PLIN019 – Machine translation

Statistical machine translation

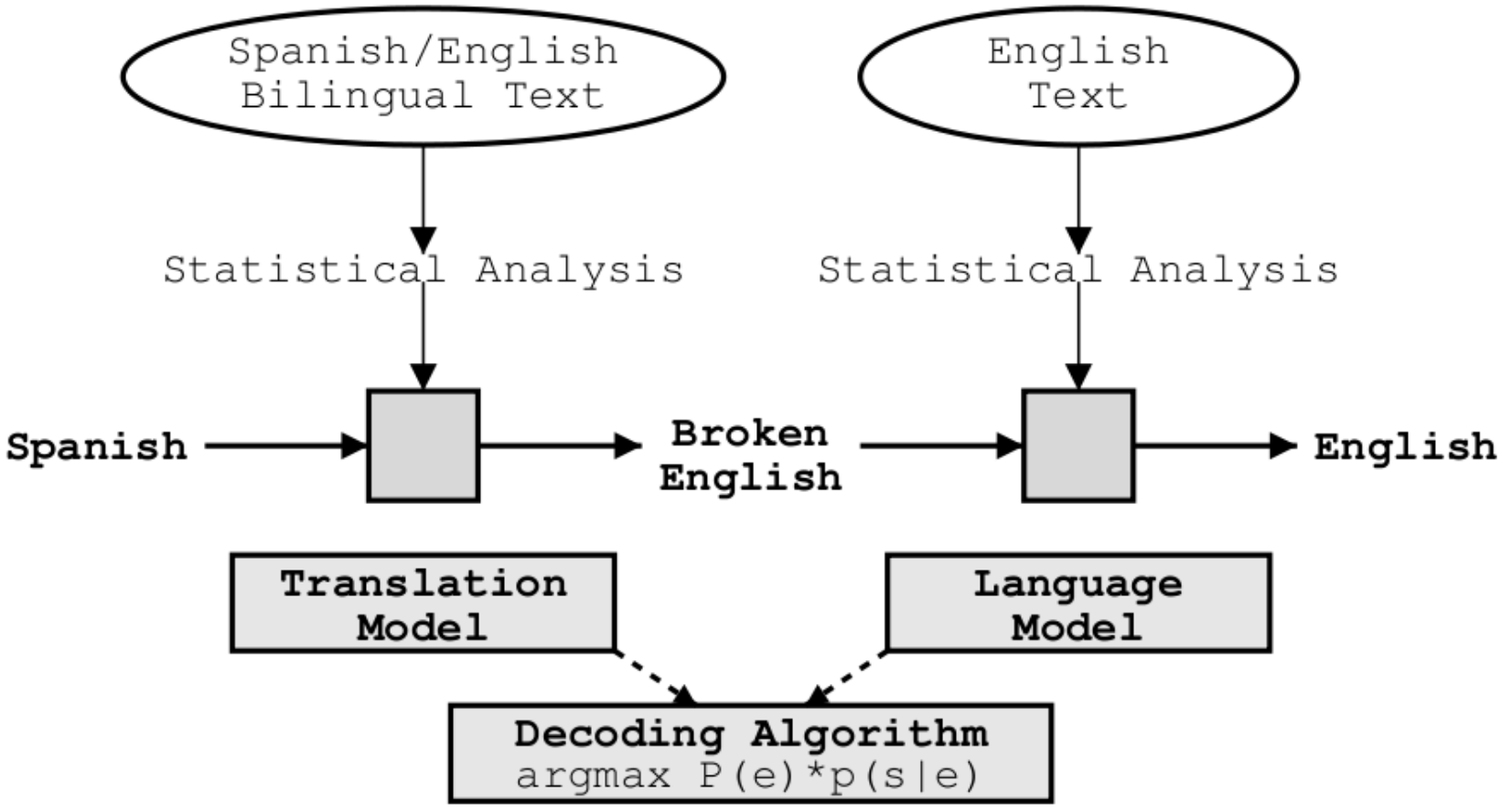
Vít Baisa

Introduction

Introduction to SMT

- rule-based systems motivated by linguistics
- SMT inspired by information theory and statistics
- currently many companies focused on SMT: Google, IBM, Microsoft
- 50 million webpages translated by SMT daily
- **gisting**: we don't need exact translation, sometimes a gist of a text is enough – the most frequent use of SMT on Internet

SMT scheme


Spanish/English
Bilingual Text
English
Text
Statistical Analysis
Statistical Analysis
Spanish
Broken
English
English
Translation
Model
Language
Model
Decoding Algorithm
argmax P(e)*p(s|e)

Tools for SMT

- GIZA++: training of IBM models, word alignment (HMM)
- SRILM: language model training
- IRST: large language model training
- Moses: phrase decoder, model training
- Pharaoh: predecessor of Moses
- Thot: phrase model training
- SAMT: tree-based models

Data for SMT – (parallel) corpora

- Linguistics Data Consorcium (LDC): parallel corpora for Arabic-English, Chinese-English etc.
  Gigaword corpus (English, 7 billion words)
- Europarl: collection of parliamentary texts of EP (11 languages, 40 M words)
- OPUS: parallel texts of various origin: subtitles, software localizations of user interfaces
- Acquis Communautaire: law documents of EU (20 lang)
- Hansards: 1.3 M pairs of text chunks from the official records of the Canadian Parliament

Regular events in SMT

Annual evaluation of SMT quality. Test set preparing, manual evaluation events, etc.

- NIST: National Institute of Standards and Technology; the oldest, prestigious; evaluation of Arabic, Chinese
- IWSLT: international workshop of spoken language translation; speech translation, Asian languages
- WMT: Workshop on SMT; mainly between European languages

Words

- for SMT (in vast majority) the basic unit = a word
- works usually with lowercased input, the original case can be restored in post-processing
- *the* makes up 7% of English texts
- top 10 words (by frequency) makes up 30% of all texts
- troublemakers: typos, numbers, proper names, loanwords

Sentences

- syntactic structure differs in various languages
- inserting of functional words typical for a given language (*the*, punctuation)
- rearranging: *er wird mit uns gehen* $\rightarrow$ *he will go with us*
- some phenomena can not be translated on the sentence level: anaphora
- level of document: theme (topic) can help with WSD
- we are not supposed to translate *bat* as *pálka* (Czech) in a text about cave animals

Parallel corpora

- basic data source for SMT
- freely available sources are about 10 to 100 M word large
- size depends heavily on a language pair
- multilingual webpages (online newspapers)
- a problem with paragraph, document alignment, . . .
- comparable corpora: texts from the same domain, not translations:
  New York Times – Le Monde
- Kapradí – corpus of Shakespeare's plays by various translators (FI+FF)
- InterCorp – manually aligned fiction books (ČNK, FF UK)

