model
  - Bayes Rule
  - $\mathrm{argmax}_{\boldsymbol{e}} p(\boldsymbol{e}|\boldsymbol{f}) = \mathrm{argmax}_{\boldsymbol{e}} \frac{p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e})}{p(\boldsymbol{f})}$
  - $\mathrm{argmax}_{\boldsymbol{e}} p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e})$
    - here $\boldsymbol{e}$ is the **translation** and $\boldsymbol{f}$ the **source** (classical notation)

Higher IBM Models

IBM Model 1	**lexical translation**
	has a global maximum
IBM Model 2	**adds absolute alignment (reordering) model**
	modeling alignments with probability distribution translating foreign word at position i to English word at position j: $a(i\|j, l_e, l_f)$
IBM Model 3	**adds fertility model**
	number of English words generated by a foreign word f: $n(\phi\|f)$ where ϕ is the number of words f translates into
IBM Model 4	**adds relative alignment (reordering) model**
	relative to previously translated words
IBM Model 5	**fixes deficiency**
	Models 1-4 are deficient i.e. – some impossible translations have positive probability – multiple output words may be placed in the same position

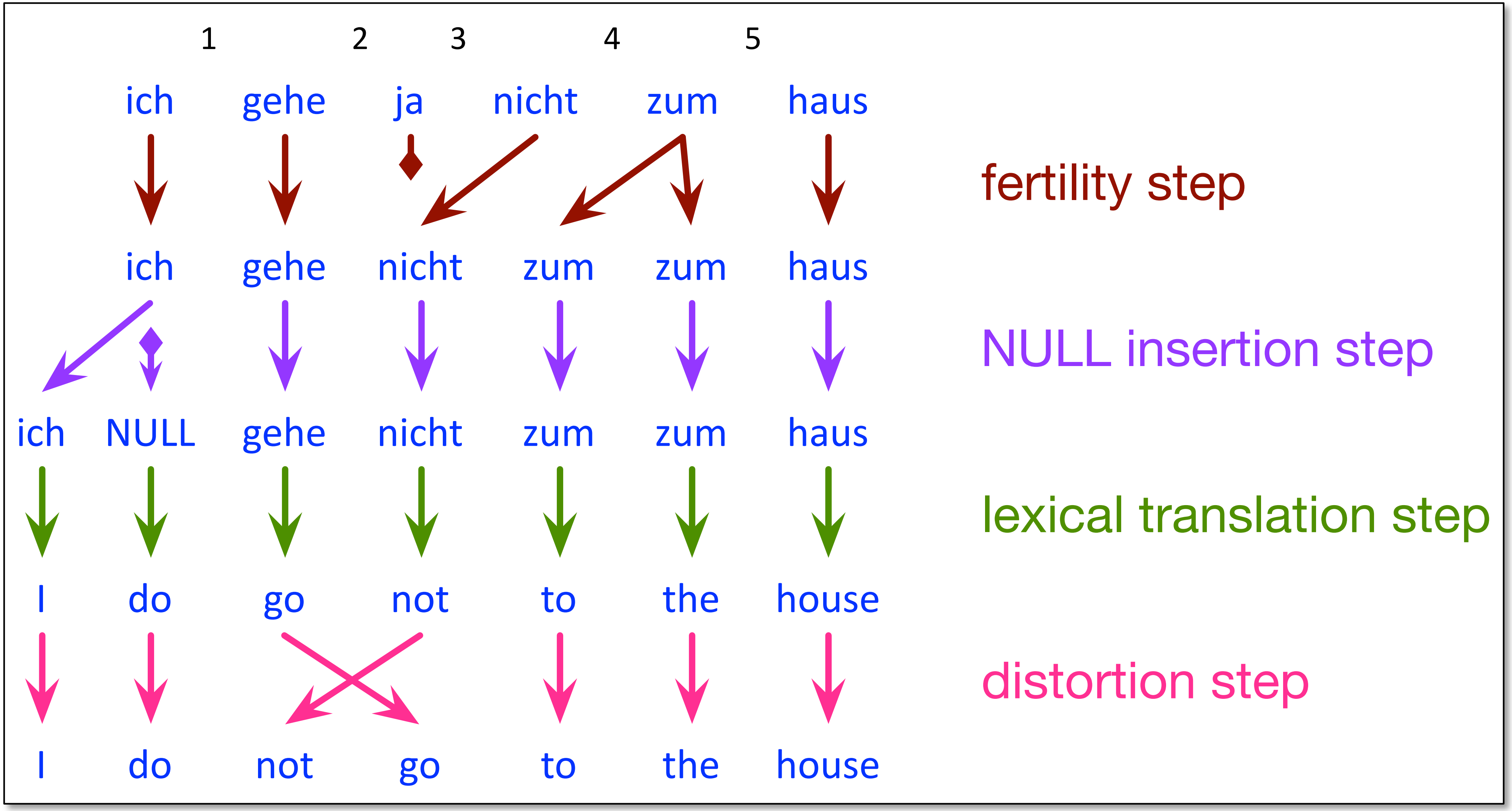
IBM Model 2

- Adding a model of alignment



IBM Model 3

- Adding a model of fertility



Summary

- Lexical translation
- Alignment
- Expectation Maximization (EM) Algorithm
- Noisy Channel Model
- IBM Models 1–5

Summary

- IBM Models were the pioneering models in statistical machine translation
- Introduced important concepts
  - generative model
  - EM training
  - reordering models
- Only used for niche applications as translation model
- ... but still in common use for word alignment
  - e.g., GIZA++ toolkit

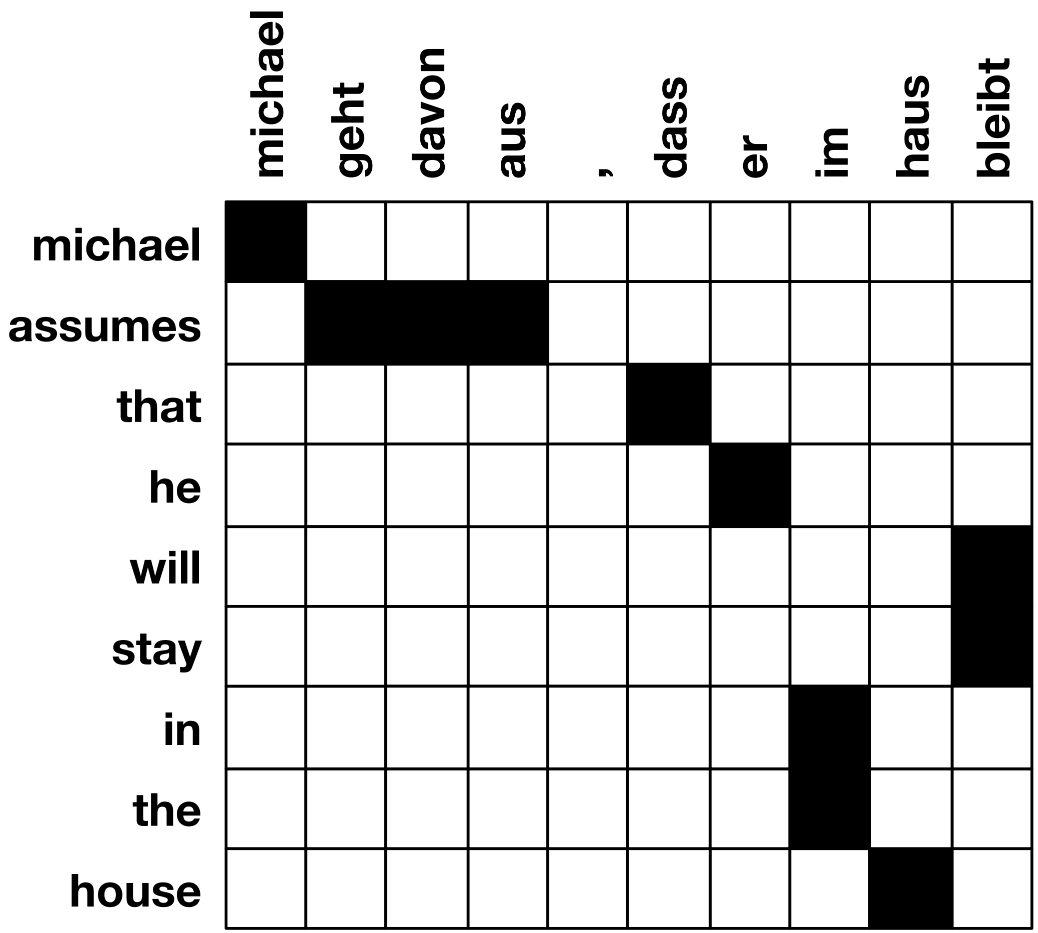
WORD ALIGNMENT

Foreword

- Important notion introduced by IBM models
- We will
  - develop this concept further
  - point out problems
  - discuss how word alignment quality is measured
  - present a method based on IBM models but fixes their most glaring problem: limitation to one-to-many alignments

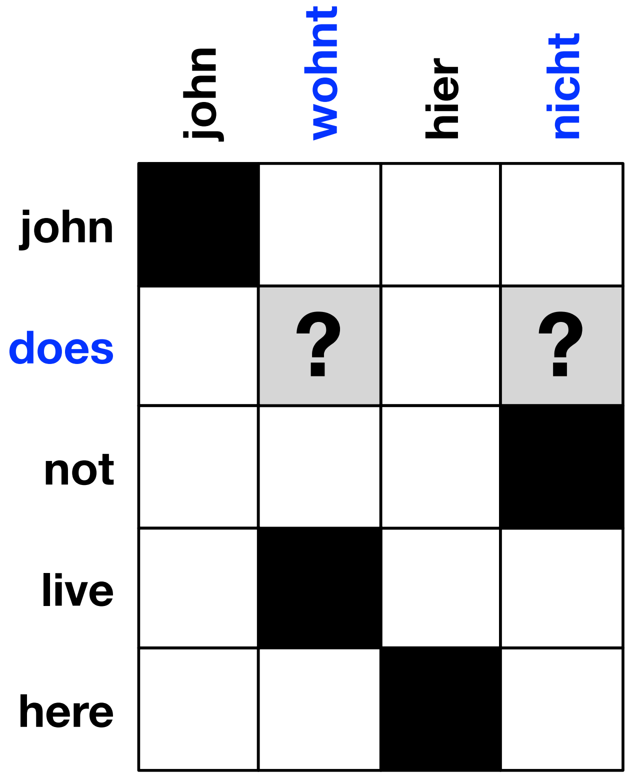
The Task

- Given a sentence pair, which words correspond to each other?



Word Alignment?

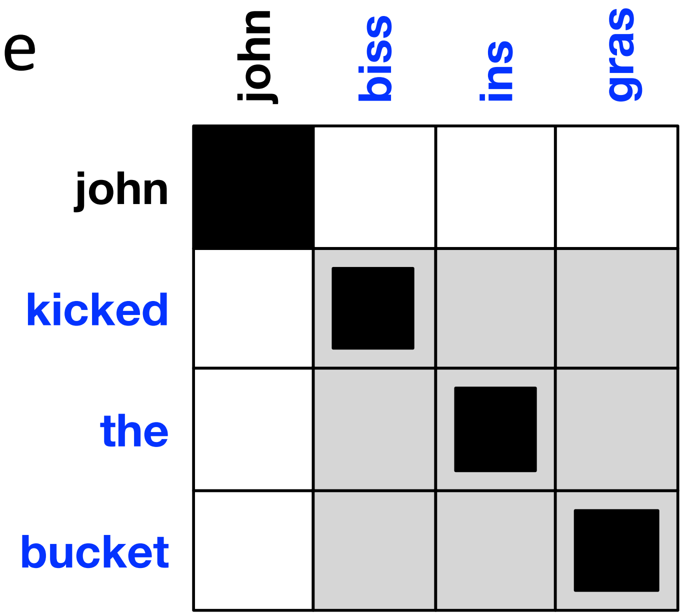
- Example



- Is the English word does aligned to the German wohnt (verb) or nicht (negation) or neither?

Word Alignment?

