Statistical Machine Translation

Te Rutherford; Based on Philipp Koehn's slides

CS114 Spring 2015

Ingredients for Statistical MT System

- Parallel corpora or bitext

- Translation model (lexical or phrasal)

- Language model (usually an n-gram model)

- Decoder

- Evaluation metrics for translation quality

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 (German text along with English translation)

Translation of $Haus$	Count
house	8,000
building	1,600
home	200
household	150
shell	50

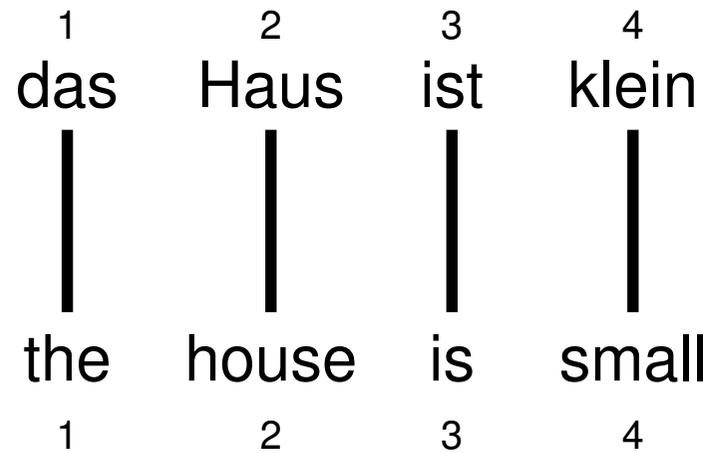# Estimate Translation Probabilities

Maximum likelihood estimation

$$p(e|f = Haus) = \begin{cases} 0.8 & \text{if } e = \text{house}, \\ 0.16 & \text{if } e = \text{building}, \\ 0.02 & \text{if } e = \text{home}, \\ 0.015 & \text{if } e = \text{household}, \\ 0.005 & \text{if } e = \text{shell}. \end{cases}$$

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

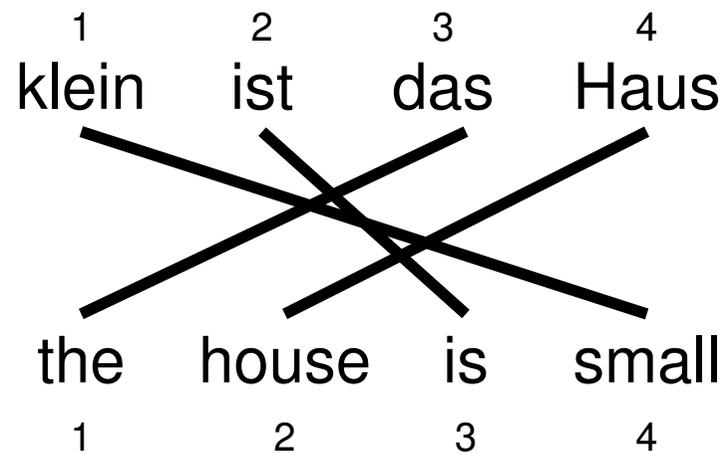- Word positions are numbered 1–4

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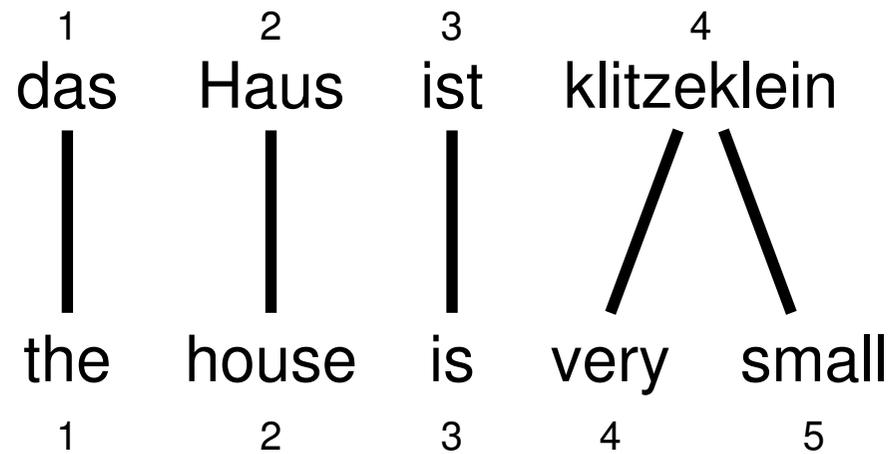
Reordering

Words may be reordered during translation

$$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$
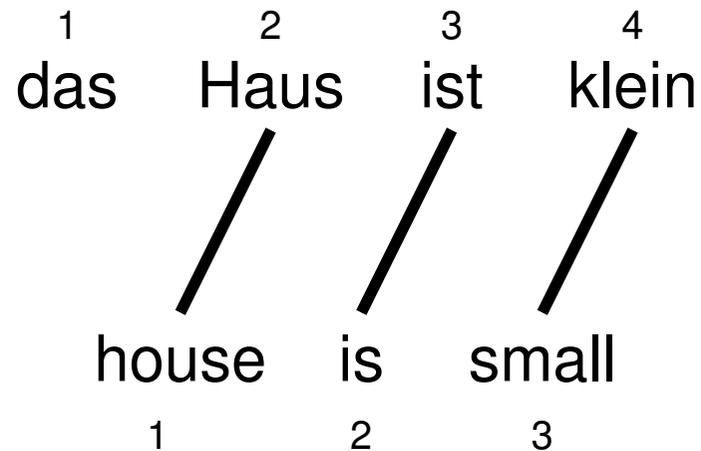
One-to-Many Translation