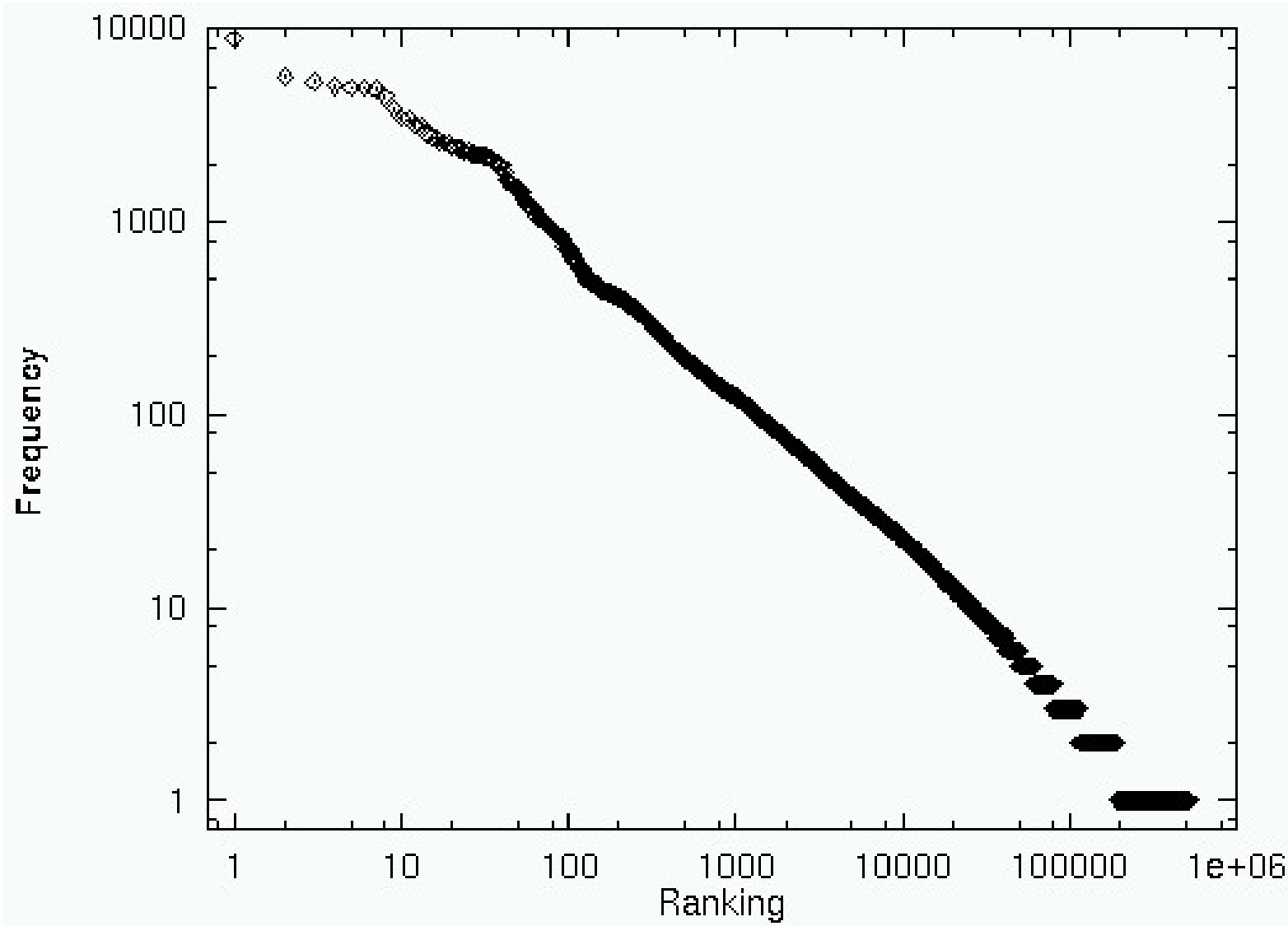
Sentence alignment

- sometimes sentences are not in 1:1 ratio in corpora
- some languages do not explicitly delimit sentence boundaries (Thai)
- *It is small, but cozy. – Es is klein. Aber es ist gemütlich.*

P	**alignment**
0.98	1:1
0.0099	1:0, 0:1
0.089	2:1, 1:2
0.011	2:2

Probability and statistics basics for SMT

Zipf's law



r rank, f = word freq., c = constant; $r \times f = c$

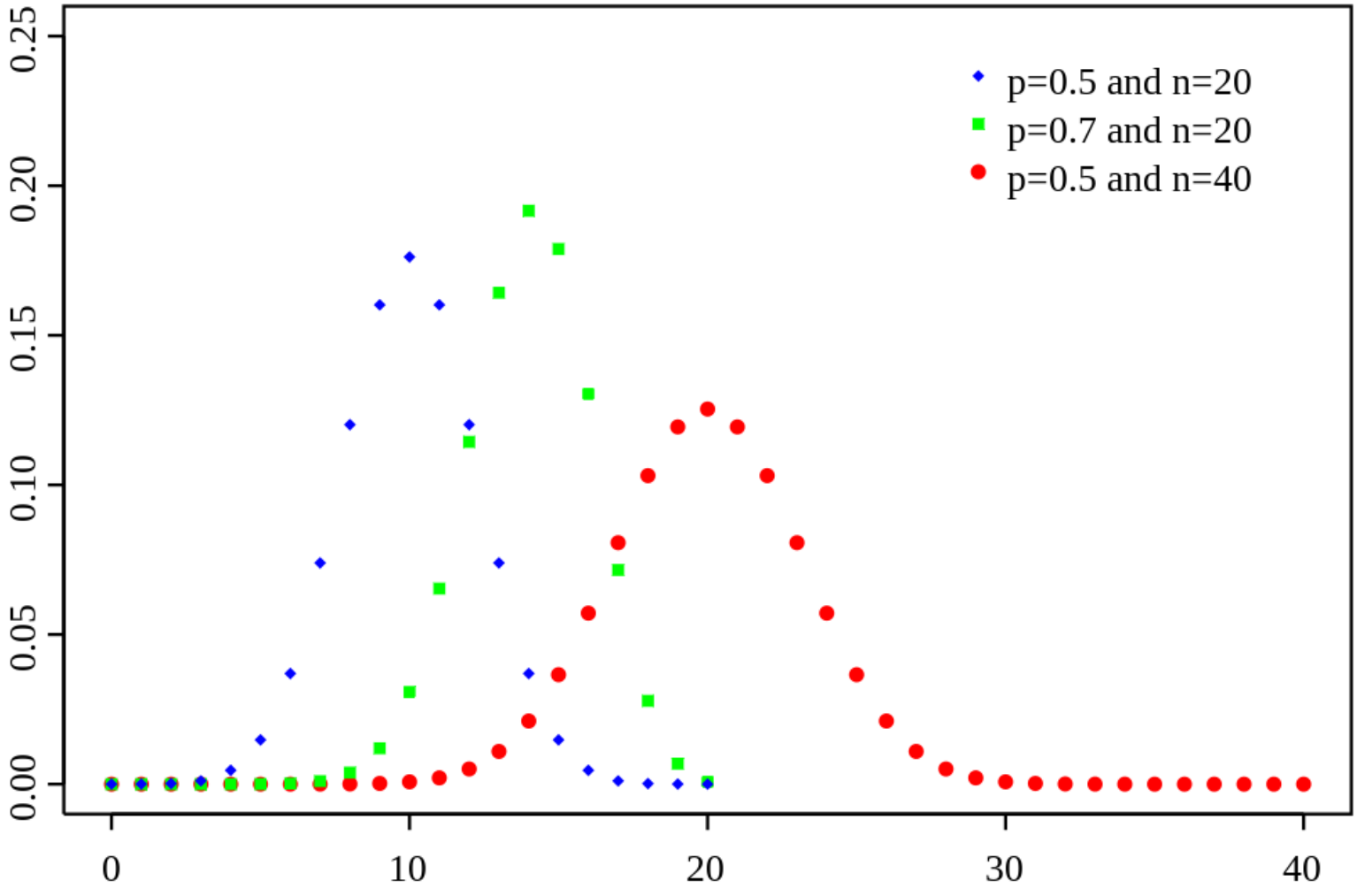
Probability distribution

- a graph of probability values for elementary phenomena of a random variable
- **uniform**: roll of dice, coin (discrete variable)
- **binomial**: multiple roll (quincunx)

$$b(n, k; p) = \binom{n}{k} p^k (1 - p)^{n-k}$$

- **normal, Gauss's**: continuous variable

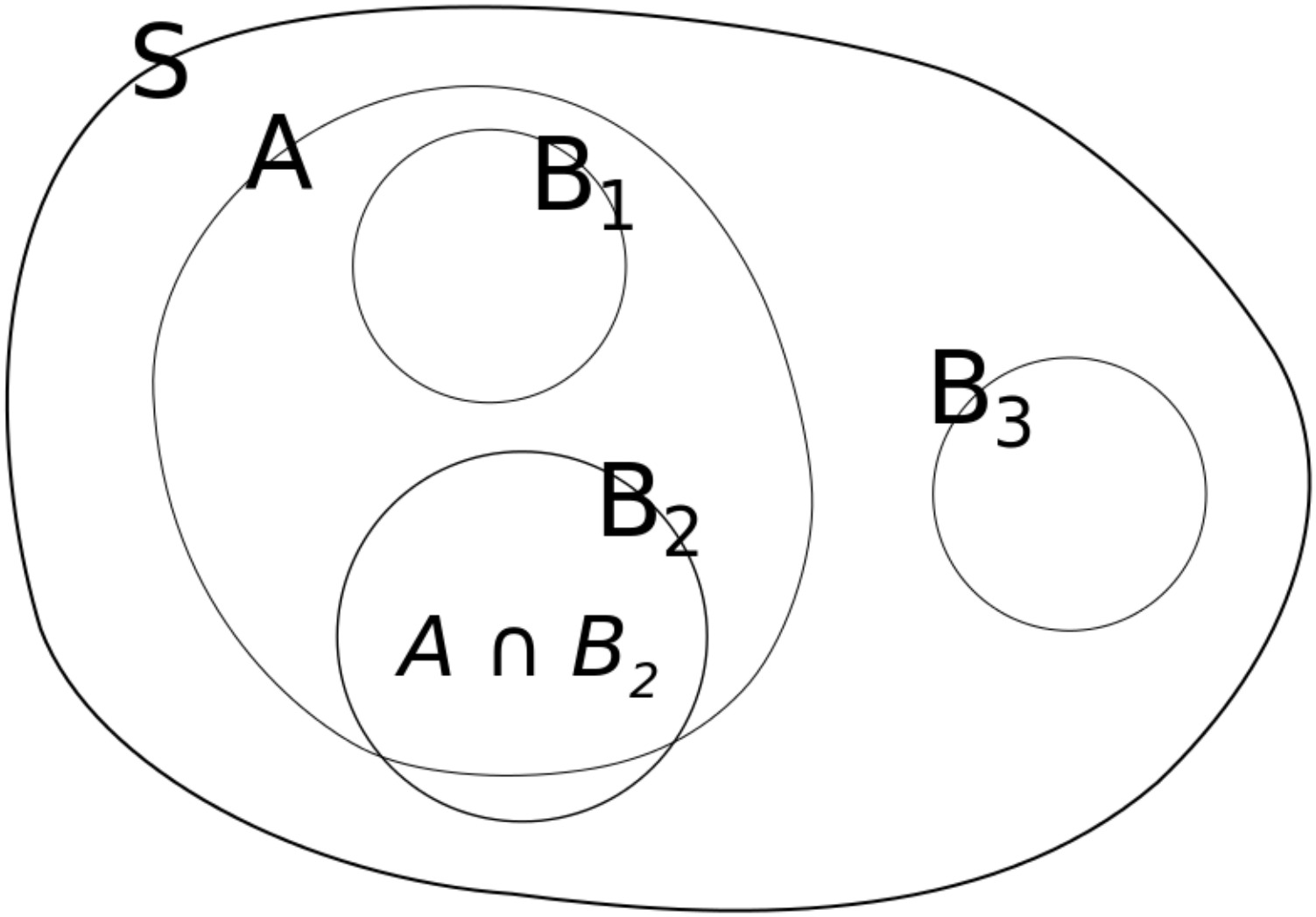
Binomial distribution



Statistics

- random variable, probability function, etc.
- we have data, we want to get distribution describing the data best
- **law of large numbers**: the more data we have the better are we able to guess its probability distribution
- e.g.: roll of a loaded dice; π calculation
- independent variables: $\forall x, y : p(x, y) = p(x).p(y)$
- **joint probability**: roll of dice and coin: $p(6, \text{heads})$
- **conditional probability**: $p(y|x) = \frac{p(x,y)}{p(x)}$
  for independent variables: $p(y|x) = p(y)$

Conditional probability



Shannon’s game

Probabilty distribution for a next letter in a text depends on previous letters.