- How do the idioms kicked the bucket and biss ins grass match up?
  - to kick the bucket => to die
    - biss *(ge)* ⟶ bite *(en)*
    - ins *(ge)* ⟶ in the *(en)*
    - gras *(ge)* ⟶ grass *(en)*



- Outside this exceptional context, bucket is never a good translation for grass

Measuring Word Aligment Quality

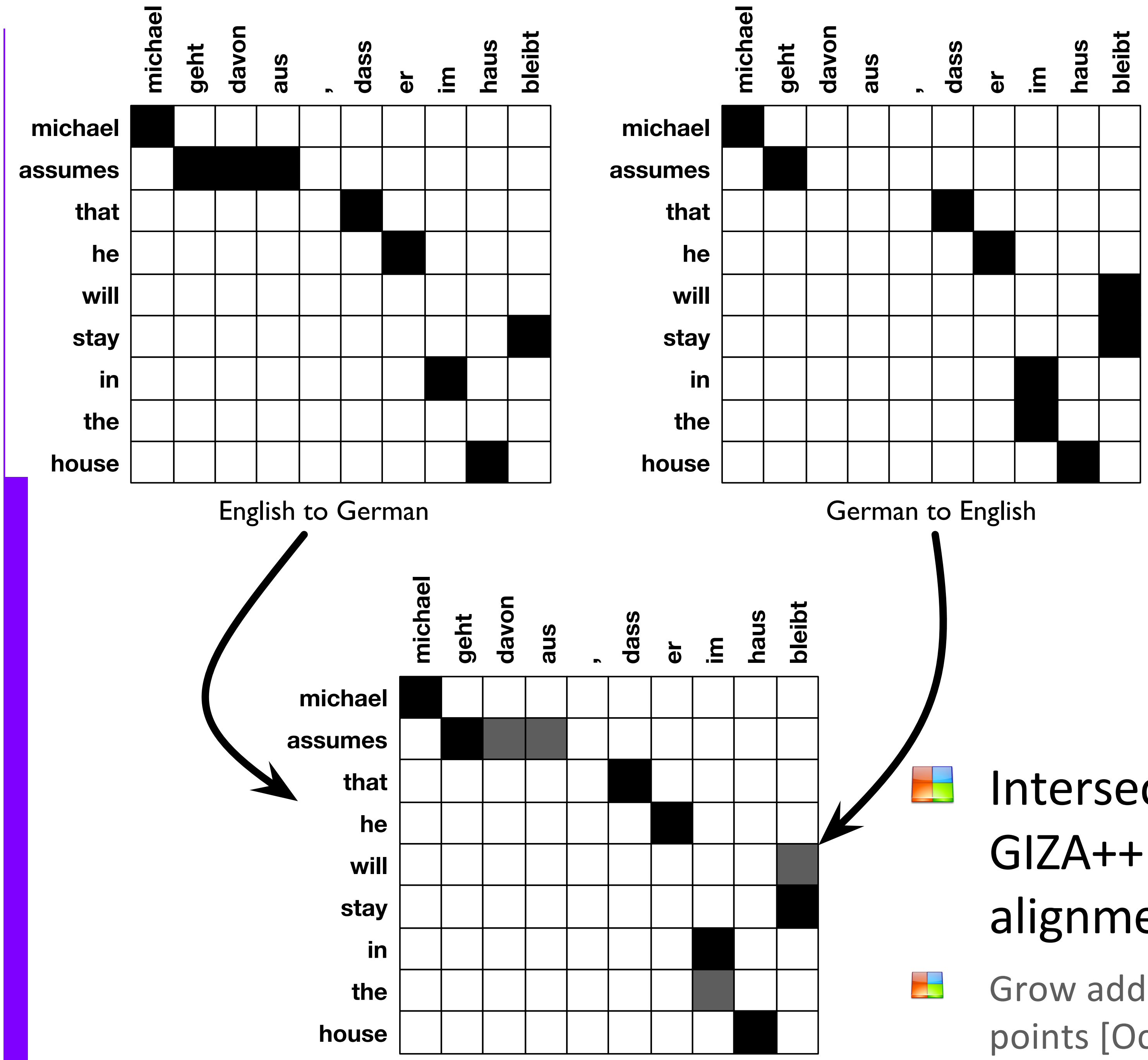
- Manually align corpus with sure (S) and possible (P) alignment points ($S \subseteq P$). **[reference]**
- Common metric for evaluating computed word alignment A: Alignment Error Rate (AER)
  - $AER(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$
- AER = 0: alignment A matches all sure, any possible alignment points
- However: different applications require different precision/recall trade-offs

Word Alignment with IBM Models

- IBM Models create a **many-to-one** mapping
  - words are aligned using an alignment function
  - a function may return the same value for different input (one-to-many mapping)
  - a function can not return multiple values for one input (no many-to-one mapping)

- But!!
  - Real word alignments have **many-to-many** mappings

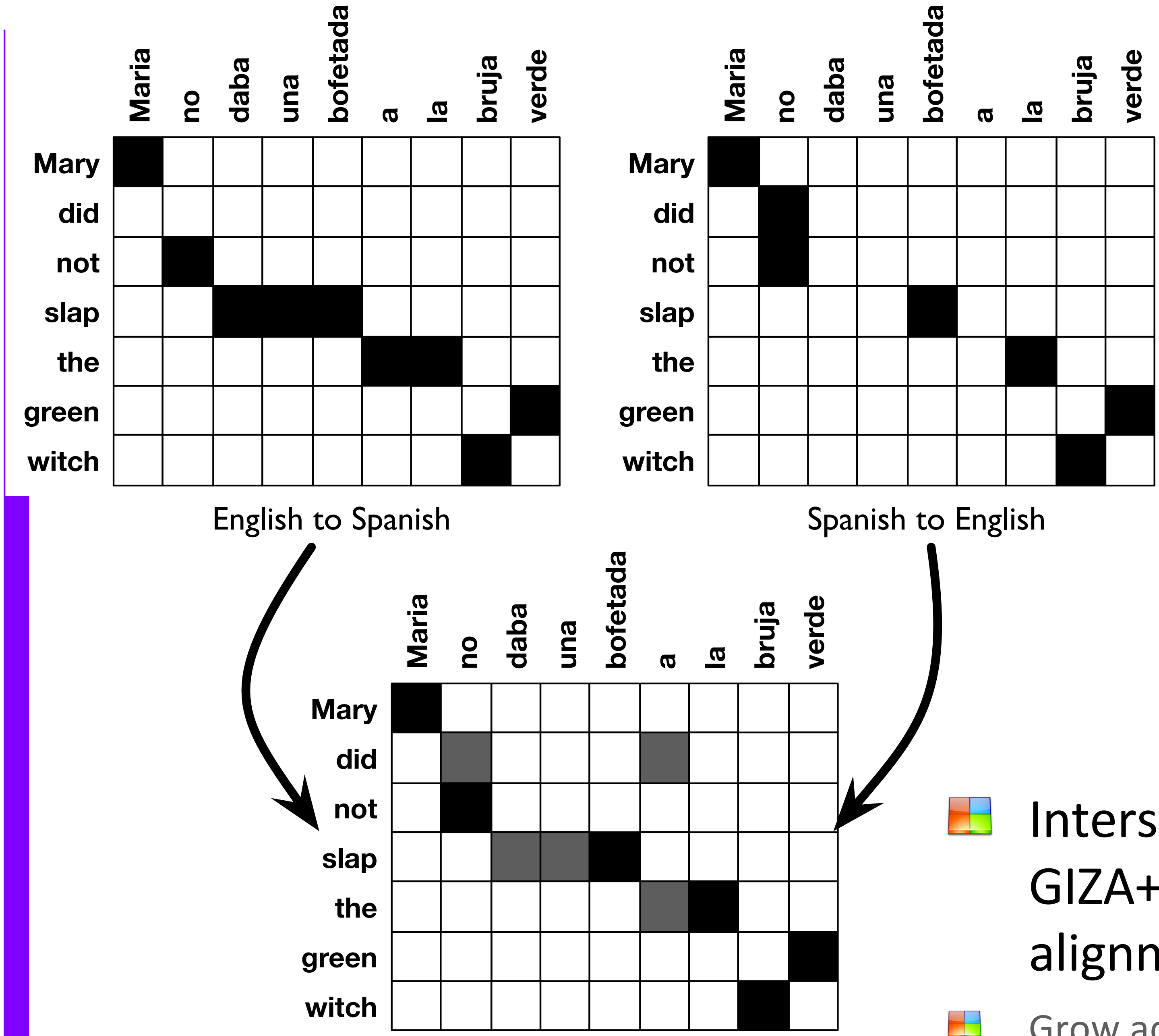
Symmetrizing Word Alignments



Intersection + grow additional alignment point

- Intersection of GIZA++ bidirectional alignments
- Grow additional alignment points [Och and Ney, CompLing2003]

Symmetrizing Word Alignments



Intersection + grow additional alignment points

- Intersection of GIZA++ bidirectional alignments
  - Grow additional alignment points [Och and Ney, CompLing2003]

Growing Heuristic

grow-diag-final(e2f,f2e)

neighboring = {(-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1)}
alignment A = intersect(e2f,f2e); grow-diag(); final(e2f); final(f2e);

grow-diag()

while new points added **do**
  for all English word $\boldsymbol{e} \in [\mathbf{1} \dots \boldsymbol{e_n}]$, foreign word $\boldsymbol{f} \in [\mathbf{1} \dots \boldsymbol{f_n}]$, $(\boldsymbol{e}, \boldsymbol{f}) \in \boldsymbol{A}$ **do**
    for all neighboring alignment points $(\boldsymbol{e_{new}}, \boldsymbol{f_{new}})$ **do**
      if ($\boldsymbol{e_{new}}$ unaligned **or** $\boldsymbol{f_{new}}$ unaligned) **and** $(\boldsymbol{e_{new}}, \boldsymbol{f_{new}}) \in$ union(e2f,f2e) **then**
        add $(\boldsymbol{e_{new}}, \boldsymbol{f_{new}})$ to A
      end if
    end for
  end for
end while

final()

for all English word $\boldsymbol{e_{new}} \in [\mathbf{1} \dots \boldsymbol{e_n}]$, foreign word $\boldsymbol{f_{new}} \in [\mathbf{1} \dots \boldsymbol{f_n}]$ **do**
  if ($\boldsymbol{e_{new}}$ unaligned **or** $\boldsymbol{f_{new}}$ unaligned) **and** $(\boldsymbol{e_{new}}, \boldsymbol{f_{new}}) \in$ union(e2f,f2e) **then**
    add $(\boldsymbol{e_{new}}, \boldsymbol{f_{new}})$ to A
  end if
end for

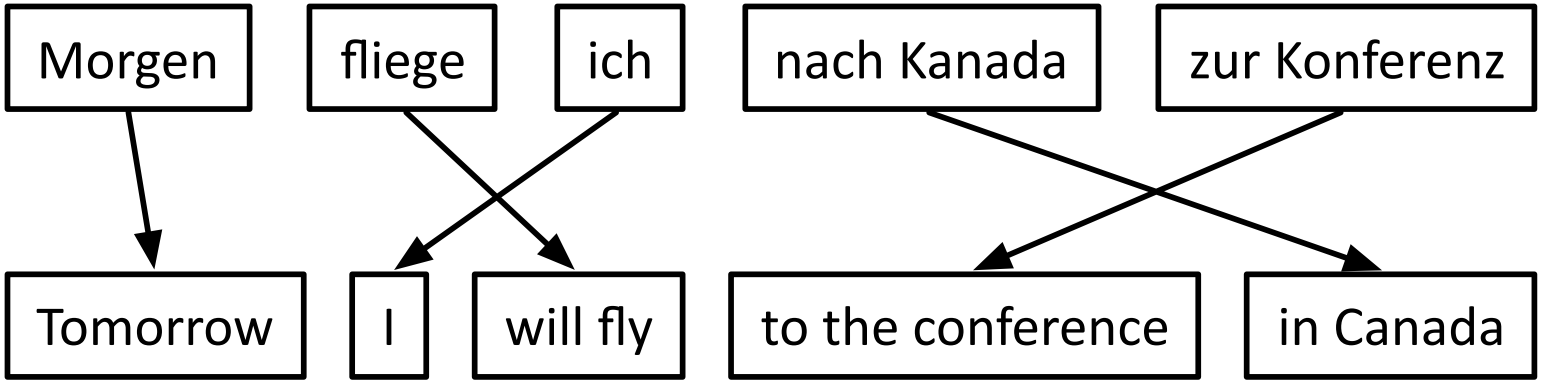
PHRASE-BASED MODELS

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

Case study

- Example



- Comments
  - Foreign input is segmented in phrases
    - any sequence of words, not necessarily linguistically motivated
  - Each phrase is translated into English
  - Phrases are reordered

Phrase-Based Translation Model

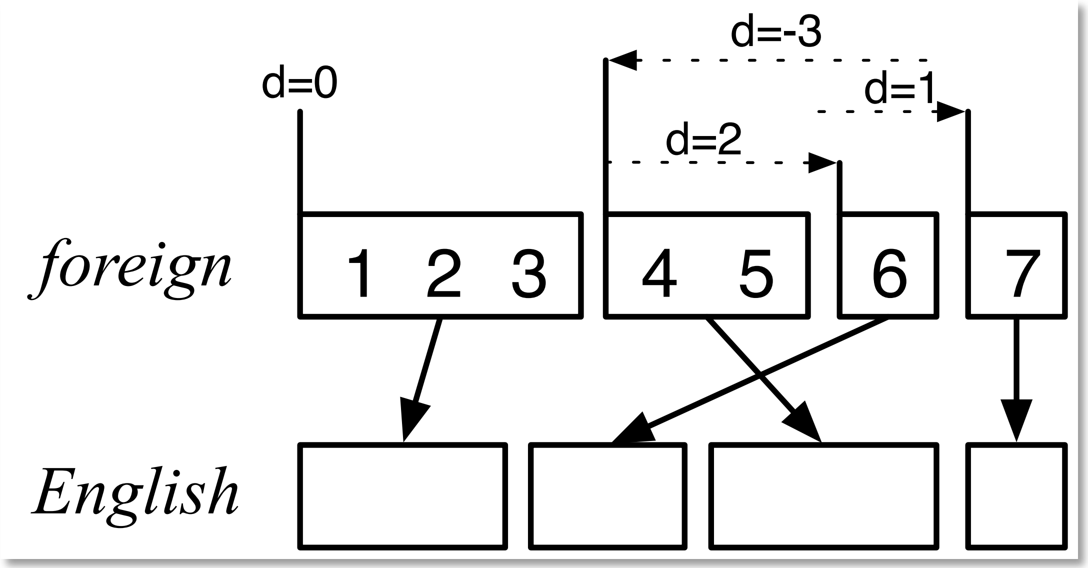
- Major components of phrase-based model
  - phrase translation model $\phi(f|e)$
  - distance-based reordering model d
  - language model $p_{\mathrm{LM}}(e)$
- Bayes rule
  - $\mathrm{e}_{\mathrm{best}} = \mathrm{argmax}_e p(e|f) = \mathrm{argmax}_e p(f|e) p_{\mathrm{LM}}(e)$
- Sentence f is decomposed into I phrases ($\bar{f}_1^I = \bar{f}_1, \ldots, \bar{f}_I$)
- For the model, $p(f|e)$ is further decomposed into
  - $p\left(\bar{f}_1^I | \bar{e}_1^I\right) = \prod_{i=1}^{I} \phi\left(\bar{f}_i | \bar{e}_i\right) d(start_i - end_{i-1} - 1)$

Breakdown of the formula

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

- each foreign phrase $\bar{f}_i$ is translated into an English phrase $\bar{e}_i$ all segmentation are equally likely
- due to noisy channel, translation is inverted thus translation probability $\phi(\bar{f}_i|\bar{e}_i)$ is modelled as translation from English to foreign
- reordering is handled by a distance-based reordering model (reordering relative to the previous point) [next slide]
  - $start_i$ position of the first word of the foreign input phrase that translates to the English phrase
  - end_i position of the last word of that foreign phrase
  - reordering computed as $start_i - end_{i-1} - 1$

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*

Phrase Translation Table: Example

- PTT for den Vorschlag learned from the EuroParl corpus

English	$\phi(e\|f)$	English	$\phi(e\|f)$
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 Phrase?

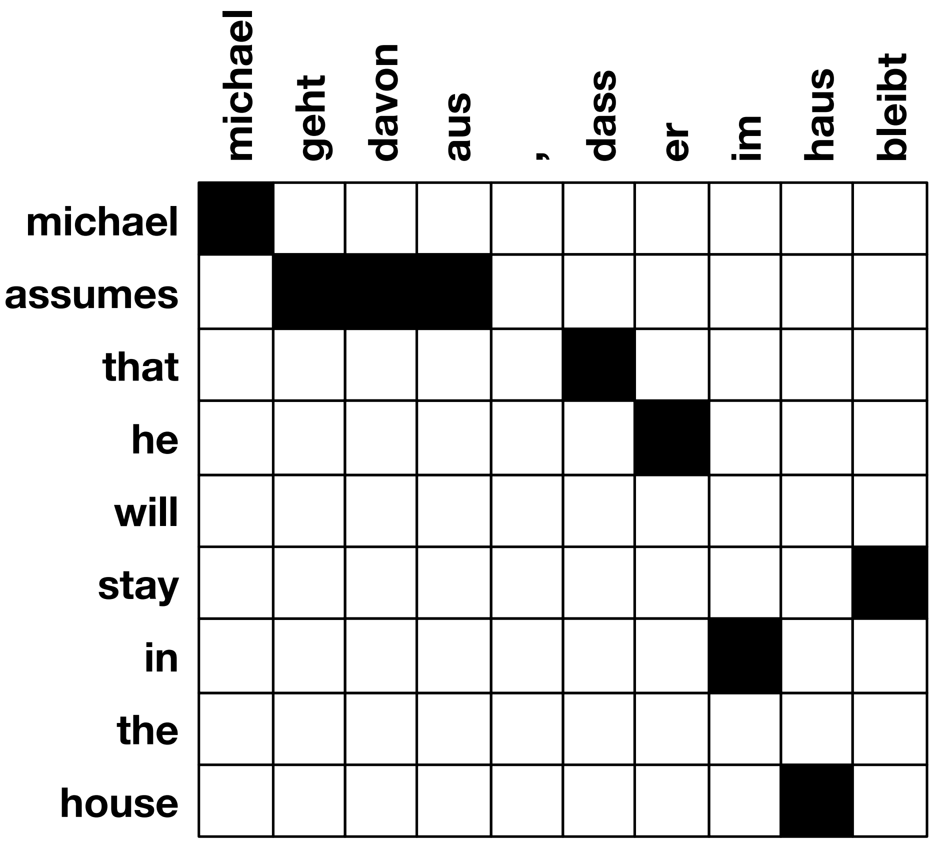
- Model is not limited to linguistic phrases
  - noun phrases, verb phrases, prepositional phrases, ...
- Example non-linguistic phrase pair
  - spass am → fun with the
- Prior noun often helps with translation of preposition
- Experiments show that limitation to linguistic phrases hurts quality

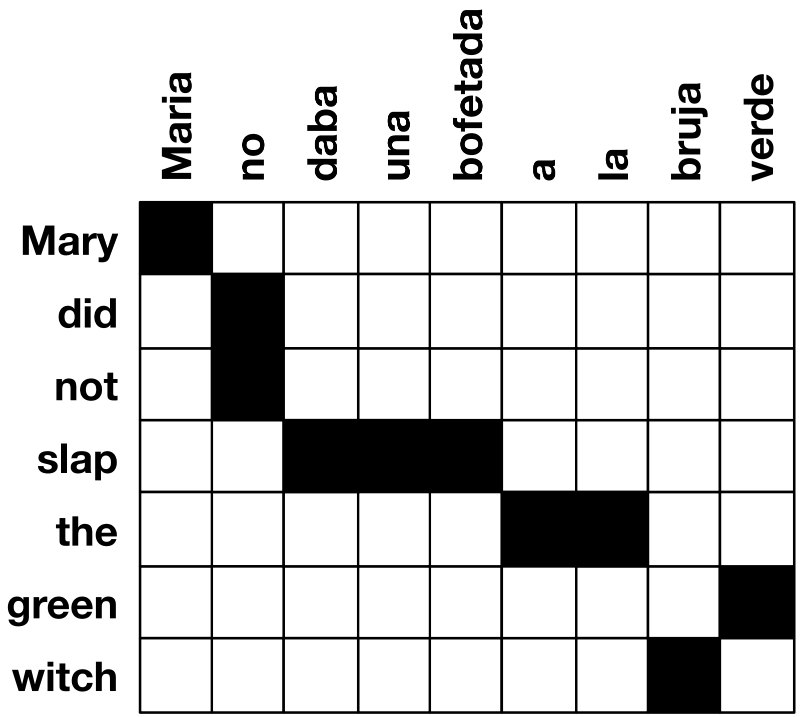
How to Learn the Translation Table?

- Three stages
  1. collect word alignments: using IBM or other
  2. extract phrase pairs
  3. score phrase pairs

Word alignments

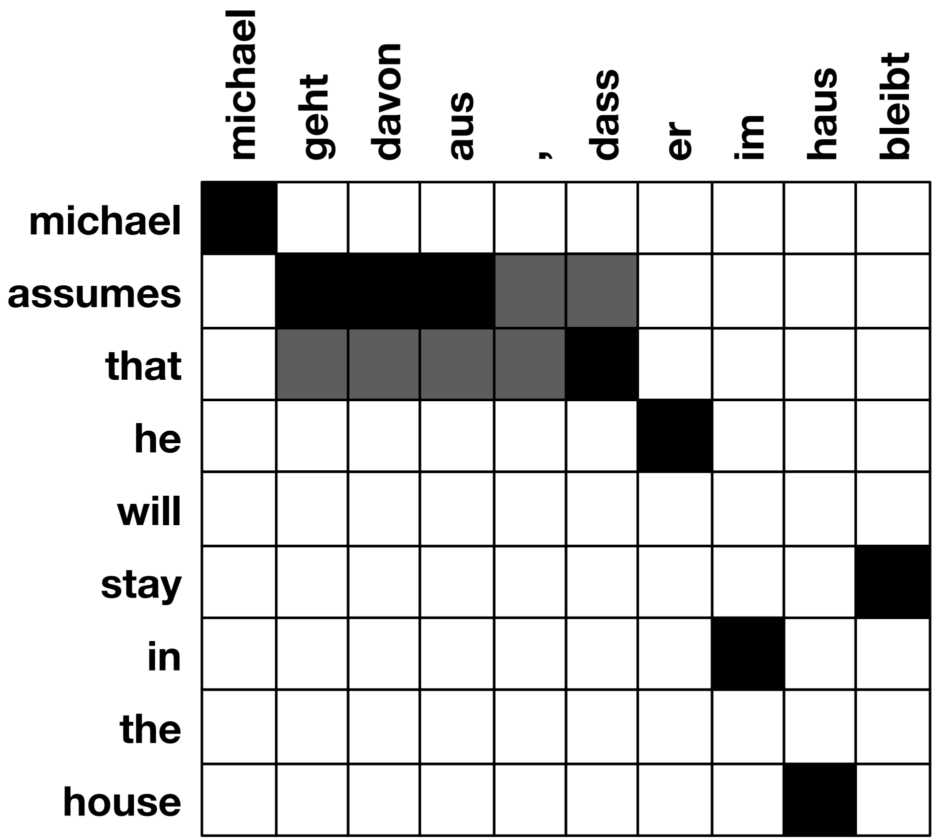
- Examples





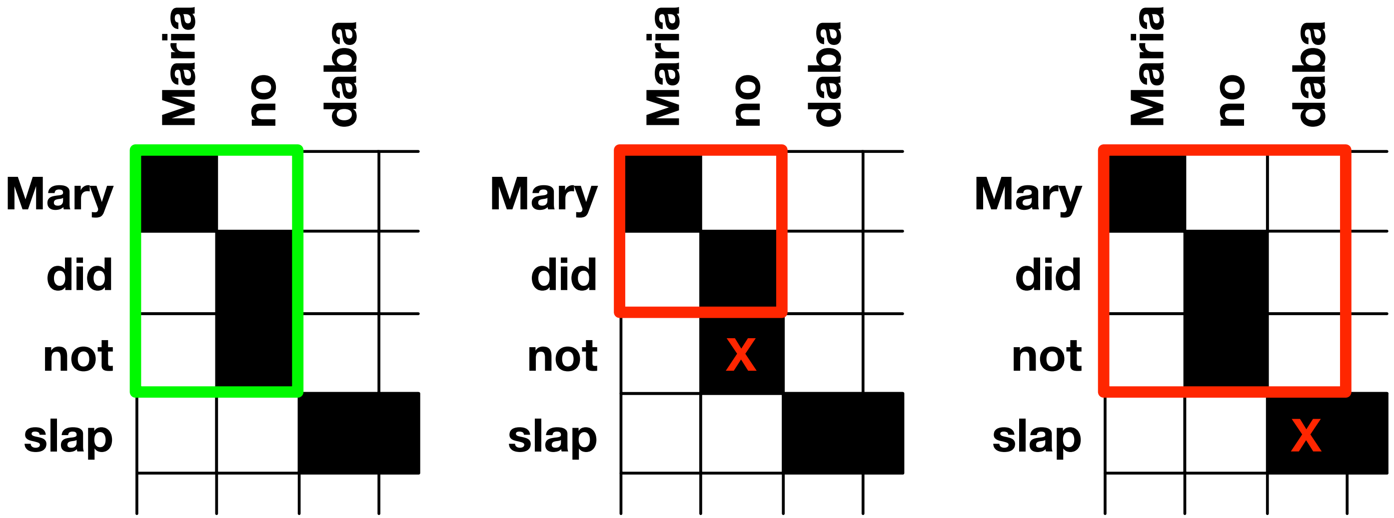
Extracting Phrase Pairs

- …consistent with word alignment
- Example
  - assumes that / geht davon aus , dass



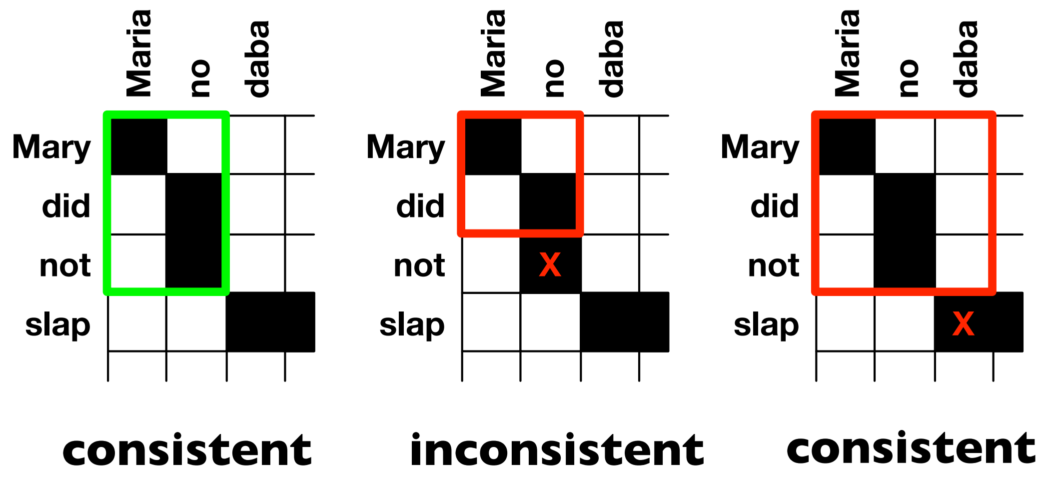
Consistency?