Some letters bear more information than the others

Bayes's rule

$$p(x|y) = \frac{p(y|x).p(x)}{p(y)}$$

- example with the dice
- $p(x)$ – prior
- $p(y|x)$ – posterior

SMT – noisy channel principle

Designed by Claude Shannon (1948) for self-correcting codes, for coded signals corrections transfered through noisy channels based on information about the original data (probabilities) and about types of errors made in the channels.

Example with OCR. Optical Character Recognition is erroneous but we can estimate what was damaged in a text (with a language model); errors I$\leftrightarrow$1$\leftrightarrow$l, rn$\leftrightarrow$m etc.

$$
\begin{aligned}
e^* &= \arg\max_e p(e|f) \\
&= \arg\max_e \frac{p(e)p(f|e)}{p(f)} \\
&= \arg\max_e p(e)p(f|e).
\end{aligned}
$$

SMT components

- language models:
  - how we get $p(e)$ for any string e
  - the more e looks like proper language the higher is $p(e)$
  - issue: what is $p(e)$ for an unseen e?
- translation model:
  - for e and f compute $p(f|e)$
  - the more e looks like a proper translation f, the higher $p(f|e)$
- decoding algorithm
  - based on TM and LM, find a sentence f as the best translation of e
  - as fast as possible and with as few memory as possible

Language models

Language models

Noam Chomsky, 1969

But it must be recognized that the notion “probability of a sentence” is an entirely useless one, under any known interpretation of this term.

Fred Jelinek, 1988, IBM

Anytime a linguist leaves the group
the recognition rate goes up.

***

What is the probability of utterance of *s*?

Ke snídani jsem měl celozrnný ...

Language models

Noam Chomsky, 1969

But it must be recognized that the notion “probability of a sentence” is an entirely useless one, under any known interpretation of this term.

Fred Jelinek, 1988, IBM

Anytime a linguist leaves the group
the recognition rate goes up.

***

What is the probability of utterance of *s*?

Ke snídani jsem měl celozrnný ...

chléb $>$ pečivo $>$ zákusek $>$ mléko $>$ babičku

Language models – probability of a sentence

- we look for a probability of a following word
- a LM is probability distribution over all possible word sequences

Probability of word sequences

p_{LM}(včera jsem jel do Brna)
p_{LM}(včera jel do Brna jsem)
p_{LM}(jel jsem včera do Brna)

Language models for fluency and WSD

- LMs help ensure fluent output (proper word order)
- LMs help with WSD in general cases
- if a word is polysemous, we can choose the most frequent translation (*pen* $\rightarrow$ pero)
- in special cases can not be used but
- LMs help with WSD using a context
- p_{LM}(i go home) $\geq$ p_{LM}(i go house)

N-gram models

- n-gram is very useful concept in NLP
- it uses statistical observation of input data
- two applications in machine translation:
  - what follows after *I go*? *home* is more frequent than *house*
  - *I go to home* $\times$ *I go home*
- language (random) generation:

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have Every enter now severally so, let. **[1-grams]**

Sweet prince, Falstaff shall die. Harry of Monmouth's grave. This shall forbid it should be branded, if renown made it empty. **[3-grams]**

Can you guess the author of the original text?

N-gram models – naive approach

$W = w_1, w_2, \cdots, w_n$

How we compute $p(W)$? Occurrences of all W in data normalized by data size. For majority of W we won't have any occurrence in our data. The goal is to generalize observed properties of (training) data which is usually sparse (sparse data).

$$P(\text{chléb}|\text{ke snídani jsem měl celozrnný}) = \frac{|\text{ke snídani jsem měl celozrnný chléb}|}{|\text{ke snídani jsem měl celozrnný}|}$$

Markov's chain, Markov's assumption

$p(W)$, where W is sequence of words; we model the probability word by word using **rule of chain**:

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

Since p is not available for longish word sequences, we limit the history to m words using **Markov's assumption**

$$p(w_n|w_1, w_2, \ldots w_{n-1}) \simeq p(w_n|w_{n-m}, \ldots w_{n-2}, w_{n-1})$$

It means that for estimation of a following word it is sufficient to have m ($m-1$) preceding words. m is order of a model. Usually, trigrams are used.

Computing, LM probabilities estimation

Trigram model uses 2 preceding words for probability learning.
Using **maximum-likelihood estimation**:

$$p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)}$$

trigram: *(the, green, w)* (1748)

w	**count**	$p(w)$
paper	801	0.458
group	640	0.367
light	110	0.063
party	27	0.015
ecu	21	0.012

Quality and comparison of LMs

We need to compare quality of various LM (various orders, various data, smoothing techniques etc.)

2 approaches: extrinsic and intrinsic evaluation.

A good LM should assign a higher probability to a good (looking) text than to an incorrect text. For a fixed testing text we can compare various LMs.

Cross-entropy

$$H(p_{LM}) = -\frac{1}{n} \log p_{LM}(w_1, w_2, \ldots w_n)$$
$$= -\frac{1}{n} \sum_{i=1}^{n} \log p_{LM}(w_i | w_1, \ldots w_{i-1})$$

Cross-entropy is average value of negative logarithms of words probabilities in testing text. It corresponds to measure of uncertainty of a probability distribution. The lower the better.

A good LM should reach entropy close to real entropy of language. That can not be measured but quite reliable estimates do exist, e.g. Shannon's game. For English, entropy is estimated to approx. 1.3 bit per letter.

Perplexity

$$PP = 2^{H(p_{LM})}$$

$$PP(W) = p_{LM}(w_1 w_2 w_3 \ldots w_N)^{-\frac{1}{N}}$$

Perplexity is simple transformation of cross-entropy.

A good LM should not waste p for improbable phenomena.

The lower entropy, the better $\rightarrow$ the lower perplexity, the better.

Language models smoothing

Issue: if an n-gram is missing in the data but is in $w \rightarrow p(w) = 0$.

We need to distinguish p also for *unseen data*. It must hold:

$$\forall w.p(w) > 0$$

The issue is more serious for high-order models.

Smoothing: attempt to amend real counts of n-grams to expected counts in any data (different corpora).

Add-one smoothing (Laplace)

Maximum likelihood estimation assigns p based on

$$p = \frac{c}{n}$$

Add-one smoothing uses

$$p = \frac{c + 1}{n + v}$$

where v is amount of all possible n-grams. That is quite inaccurate since all permutations might outnumber real (possible) n-grams by several magnitudes.

Europarl has (unique) 139,000 words, so 19 bilion possible bigrams. In fact it has only 53 mil. tokens, so maximally 53 mil. bigrams. **Why?**

This smoothing overvalues unseen n-grams.

Add-α smoothing

We won't add 1, but α. This can be estimated for the smoothing to be the most just and balanced.

$$p = \frac{c + \alpha}{n + \alpha v}$$

α can be obtained experimentally: we can try several different values and find the best one using perplexity.

Usually it is very small (0.000X).

Deleted estimation

We can find unseen n-grams in another corpus. N-grams contained in one of them and not in the other help us to estimate general amount of unseen n-grams.

E.g. bigrams not occurring in a training corpus but present in the other corpus million times (given the amount of all possible bigrams equals 7.5 billions) will occur approx.

$$\frac{10^6}{7.5 \times 10^9} = 0.00013\times$$

Good–Turing smoothing

We need to amend occurrence counts (frequencies) of n-grams in a corpus in such a way they correspond to general occurrence in texts. We use *frequency of frequencies*: number of various n-grams which occurr $n\times$.

We use frequency of hapax legomena (singletons in data) to estimate unseen data.

$$r^* = (r+1)\frac{N_{r+1}}{N_r}$$

Especially for n-grams not in our corpus we have

$$r_0^* = (0+1)\frac{N_1}{N_0} = 0.00015$$

where $N_1 = 1.1 \times 10^6$ a $N_0 = 7.5 \times 10^9$ (Europarl).

Example of Good–Turing smoothing (Europarl)

r	**FF**	r^*
0	7,514,941,065	0.00015
1	1,132,844	0.46539
2	263,611	1.40679
3	123,615	2.38767
4	73,788	3.33753
5	49,254	4.36967
6	35,869	5.32929
8	21,693	7.43798
10	14,880	9.31304
20	4,546	19.54487

Comparing smoothing methods (Europarl)

method	**perplexity**
add-one	382.2
add-α	113.2
deleted est.	113.4
Good–Turing	112.9

Interpolation and back-off

Previous methods treated all unseen n-grams the same.
Consider trigrams

nádherná červená řepa
nádherná červená mrkev

Despite we don’t have any of these in our training data, the former trigram should be probably more probable.

We will use probability of lower order models, for which we have necessary data:

červená řepa
červená mrkev
nádherná červená

Interpolation

$$p_I(w_3|w_1 w_2) = \lambda_1 p(w_3) \times \lambda_2 p(w_3|w_2) \times \lambda_3 p(w_3|w_1 w_2)$$

If we have enough data we can trust higher order models more and assign a higher significance to corresponding n-grams.

p_I is probability distribution, thus this must hold:

$$\forall \lambda_n : 0 \leq \lambda_n \leq 1$$

$$\sum_n \lambda_n = 1$$

Large LM – n-gram counts

How many unique n-grams are in a corpus?

order	unique	singletons
unigram	86 700	33 447 (38,6 %)
bigram	1 948 935	1 132 844 (58,1 %)
trigram	8 092 798	6 022 286 (74,4 %)
4-gram	15 303 847	13 081 621 (85,5 %)
5-gram	19 882 175	18 324 577 (92,2 %)

Taken from Europarl with 30 mil. tokens.

Lexical translation

Standard lexicon does not contain information about frequency of translations of individual meanings of words.

key $\rightarrow$ *klíč, tónina, klávesa*

How often are the individual translations used in translations?

key $\rightarrow$ *klíč* (0.7), *tónina* (0.18), *klávesa* (0.12)

We need lexical probability distribution p_f with property:

$$\sum_e p_f(e) = 1$$

$$\forall e : 0 \leq p_f(e) \leq 1$$

$$p_{\text{klíč}}(\text{key}) \; ? \; p_{\text{mrkev}}(\text{carrot})$$

Word alignment, alignment function

Translations differ in number of words and in word order. SMT uses *alignment function*

$$a : j \rightarrow i$$

where j is position of particular word in target sentence (Czech), i is position in a source sentence (English).

i:	1	2	3	4	5
	the	castle	is	very	old
	ten	hrad	je	velmi	starý
j:	1	2	3	4	5

a is function, therefore for each word w_e from the target sentence there is exactly one word w_f from the source sentence.

Word alignment – more examples

- different word order:
  it was written here
  bylo to zde napsané
  $a : 1 \rightarrow 2, 2 \rightarrow 1, 3 \rightarrow 4, 4 \rightarrow 3$
- different word number:
  jsem maličký
  i am very small
  $a : 1 \rightarrow 1, 2 \rightarrow 1, 3 \rightarrow 2, 4 \rightarrow 2$
- no translational equivalents:
  have you got it ?
  máš to ?
  $a : 1 \rightarrow 1, 2 \rightarrow 4, 3 \rightarrow 5$
- the opposite case, we add token NULL, pozice 0:
  NULL laugh
  smát se
  $a : 1 \rightarrow 1, 2 \rightarrow 0$

IBM model 1

p_f for individual sentences is not available. We split translation into several steps, using p_f for words. This approach is called *generative modeling*.

Translation model IBM-1 is defined as follows:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

where $\mathbf{e} = (e_1, \ldots e_{l_e})$ is target sentence, $\mathbf{f} = (f_1, \ldots f_{l_f})$ source sentence, l_e is target sentence length, l_f source sentence length, ϵ is normalizing constant, for resulting product to be proper probability distribution. $(l_f + 1)^{l_e}$ is number of all possible alignments between $\mathbf{e}$ a $\mathbf{f}$, whereas l_f is increased by 1 due to NULL, t is probability translation function.

Translation probability computation

For computing $p(\mathbf{e}, a|\mathbf{f})$ we need to know value of t for all words.

The basic resource for SMT will be used: **parallel corpus** with aligned sentences.

Unfortunately we don't have word-level alignment (CEMAT). It is task for **word-alignment** and it is time for **expectation-maximization (EM)** algorithm.

EM algoritmus

1. initialize the model (usually uniform distribution)
2. apply model to the data (expectation step)
   we are looking for $p(a|e,f) = \frac{p(e,a|f)}{p(e|f)}$
   where $p(e|f) = \sum_a p(e,a|f)$
3. adjust model according to the data (maximization step)
   we amend word alignments count w_e to w_f (function c)
   according previous
   $c(w_e|w_f;e,f) = \sum_a p(a|e,f) \sum_{j=1}^{l_e} \delta(e,e_j)\delta(f,f_{a(j)})$
   where $\delta(x,y) = 1 \iff x == y$, else 0
4. repeat E-M steps until the model can be improved

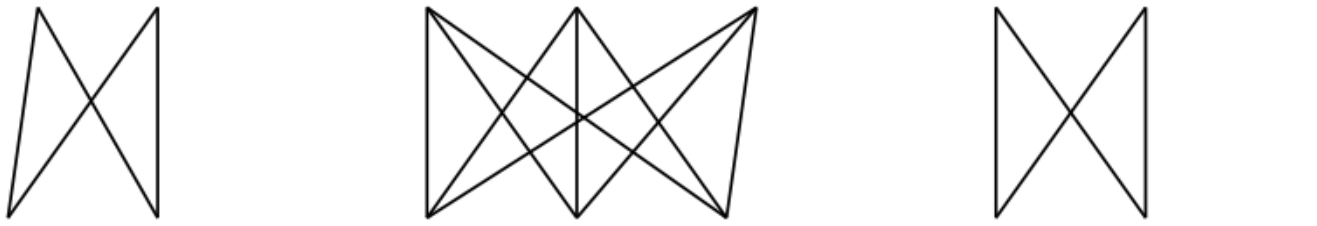
Translation probability from EM algorithm

The resulting translation probability is computed using c:

$$t(w_e|w_f) = \frac{\sum_{(e,f)} c(w_e|w_f; e, f)}{\sum_{w_e} \sum_{(e,f)} c(w_e|w_f; e, f)}$$

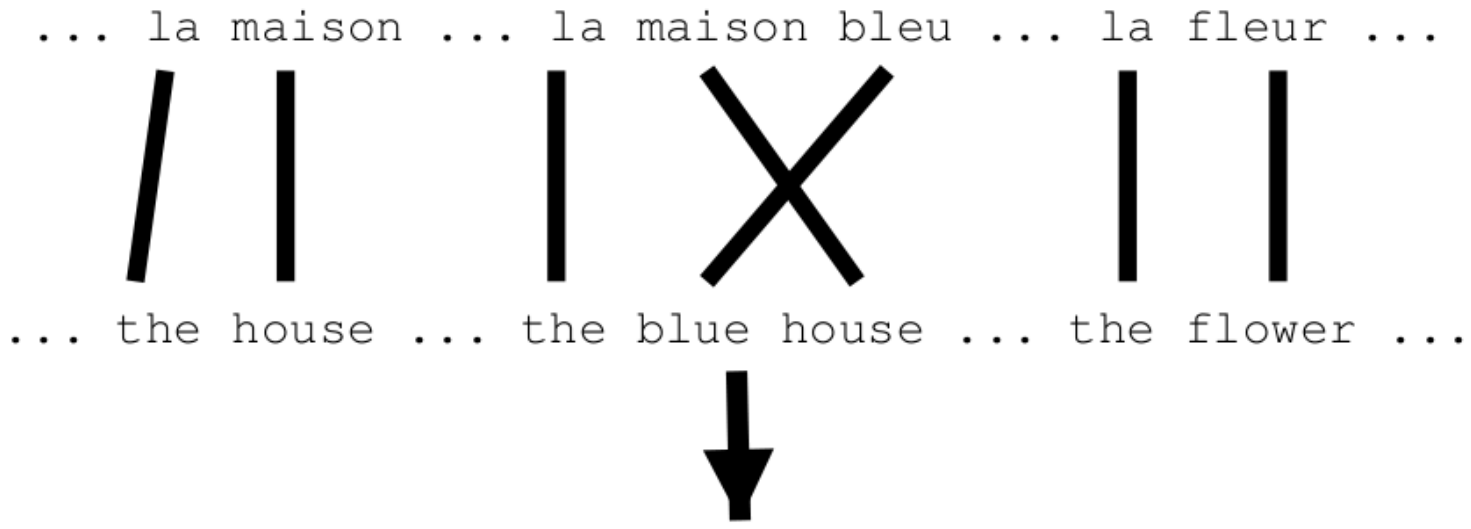
EM algorithm – initialization

... la maison ... la maison blue ... la fleur ...



... the house ... the blue house ... the flower ...

EM algorithm – final phase



p(la|the) = 0.453
p(le|the) = 0.334
p(maison|house) = 0.876
p(bleu|blue) = 0.563
...

IBM models

IBM model 1 is very simple. It does not take context into account, can not add and skip words. All alignments are of the same probability. Each of following models adds something more to the previous.