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



consistent	**inconsistent**	**consistent**
OK	one alignment point outside	OK unaligned word is fine

Consistency?



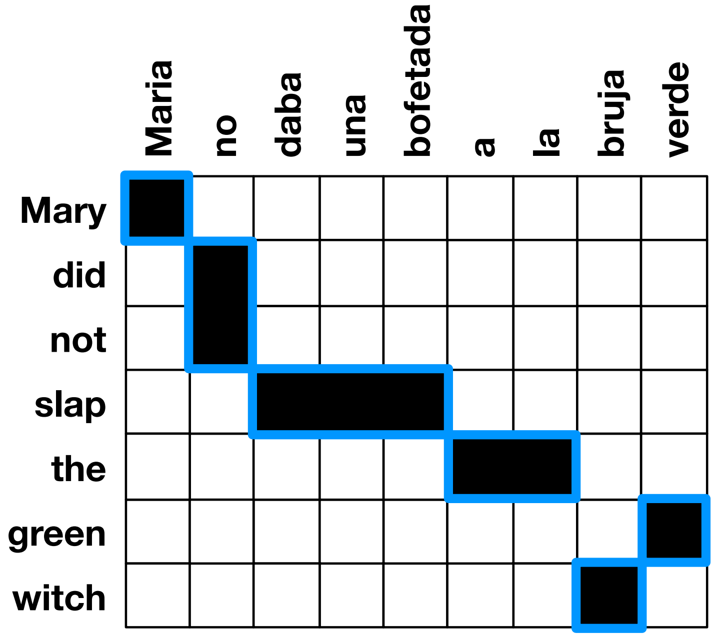
- A phrase pair $(\bar{e}, \bar{f})$ is consistent with an alignment A, if all words $f_1, \ldots, f_n$ in $\bar{f}$ that have alignment points in A have these with words $e_1, \ldots, e_m$ in $\bar{e}$ and vice versa:
  - $(\bar{e}, \bar{f})$ consistent with $A \Leftrightarrow$

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

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

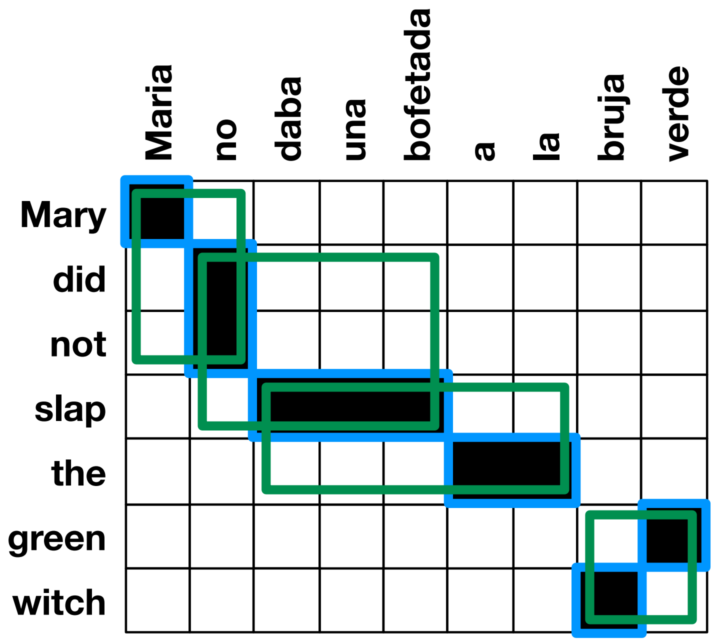
$$\text{AND} \quad \exists e_i \in \bar{e}, f_j \in \bar{f} : (e_i, f_j) \in A$$

Word Alignment Induced Phrases



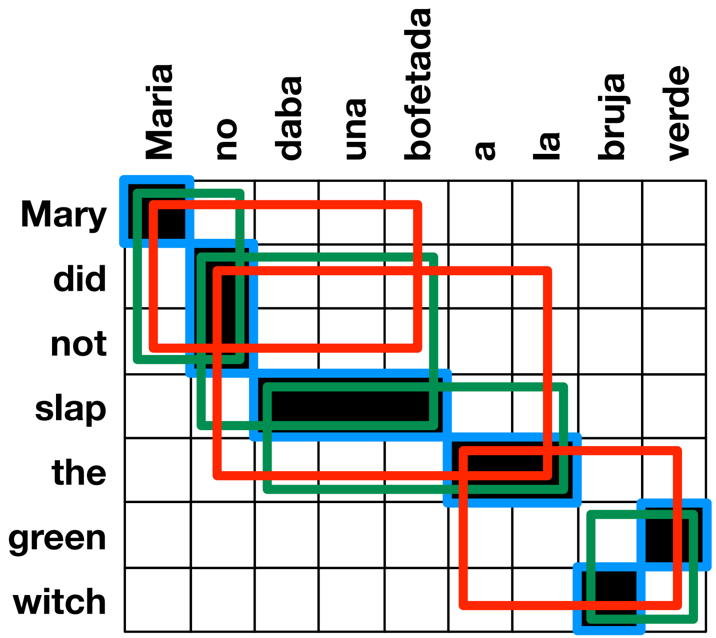
- (Maria, Mary), (no, did not), (daba una bofetada, slap), (a la, the), (bruja, witch), (verde, green)

Word Alignment Induced Phrases



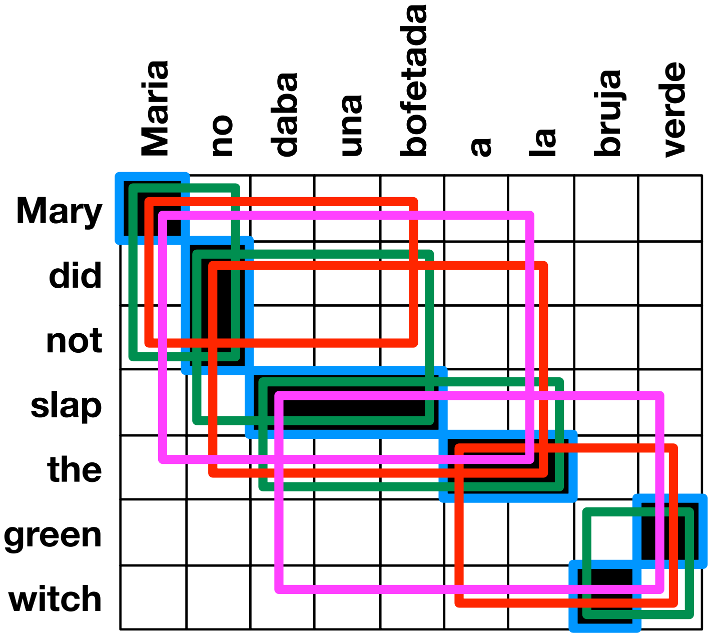
- (Maria, Mary), (no, did not), (daba una bofetada, slap), (a la, the), (bruja, witch), (verde, green)
- (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)

Word Alignment Induced Phrases



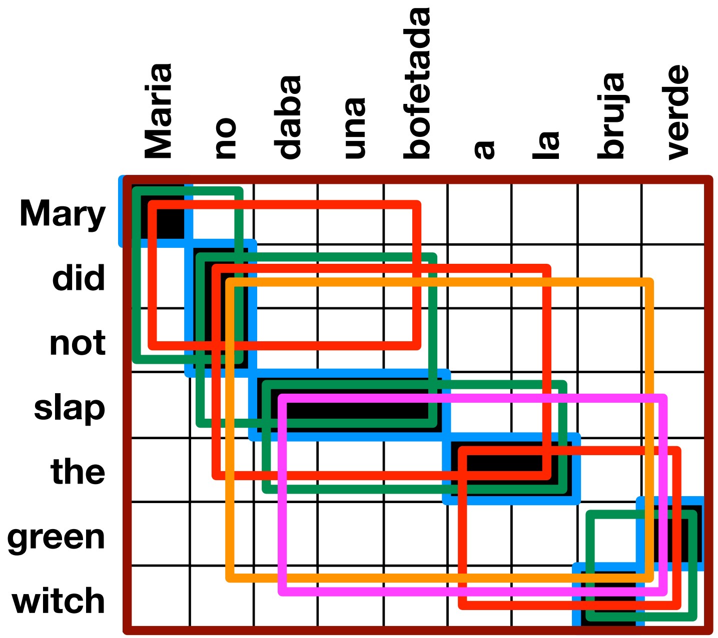
- (Maria, Mary), (no, did not), (daba una bofetada, slap), (a la, the), (bruja, witch), (verde, green)
- (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)
- (Maria no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)

Word Alignment Induced Phrases



- (Maria, Mary), (no, did not), (daba una bofetada, slap), (a la, the), (bruja, witch), (verde, green)
- (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)
- (Maria no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)
- (Maria no daba una bofetada a la, Mary did not slap the), (daba una bofetada a la bruja verde, slap the green witch)

Word Alignment Induced Phrases



- (Maria, Mary), (no, did not), (daba una bofetada, slap), (a la, the), (bruja, witch), (verde, green)
- (Maria no, Mary did not), (no daba una bofetada, did not slap), (daba una bofetada a la, slap the), (bruja verde, green witch)
- (Maria no daba una bofetada, Mary did not slap), (no daba una bofetada a la, did not slap the), (a la bruja verde, the green witch)
- (Maria no daba una bofetad a la, Mary did not slap the), (daba una bofetada a la bruja verde, slap the green witch)
- (no daba una bofetada a la bruja verde, did not slap the green witch)
- (Maria no daba una bofetada a la bruja verde, Mary did not slap the green witch)

Scoring Phrase Translations

- Phrase pair extraction:
  - collect all phrase pairs from the data
- Phrase pair scoring:
  - assign probabilities to phrase translations
    - probability distribution of phrase pairs: $\phi(\bar{f}, \bar{e})$
  - Score by relative frequency:

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

Size of the Phrase Table

- Comment
  - 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

Reordering

- Several options
  - Monotone translation, i.e. do not allow any reordering
    - worse translations
  - Limiting reordering (to movement over max. number of words) helps
  - Distance-based reordering cost
    - moving a foreign phrase over n words: cost ω^n
  - Lexicalized reordering model

Log-linear Models

- IBM Models provided mathematical justification for factoring components *(features)* together
  - $p_{\mathrm{LM}} \times p_{\mathrm{D}} \times p_{\mathrm{TM}}$
  - (Language Model, Translation Model, Distortion)
- The models *(features)* may be weighted
  - $p_{\mathrm{LM}}^{\lambda_{\mathrm{LM}}} \times p_{\mathrm{D}}^{\lambda_{\mathrm{D}}} \times p_{\mathrm{TM}}^{\lambda_{\mathrm{TM}}}$
- Many components *(features)* p_i with weights λ_i
  - $\prod_i p_i^{\lambda_i} = exp(\sum_i \lambda_i log(p_i))$
  - $log \prod_i p_i^{\lambda_i} = \sum_i \lambda_i log(p_i)$

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 $h1 = \log \phi$
  - feature function $h2 = \log d$
  - feature function $h3 = \log p_{\mathrm{LM}}$