- IBM-1: lexical translation
- IBM-2: + absolute alignment model
- IBM-3: + **fertility** model
- IBM-4: + relative alignment model
- IBM-5: + treats shortcomings of the previous models

IBM-2

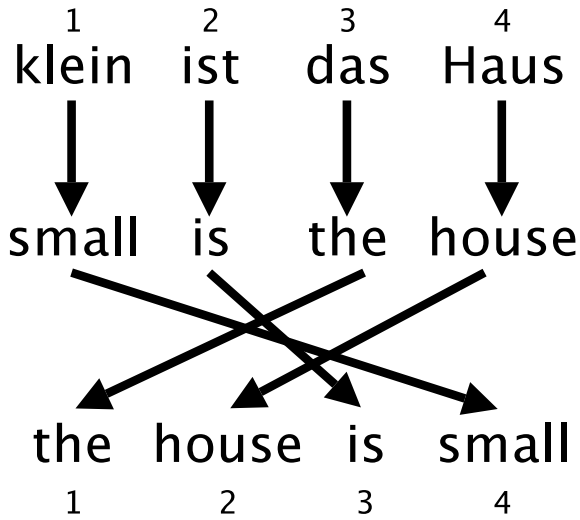
In IBM-1, all translations in different word order are of the same probability. IBM-2 adds explicit model of alignment: **alignment probability distribution**:

$$a(i|j, l_w, l_f)$$

where i is source word position, j target word position.

IBM-2 – 2 kroky překladu

Translation is split to two steps. In the first, lexical units are translated and in the second, words are rearranged according to the model.



lexical translation step

alignment step

IBM-2

The first step is the same as in IBM-1, $t(e|f)$ is used. Function a with probability distribution a is in the opposite direction to the translation direction. Both distributions are combined to formula for IBM-2:

$$p(e, a|f) = \epsilon \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) a(a(j)|j, l_e, l_f)$$

$$\begin{aligned} p(e|f) &= \sum_a p(e, a|f) \\ &= \epsilon \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i) a(i|j, l_e, l_f) \end{aligned}$$

IBM model 3

Models 1 and 2 don't consider situations where one word is translated to two and more words, or is not translated at all. IBM-3 solves this with **fertility**, which is modelled with a probability distribution

$$n(\phi|f)$$

For each source word f, the distribution n expresses, to how many words f is usually translated.

$$n(0|\text{a}) = 0.999$$
$$n(1|\text{king}) = 0.997$$
$$n(2|\text{steep}) = 0.25$$
$$\ldots$$

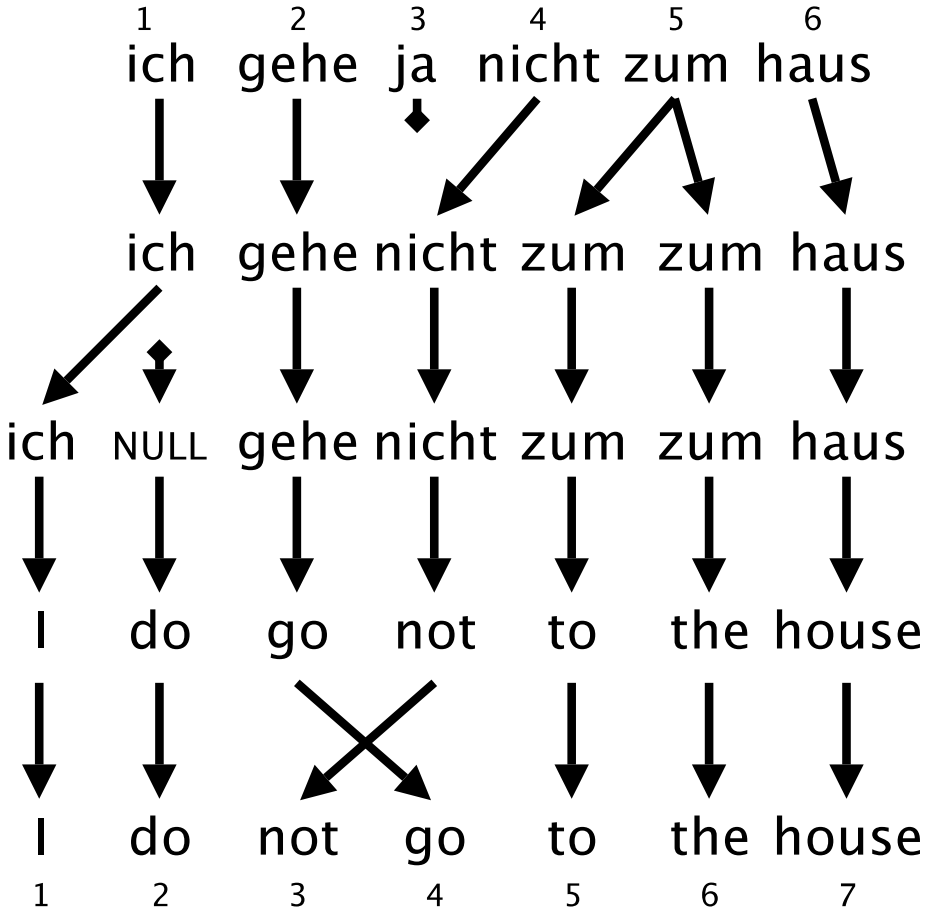
NULL token insertion

If we want to translate properly to a target language which uses words without translational equivalents we have to tackle with inserting NULL token.

$n(x|NULL)$ is not used since NULL insertion depends on a sentence length.

We add another step **NULL insertion** to the translation process. We use p_1 and $p_0 = 1 - p_1$, where p_1 stands for probability of NULL token insertion after any word in a sentence.

IBM-3


1 2 3 4 5 6
ich gehe ja nicht zum haus
ich gehe nicht zum zum haus
ich NULL gehe nicht zum zum haus
I do go not to the house
I do not go to the house
1 2 3 4 5 6 7
fertility step
NULL insertion step
lexical translation step
distortion step

IBM-3 – distortion

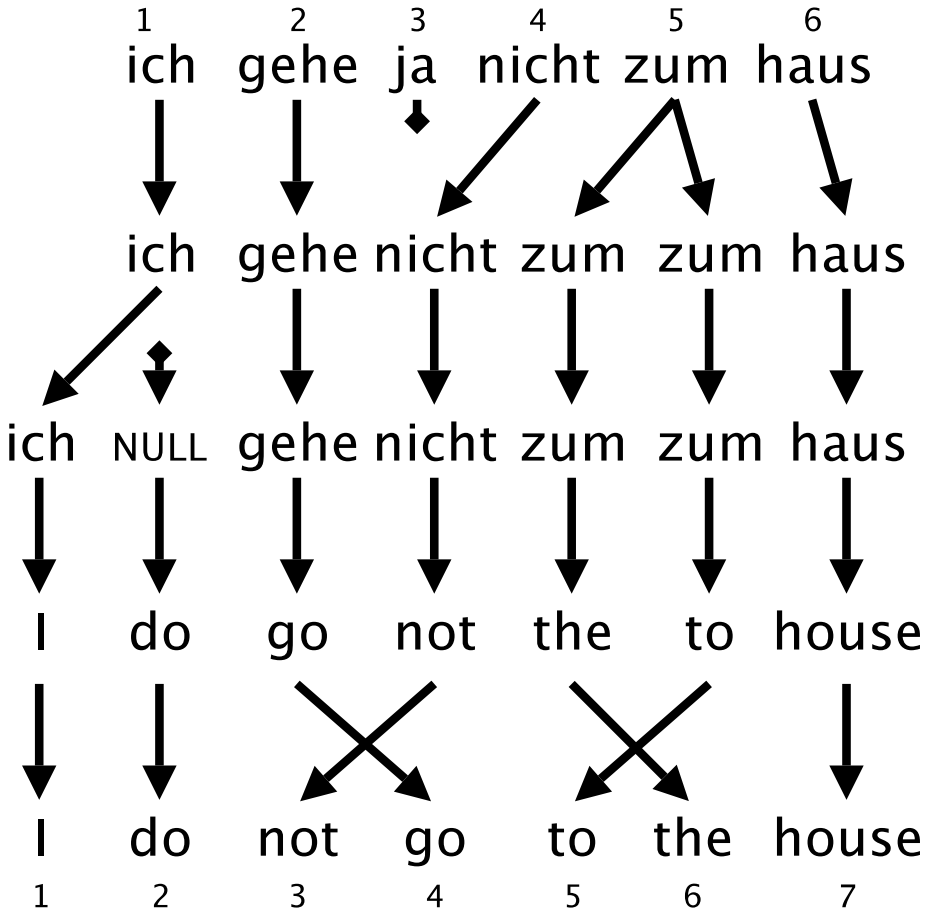
The last step is almost the same as the 2. step in IBM-2 and is modelled with **distortion probability distribution**:

$$d(j|i, l_e, l_f)$$

which models positions in an opposite direction: for each source word at position i it models position j of a target word.

The process of translation from the previous figure might differ a bit (see the next figure).

IBM-3


1 2 3 4 5 6
ich gehe ja nicht zum haus
ich gehe nicht zum zum haus
ich NULL gehe nicht zum zum haus
I do go not the to house
I do not go to the house
1 2 3 4 5 6 7
fertility step
NULL insertion step
lexical translation step
distortion step

IBM-4, IBM-5

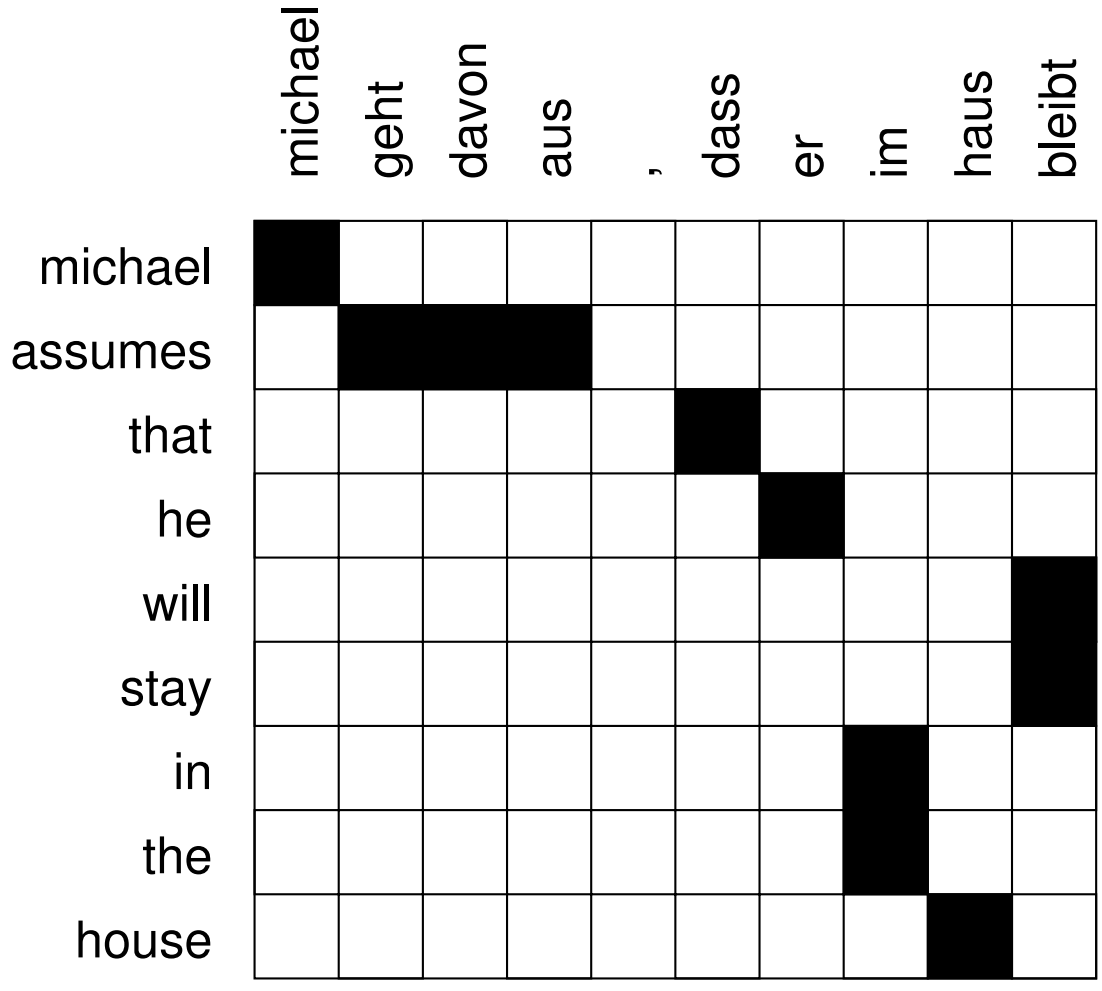
IBM-4

The problem of distortion lies in sparse data for long sentences. IBM-4 introduces **relative distortion**, where rearrangements of word positions depend on particular previous words. It comes from an assumption that we translate in phrases, which are moved together, or some rearrangement are more frequent (English: ADJ SUB $\rightarrow$ French SUB ADJ etc.).

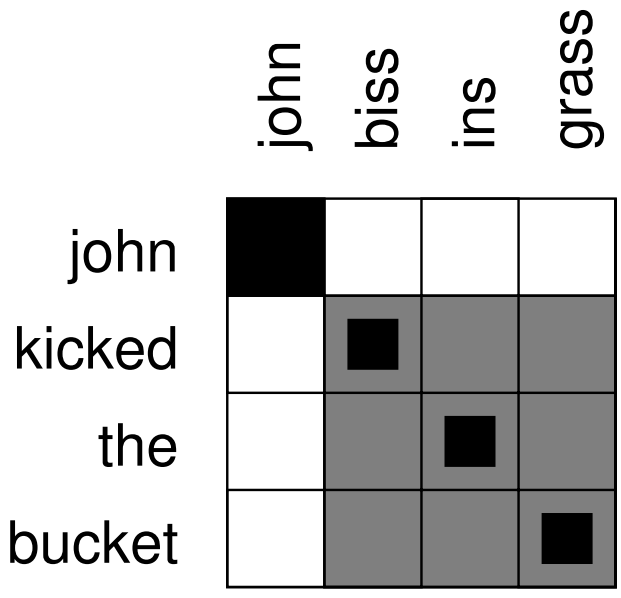
IBM-5

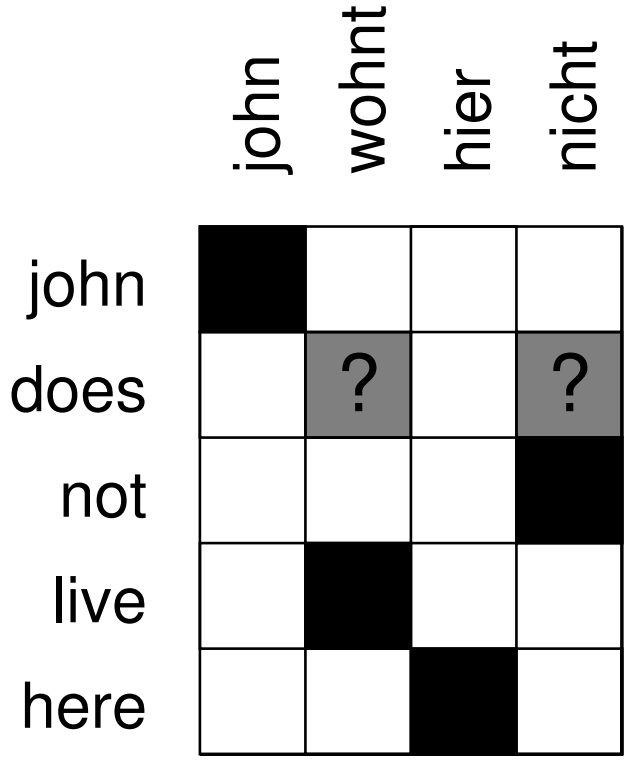
This model solves other shortcomings of the previous models. E.g. it takes care of situations where two different source words could get to one target position.

Word alignment matrix



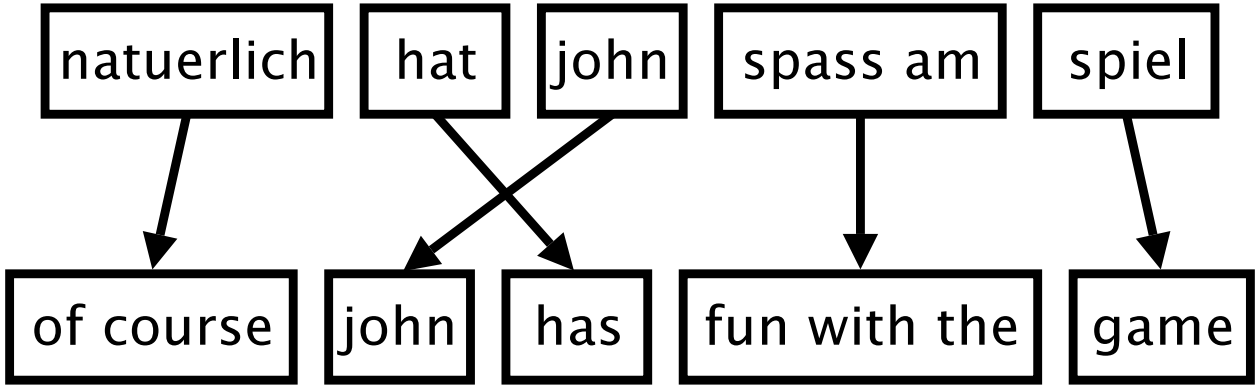
Word alignment issues





Phrase-base Translation Model

State-of-the-art of SMT. Not only single words but whole phrases (n-grams, word sequences) can be translated in one step.



Phrases are not linguistically motivated, only statistically. German *am* is seldom translated with single English *to*. Statistically significant context *spass am* helps better translation. Linguistic phrase would be different: *(fun (with (the game)))*.

Advantages of PBTM

- we often translate $n : m$ words, a word is not a perfect element for translation
- translation of word sequences helps solve some ambiguities
- models can learn to translate longer and longer phrases
- a simplified model: no fertility, no NULL token etc.