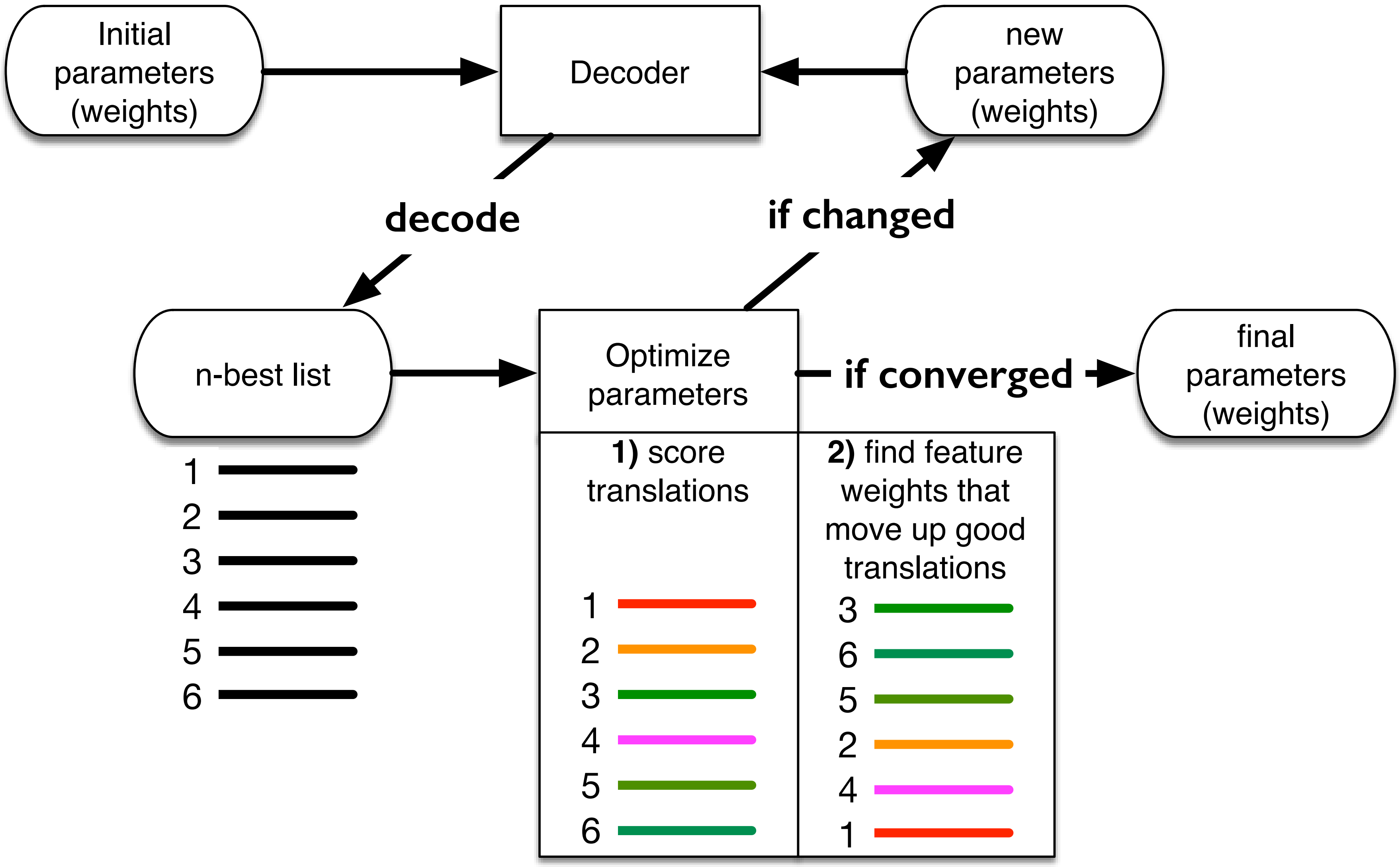
Knowledge Sources (features)

- Quite a lot:
  - language model
  - reordering (distortion) model
  - phrase translation model
  - word translation model
  - word count
  - phrase count
  - drop word feature
  - phrase pair frequency
  - additional language models
  - additional features

Tuning Feature Weights

- Goal
  - for each component *(feature)* p_i; determine its weight λ_i, i.e. the contribution of p_i
- Methods
  - manual setting of weights: try a few, take best
  - automate this process: learn weights
- Learn weights
  - set aside a **development corpus**
  - set the weights, so that **optimal translation performance** on this development corpus is achieved
  - requires *automatic scoring* method (e.g., BLEU)

Feature Weights Learning



Discriminative vs. Generative Models

Generative models

- translation process is broken down to *steps*
- each step is modeled by a *probability distribution*
- each probability distribution is estimated from the data by *maximum likelihood*

Discriminative models

- model consist of a number of *features* (e.g. the language model score)
- each feature has a *weight*, measuring its value for judging a translation as correct
- feature weights are *optimized on development* data, so that the system output matches correct translations as close as possible

Discriminative Training

- Training set (*development set*)
  - different from original training set
  - small (maybe 1000 sentences)
  - must be different from test set
- Current model *translates* this development set
  - *n-best list* of translations (n=100, 10000)
  - translations in n-best list can be *scored*
- Feature weights are *adjusted*
- n-best list generation and feature weight adjustment repeated for a number of iterations

Methods to Adjust Feature Weights

- **Maximum entropy** [Och and Ney, ACL2002]
  - match *expectation* of feature values of model and data
- **Minimum error rate training** [Och, ACL2003]
  - try to *rank best translations first* in n-best list
  - can be adapted for various error metrics, even BLEU
- **Ordinal regression** [Shen et al., NAACL2004]
  - *separate* k worst from the k best translations

Summary

- Phrase Model
- Training the model
  - word alignment
  - phrase pair extraction
  - phrase pair scoring
- Log linear model
  - sub-models as feature functions
  - lexical weighting
  - word and phrase count features
- EM training of the phrase model

DECODING

The Task

- A mathematical model for translation
  - $p(\mathrm{e}|\mathrm{f})$
- Find the best scoring translation $\mathrm{e}_{\mathrm{best}}$ according to the features and their respective weights
  - $\mathrm{e}_{\mathrm{best}} = \mathrm{argmax}_{\mathrm{e}} p(\mathrm{e}|\mathrm{f})$
- A very hard problem
  - NP-complet [Knight 1999]
  - i.e. examining all possible translations, scoring them, and picking the best is computationally too expensive even for a sentence of modest length
- In practice
  - heuristic search methods
- Two types of error
  - the most probable translation is bad: fix de model
  - search does not find the most probable translation: fix the search
- Decoding is evaluated by search errors, not quality of translation
  - although these are often correlated

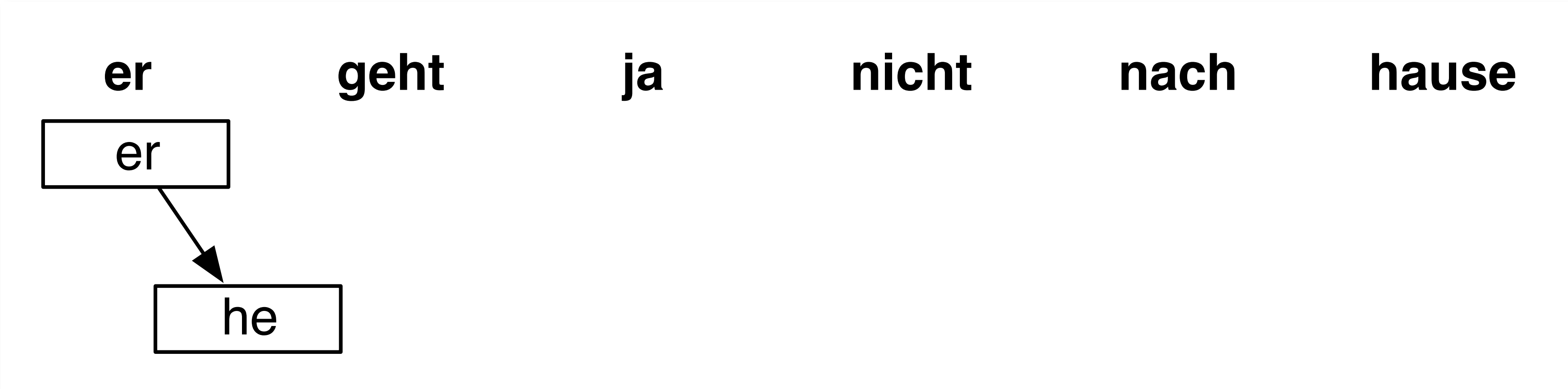
Translating a Sentence

- Input sentence to be translated in English

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

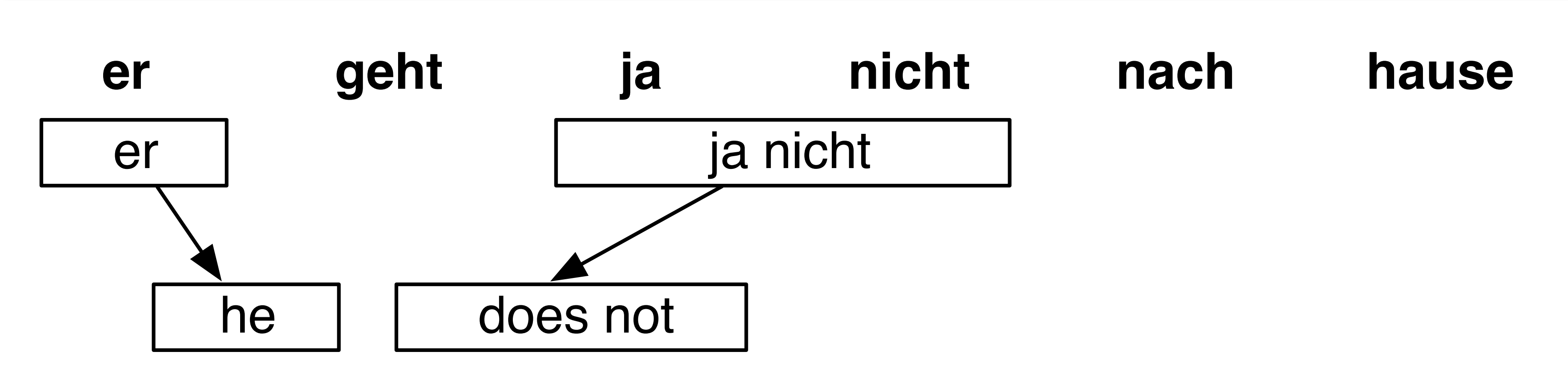
Translating a Sentence

- Pick a phrase in the input, translate it



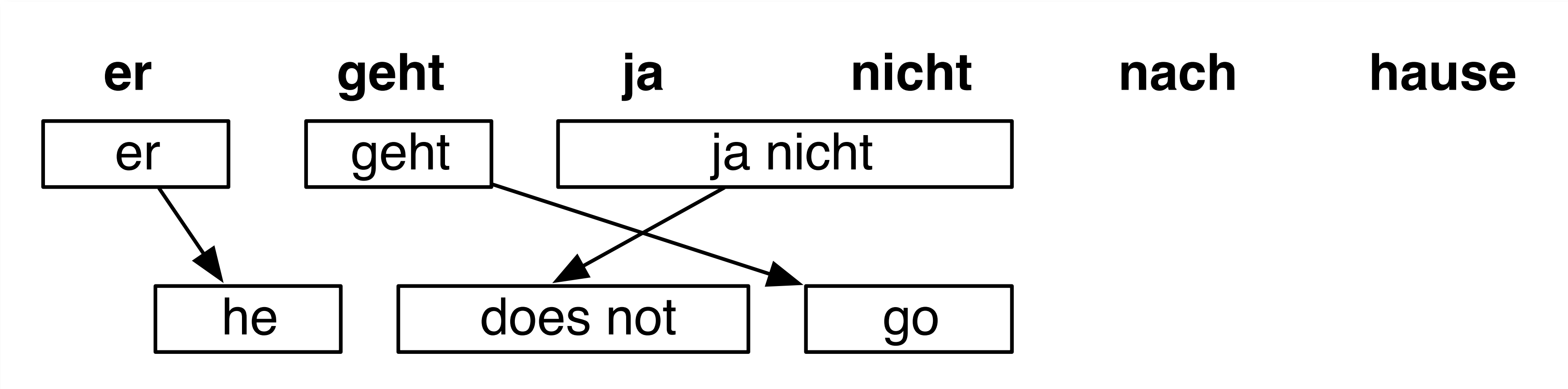
Translating a Sentence

- Pick a phrase and translate
  - possible skip to accommodate some features of the model
    - “negation before the verb in English”