Phrase-based model – formula

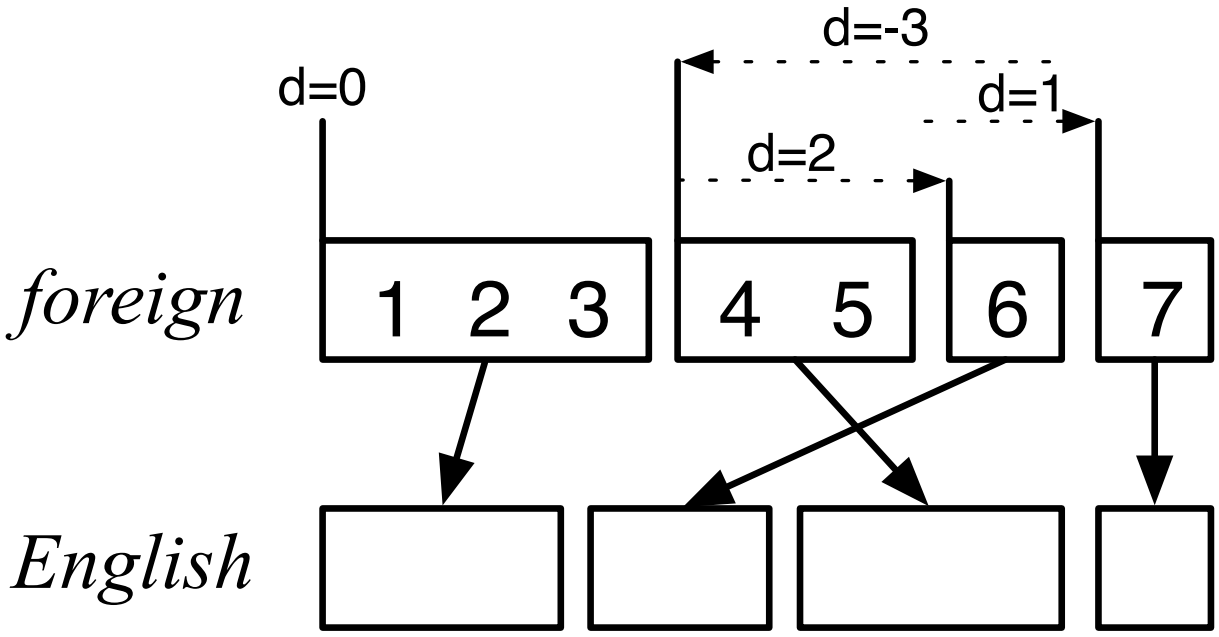
Translation probability $p(f|e)$ is split to phrases

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(\text{start}_i - \text{end}_{i-1} - 1)$$

Sentence f is split to I phrases $\bar{f}_i$, all segmentations are of the same probability. Function ϕ is translation probability for phrases. Function d is distance-based reordering model. We model according a previous phrase. start_i is position of the first word of phrase from sentence f, which is translated to i-th phrase of sentence e.

Distance-based reordering model

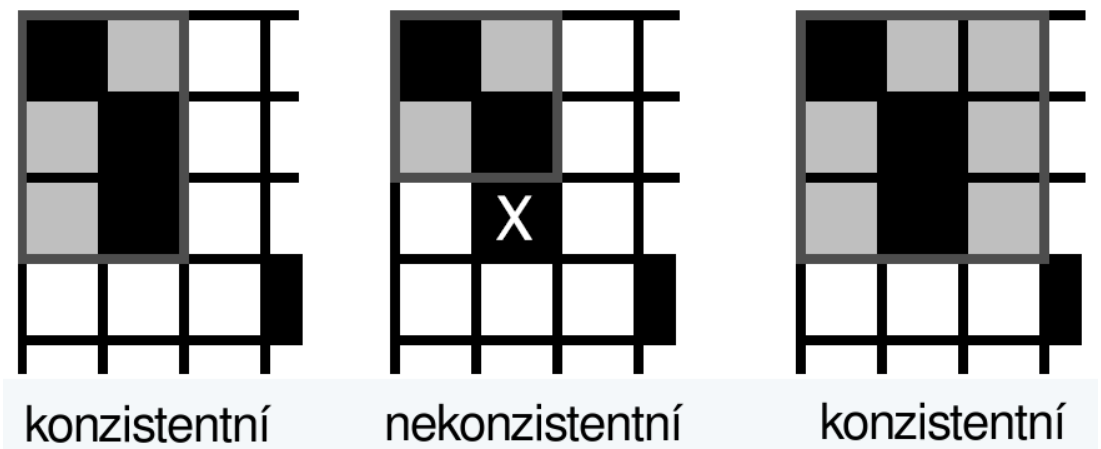
Minimal reordering is preferred. The bigger reordering (measured on source side) the less preferred this operation is.



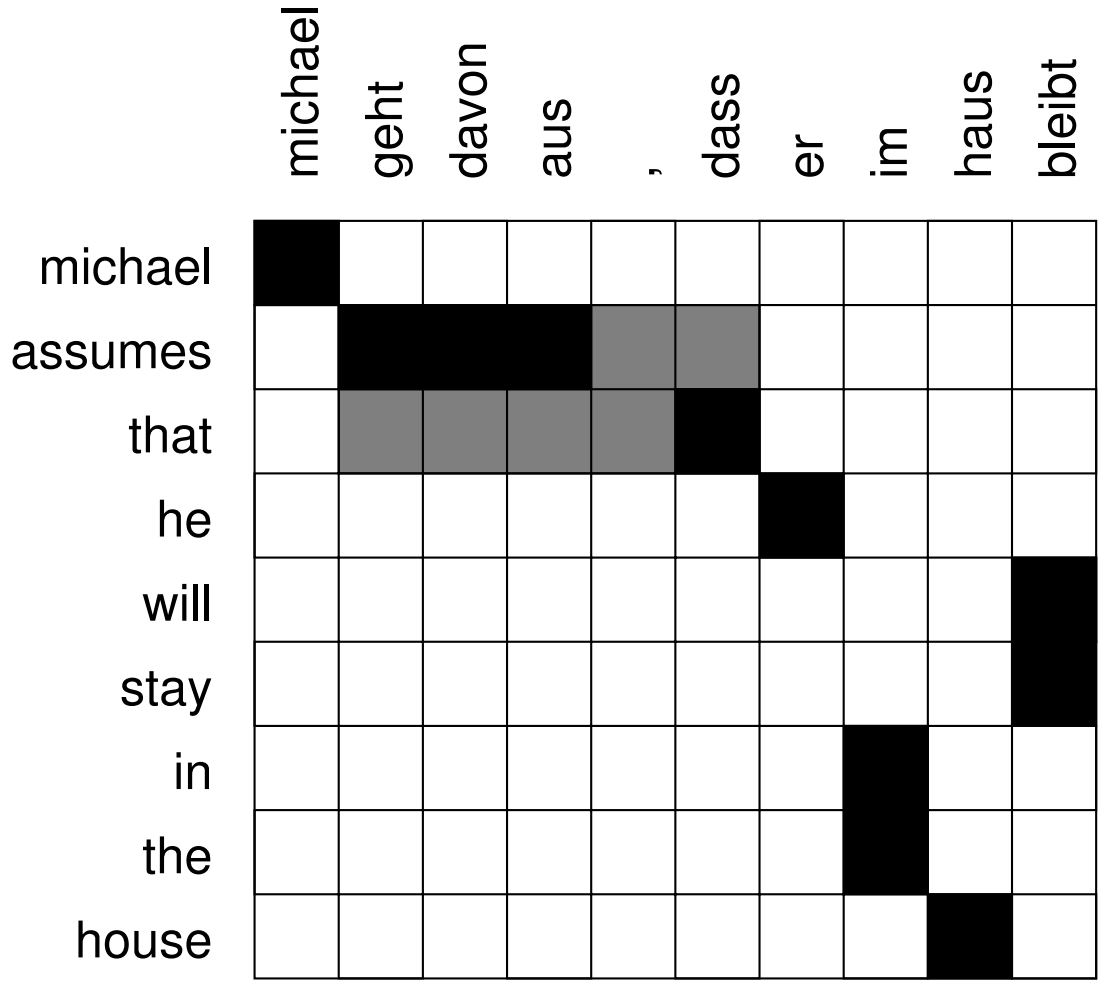
Building phrase matrix

We use word alignments from EM algorithm and then we look for consistent phrases.

Phrase $\bar{f}$ and $\bar{e}$ are consistent with alignment A if all words $f_1, \ldots f_n$ in phrase $\bar{f}$, which have alignment in A, are aligned with words $e_1, \ldots e_n$ in phrase $\bar{e}$ and vice versa.



Phrase extraction



Extracted phrases

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

Phrase translation probability estimation

Phrase translation probability estimation

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

Phrase-based model of SMT

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

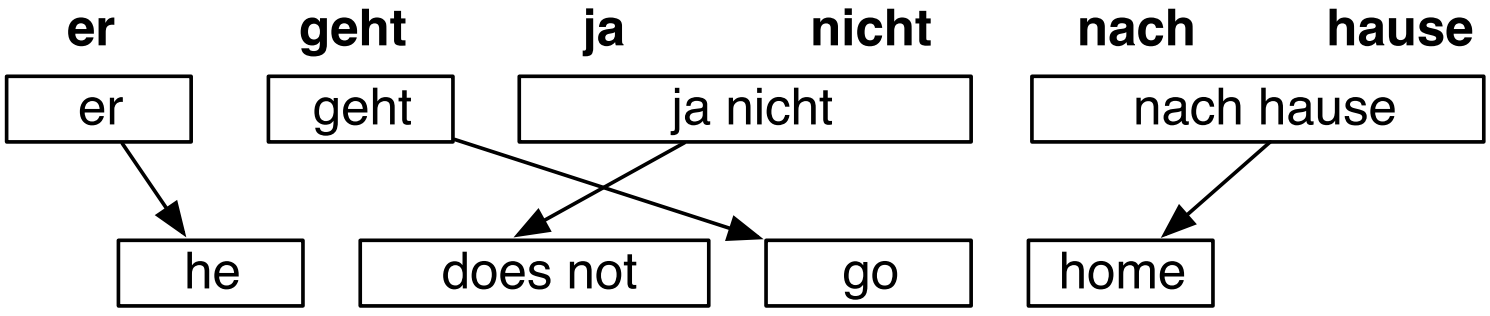
Decoding

Given a model p_{LM} and translation model $p(f|e)$ we need to find a translation with the highest probability but from exponential number of all possible translations.

Heuristic search methods are used. It is not guaranteed we will find the best translation.

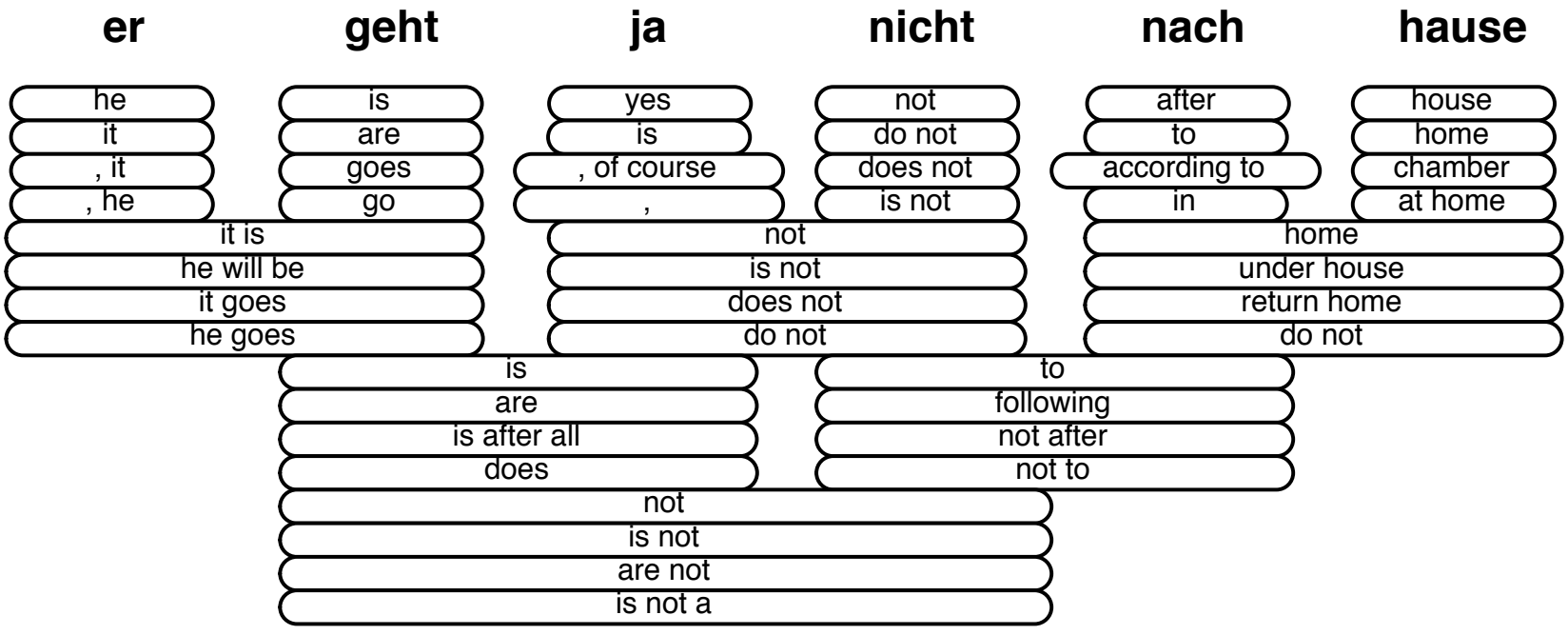
Errors in translations are caused by an error in 1) decoding process, where the best translation is not found owing to the heuristics or 2) models, where the best translation according to the probability functions is not the best.

Phrase-wise sentence translation



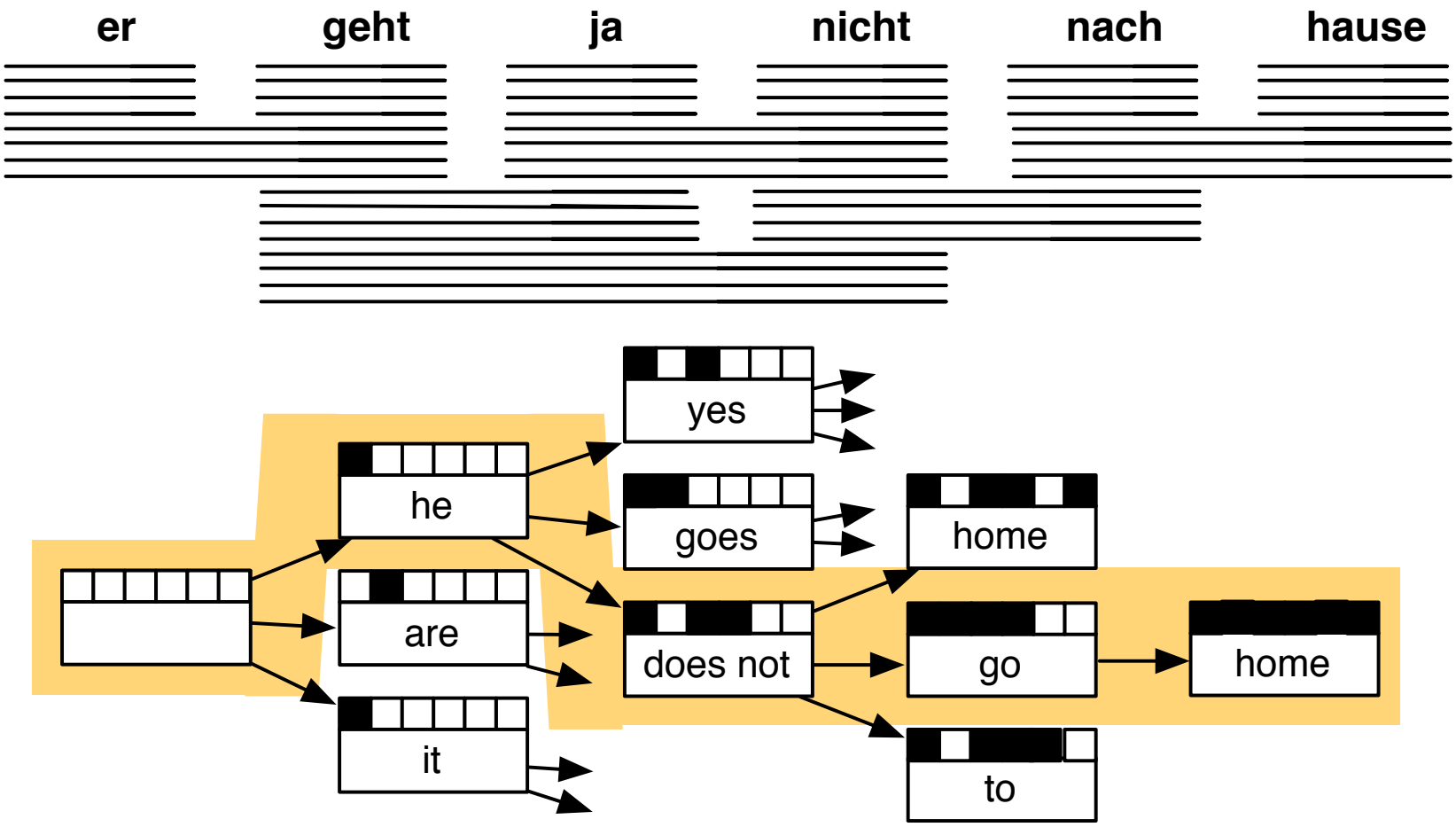
In each step of translation we count preliminary values of probabilities from the translation, reordering and language models.

Search space of translation hypotheses



We are dealing with exponential space of all possible translations. We need to limit this space using various methods.

Hypothesis construction, beam search



Beam search

Beam search uses so called *breadth-first* search. On each level of the search tree it generates all children of nodes on that level, sorts them according to various heuristics. It stores only a limited number of the best states on each level (beam width). Only these states are investigated further. The wider beam width the smaller number of children (descendants) are thrown away (branches are pruned). With an unlimited width it becomes breadth-first search algorithm. The width determines memory consumption. The best final state might not be found since it can be pruned somewhere in the middle of the process.