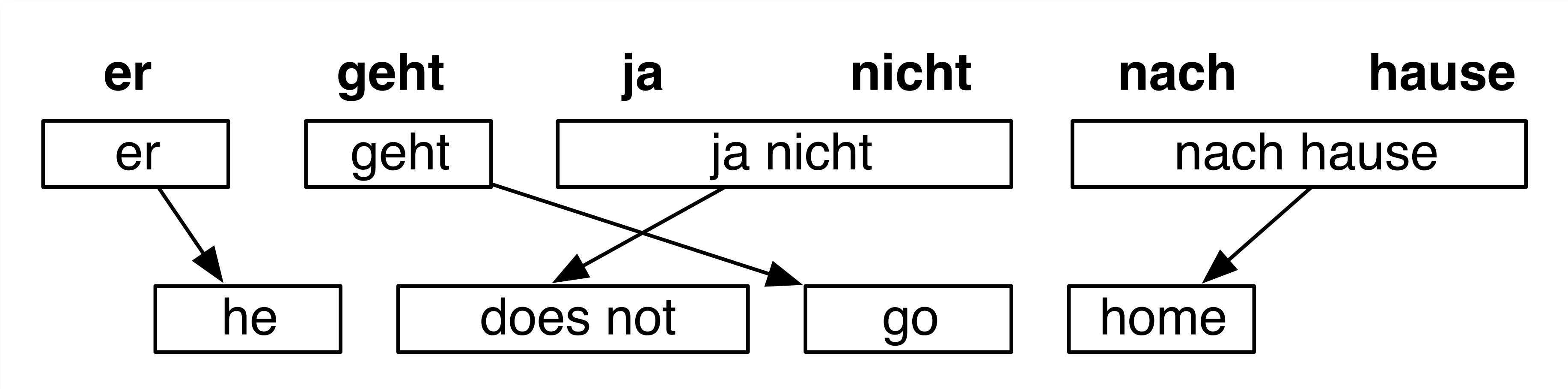
Translating a Sentence

- Go on…



Translating a Sentence

- … until every source phrase is translated



Computing Translation Probability

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

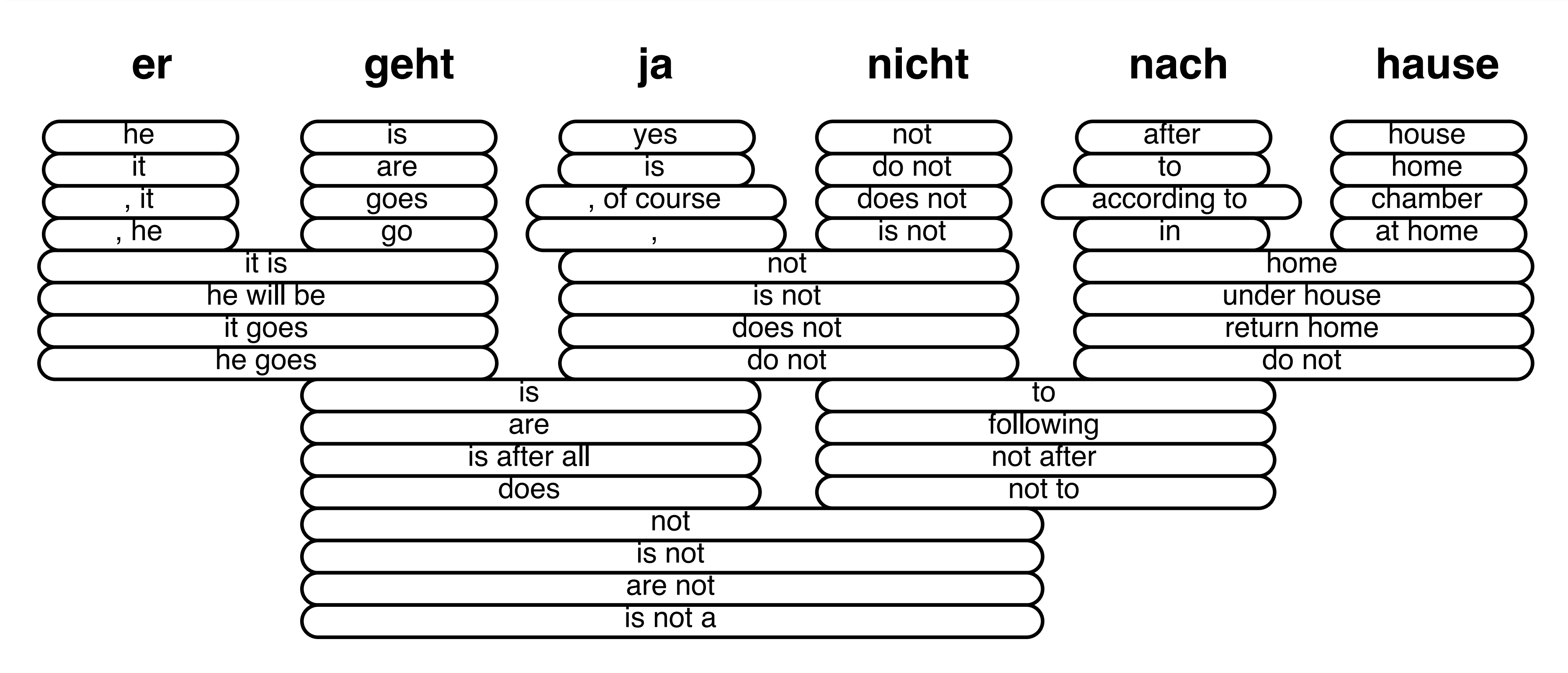
Reordering Previous phrase ended in end_{i-1}, current phrase starts at $start_i$

$\rightarrow$ compute $d(start_i - end_{i-1} - 1)$

Language model For n-gram model, need to keep track of last $n - 1$ words

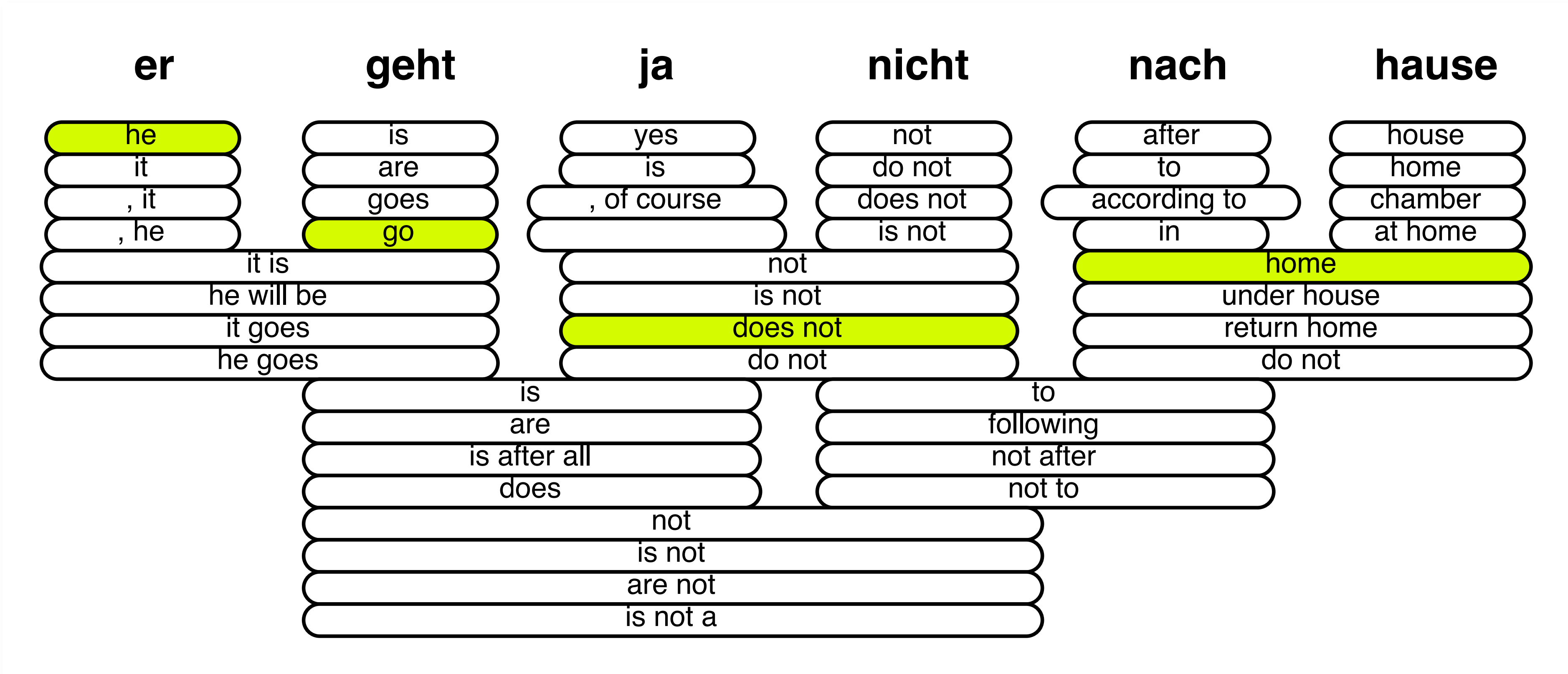
$\rightarrow$ compute score $p_{\mathrm{LM}}(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$ for added words w_i

Translation Options



- Many translation options to choose from (search graph)
  - in Europarl phrase table: 2727 matching phrase pairs for this sentence
  - by pruning to the top 20 per phrase, 202 translation options remain

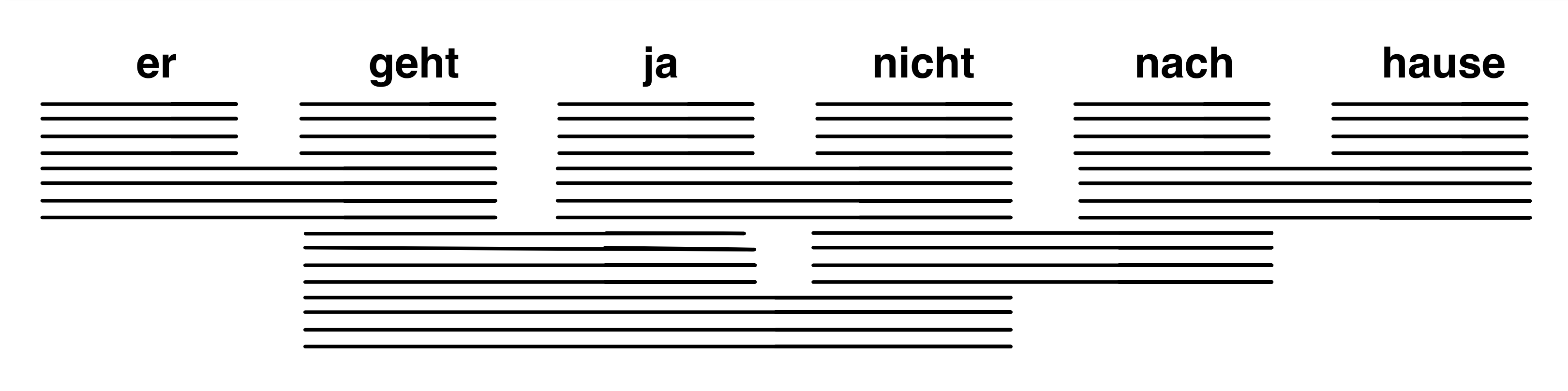
Translation Options



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

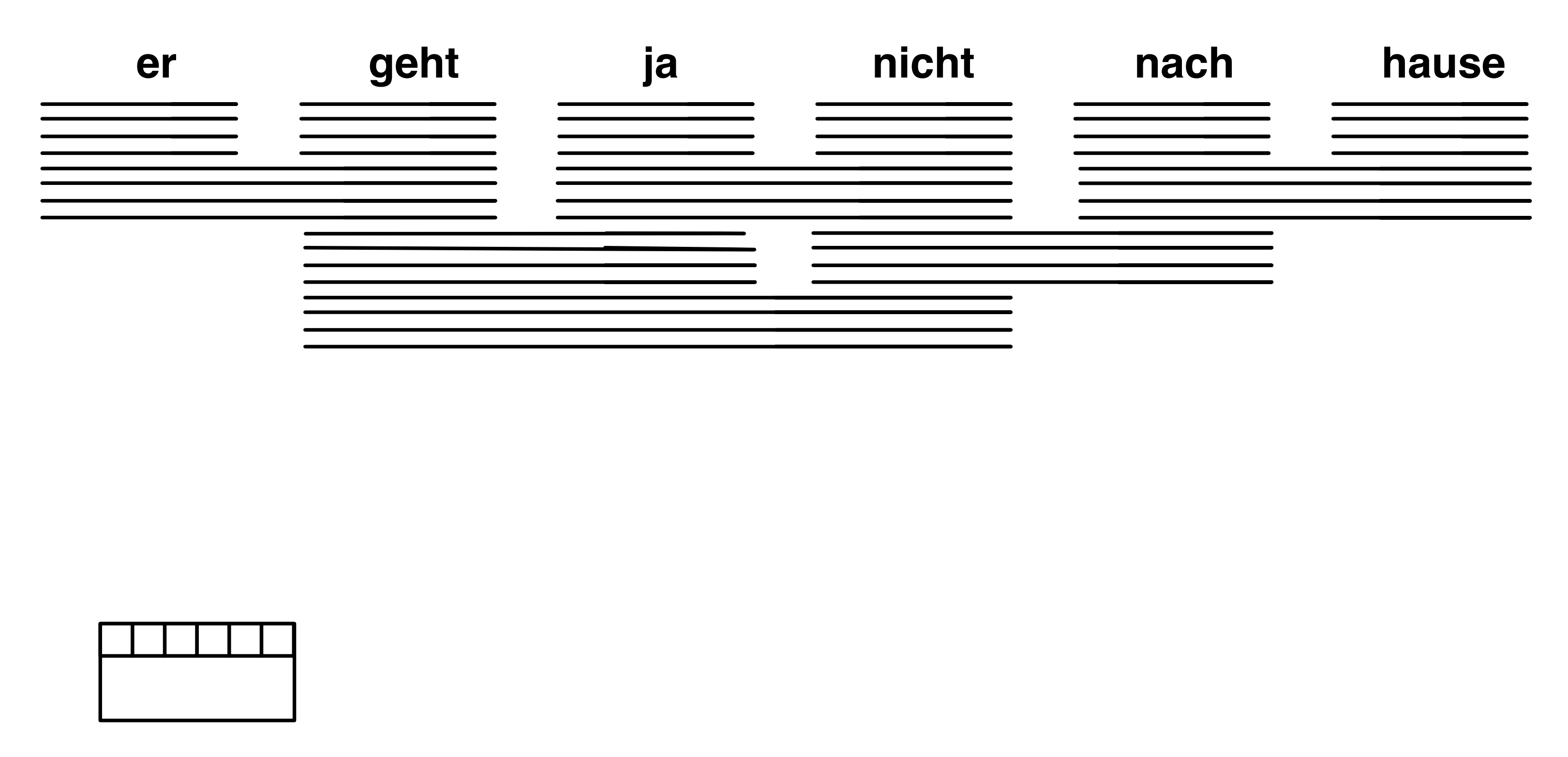
→ Search problem solved by heuristic beam search

Decoding: Precompute Translation options



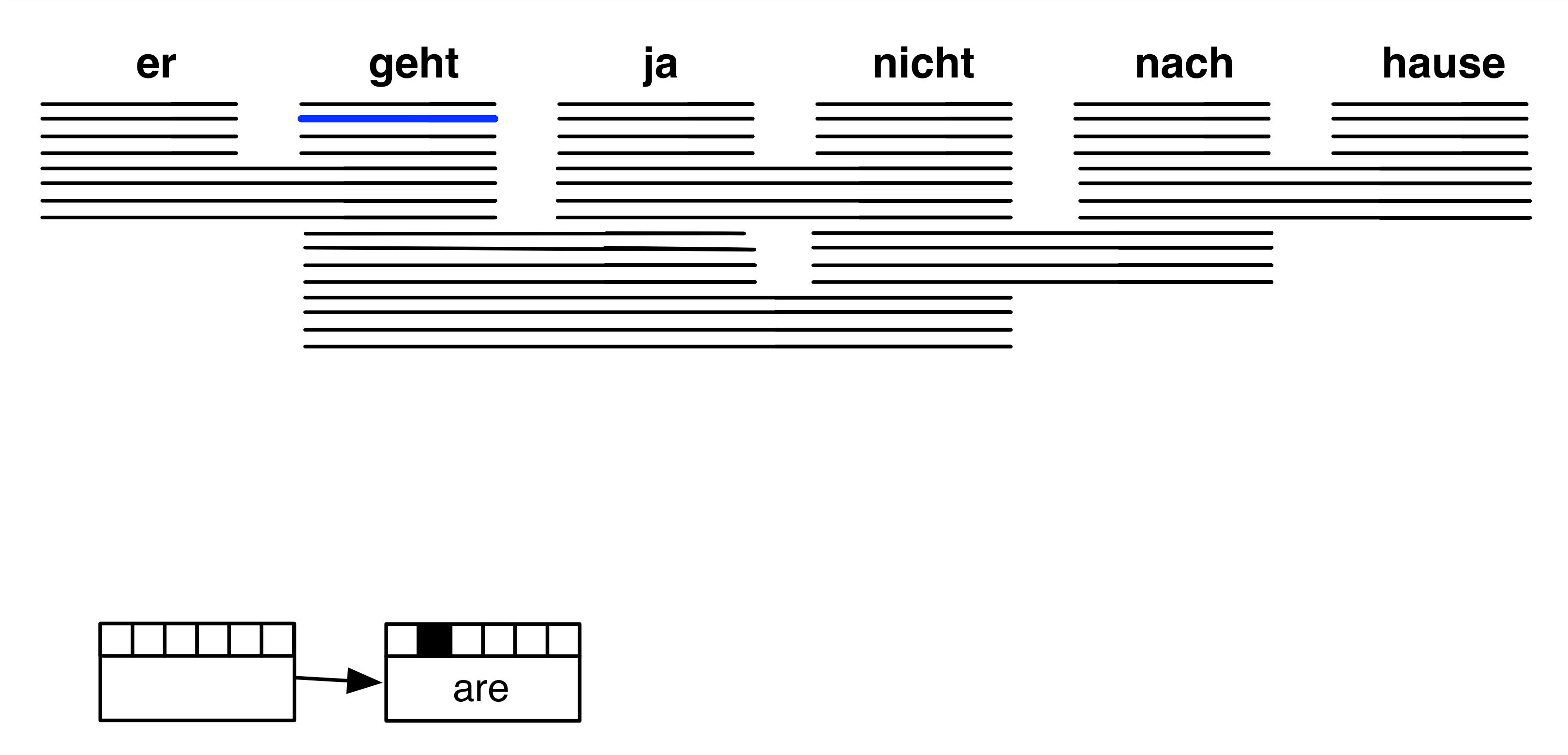
- consult phrase translation table for all input phrases

Decoding: Precompute Translation options



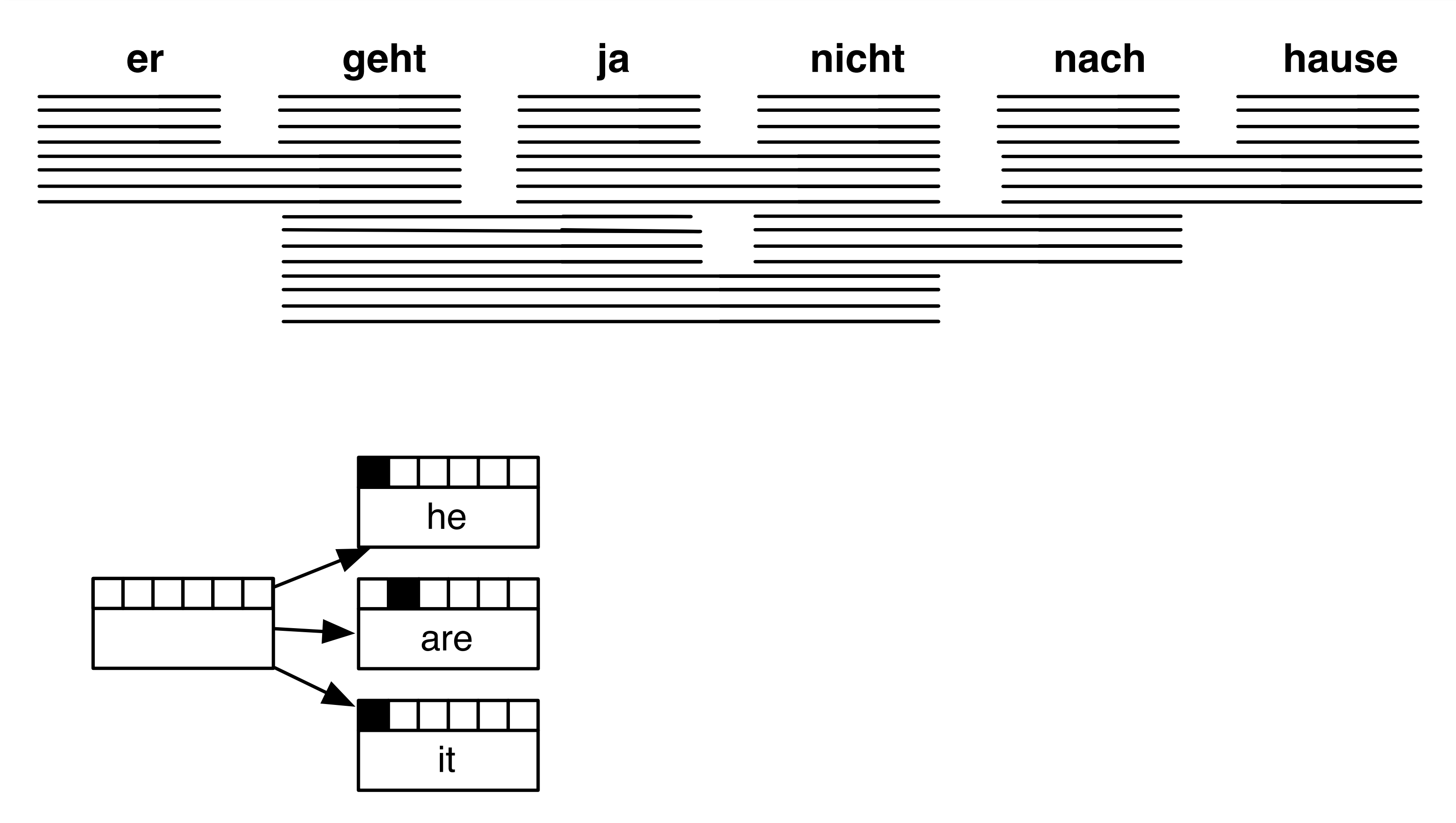
- initial hypothesis: no input words covered, no output produced

Decoding: Precompute Translation options



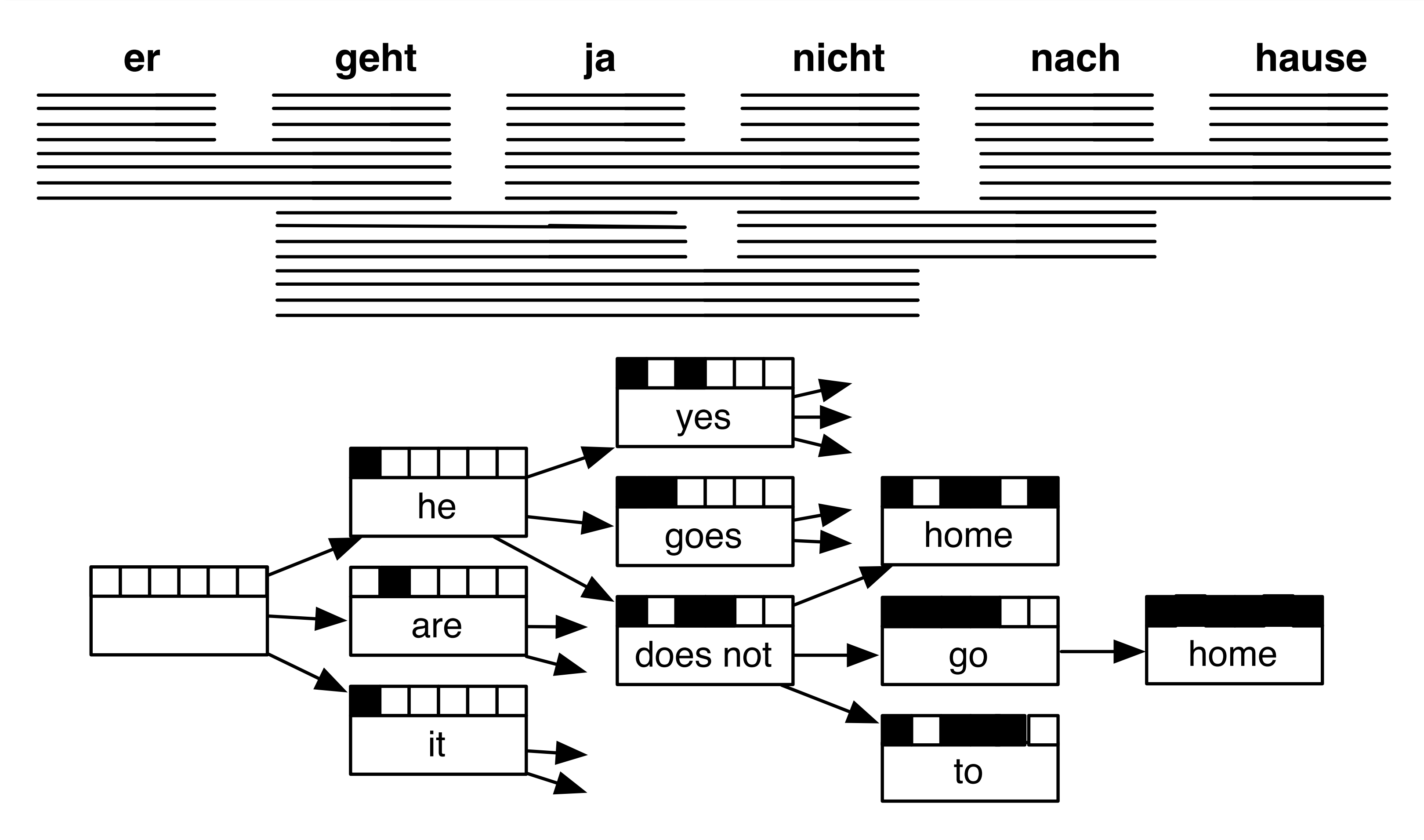
- pick any translation option, create new hypothesis

Decoding: Precompute Translation options



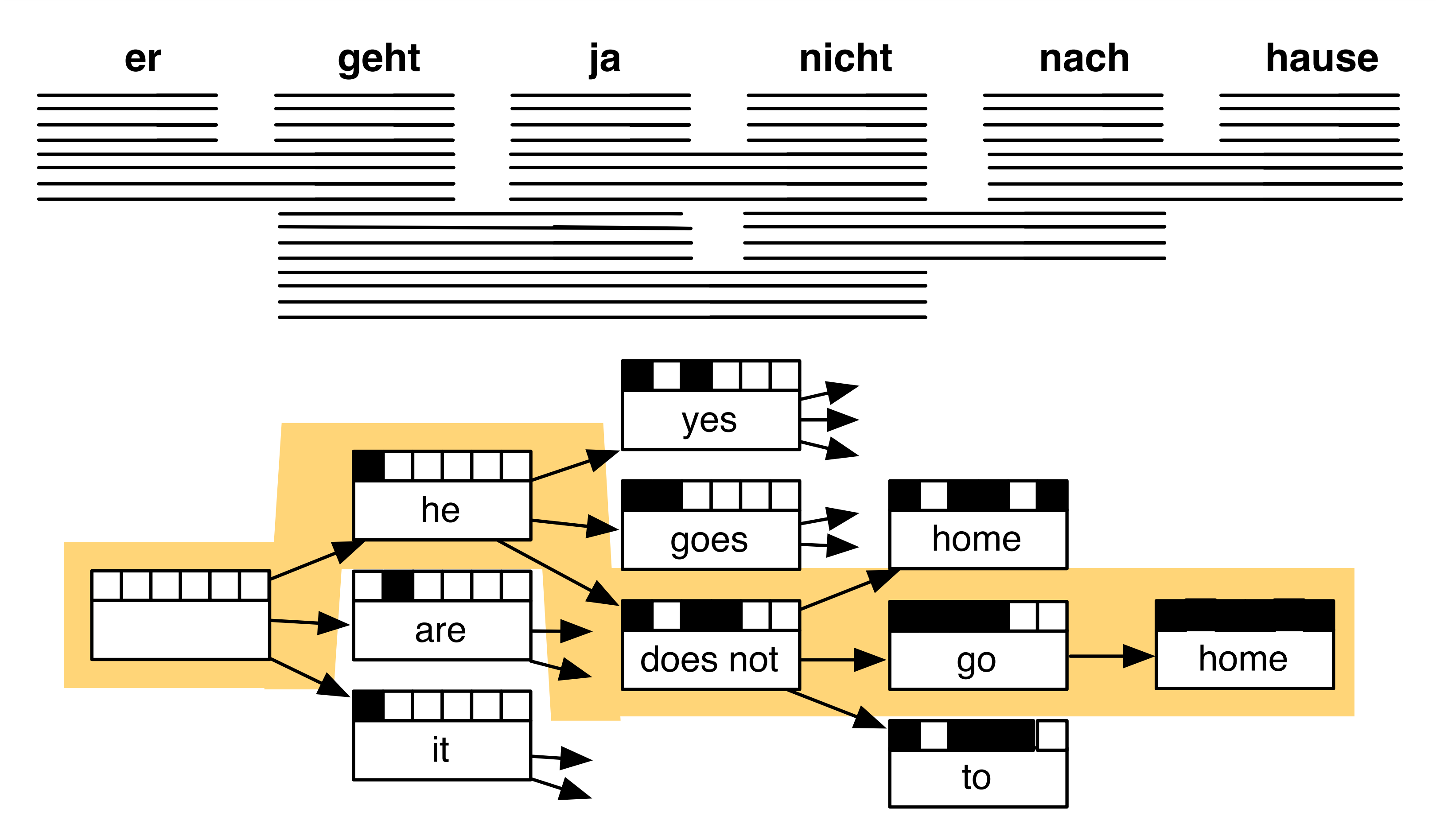
- create hypotheses for all other translation options

Decoding: Precompute Translation options



- also create hypotheses from created partial hypothesis

Decoding: Precompute Translation options



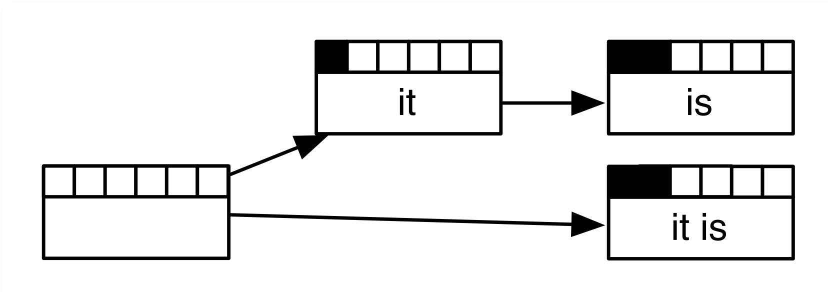
- backtrack from highest scoring complete hypothesis

Computational complexity

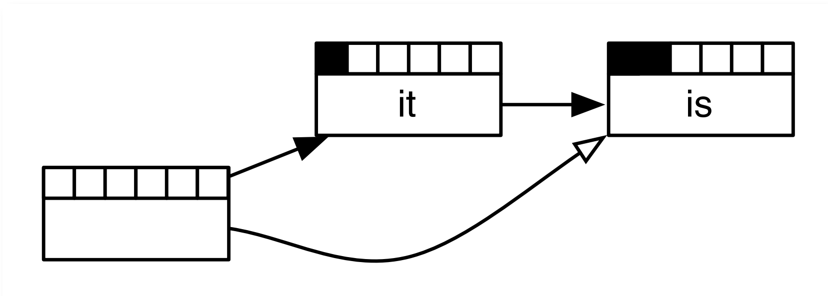
- The suggested process creates exponential number of hypothesis
- Machine translation decoding is NP-complete
- Reduction of search space:
  - recombination (risk-free)
  - pruning (risky)

Recombination

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

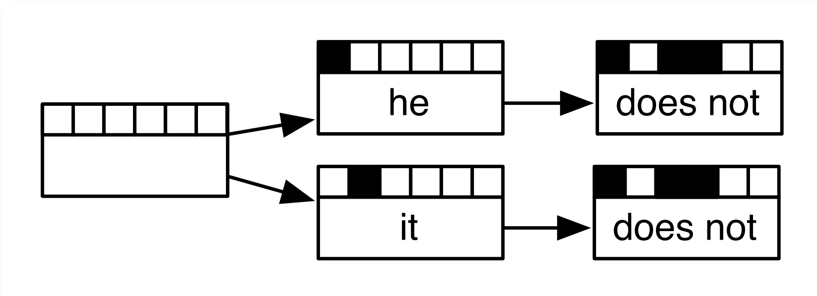


- Worse hypothesis is dropped

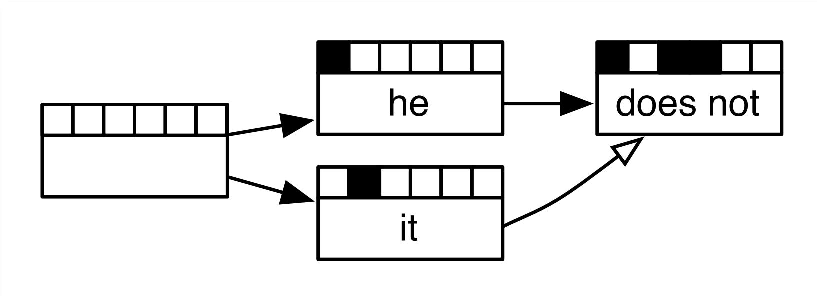


Recombination

- 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



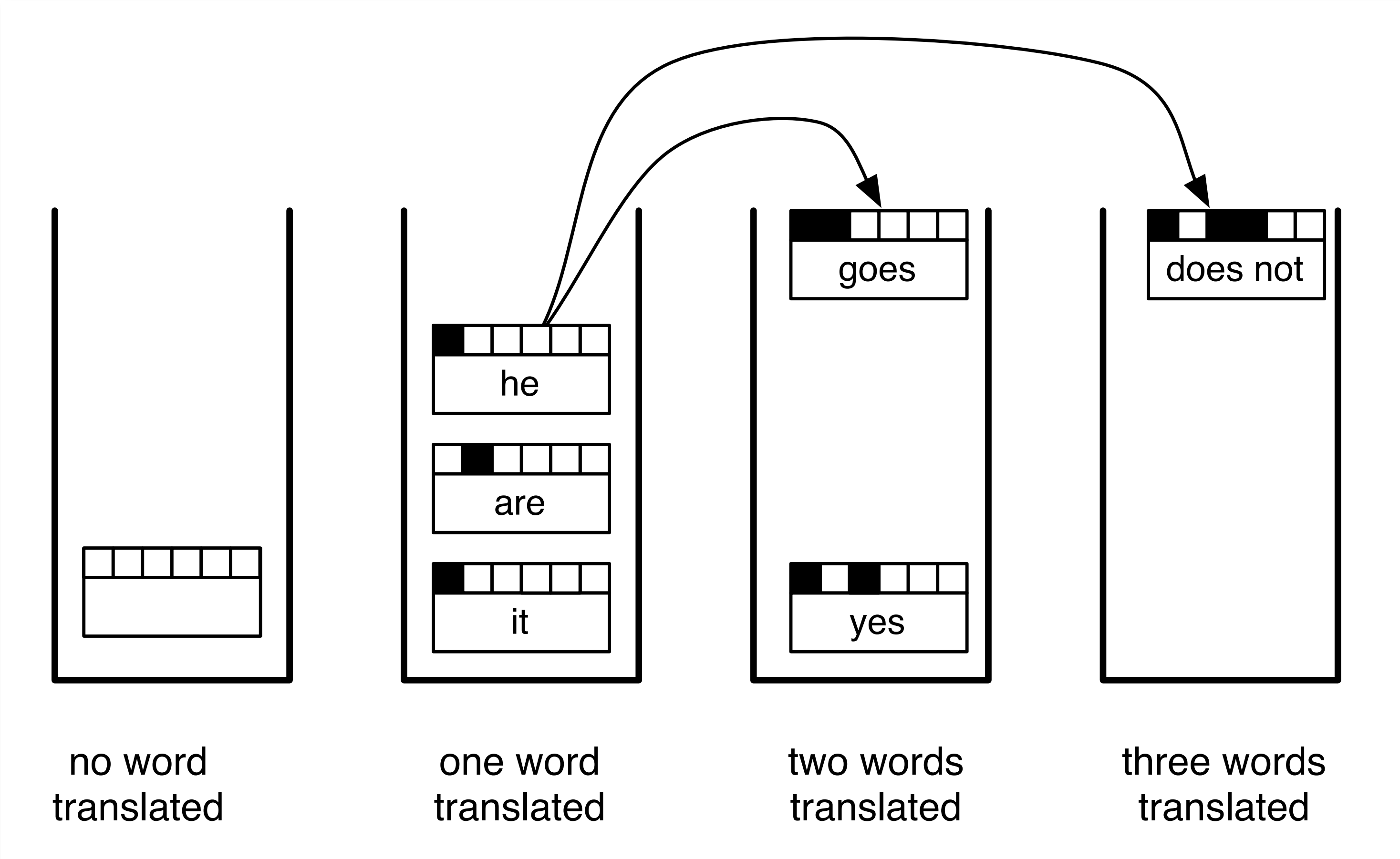
Restrictions on Recombination

- **Translation model**
  - Phrase translation independent from each other
  - $\longrightarrow$ no restriction to hypothesis recombination
- **Language model**
  - Last $n - 1$ words used as history in n-gram language model to compute the probability of word n
  - $\longrightarrow$ 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
  - $\longrightarrow$ recombined hypotheses must have that same end position
- Other feature function
  - $\longrightarrow$ may introduce additional restrictions

Pruning

- Organize hypotheses in stacks
  - same source words covered
  - same number of source words covered
  - same number of target words translated
- Compare the hypotheses in stacks ; remove the bad ones
  - **histogram pruning**: keep the k best hypotheses for each cell (eg, $n = 100$)
    - 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
  - **threshold pruning**: keep the hypotheses that have a score equal to $\alpha \times \text{best score}$ (score of the best hypothesis) ($\alpha < 1$)

Pruning: Stacks Based on previous words translated



- Hypothesis expansion in a stack decoder
  - translation option is applied to hypothesis
  - new hypothesis is dropped into a stack further down

What About Reordering?

- 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

What About Translating “Easy Phrases”?

- Balance current cost with 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

Beam Search

- Described algorithm…
  - …resembles the one of beam of light that follows the presumably best hypothesis path, but with a certain width it also illuminates neighboring hypotheses that differ not to much from the best one
  - Hence the name!
- Other algorithms for decoding
  - A* search
  - Greedy hill-climbing
  - Using finite state transducers (standard toolkits)

SUMMARY

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