A source word may translate into multiple target words

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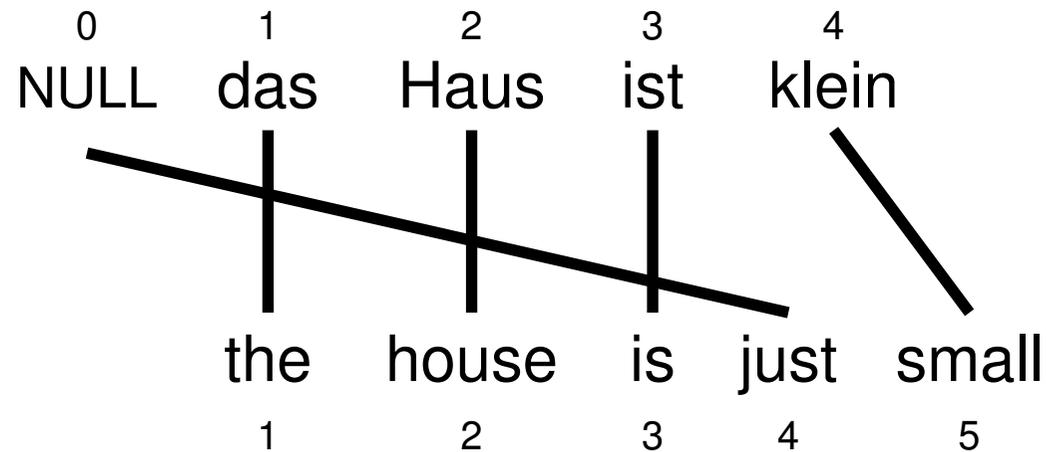
Dropping Words

Words may be dropped when translated
(German article das is dropped)

<pre>
  1   2   3   4
 das   Haus   ist   klein
           /   /   /
          /   /   /
        house   is   small
          1   2   3
</pre>

$$a : \{1 \to 2, 2 \to 3, 3 \to 4\}$$

Inserting Words

- Words may be added during translation

  – The English just does not have an equivalent in German
  – We still need to map it to something: special NULL token

$$
\begin{array}{ccccc}
0 & 1 & 2 & 3 & 4 \\
\text{NULL} & \text{das} & \text{Haus} & \text{ist} & \text{klein}
\end{array}
$$

the   house   is   just   small

1   2   3   4   5

$$a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 0, 5 \to 4\}$$

IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

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

  - parameter ϵ is a normalization constant

Example

das		Haus		ist		klein	
e	$t(e\|f)$	e	$t(e\|f)$	e	$t(e\|f)$	e	$t(e\|f)$
the	0.7	house	0.8	is	0.8	small	0.4
that	0.15	building	0.16	's	0.16	little	0.4
which	0.075	home	0.02	exists	0.02	short	0.1
who	0.05	household	0.015	has	0.015	minor	0.06
this	0.025	shell	0.005	are	0.005	petty	0.04

$$p(e, a|f) = \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein})$$

$$= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4$$

$$= 0.0028\epsilon$$

Learning Lexical Translation Models

- We would like 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*,
    $\rightarrow$ we could estimate the *parameters* of our generative model

  - if we had the *parameters*,
    $\rightarrow$ we could estimate the *alignments*

EM Algorithm

- Incomplete data

  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

- Expectation Maximization (EM) in a nutshell

  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

EM Algorithm
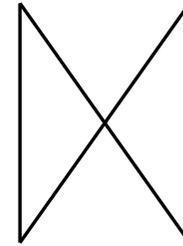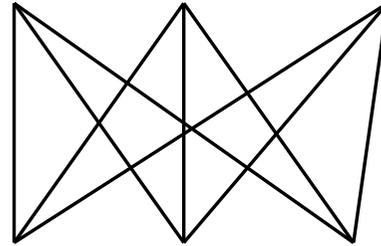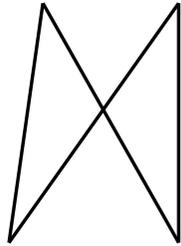
```
... la maison ... la maison blue ... la fleur ...



... the house ... the blue house ... the flower ...
```
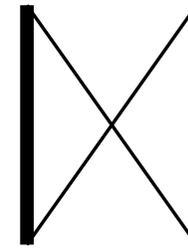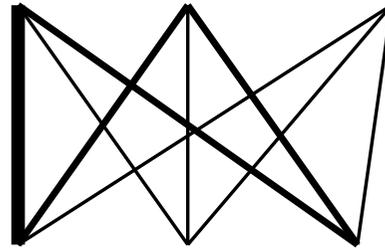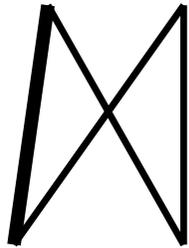
- Initial step: all alignments equally likely

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

EM Algorithm
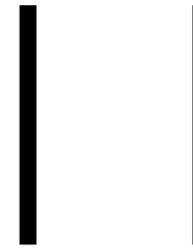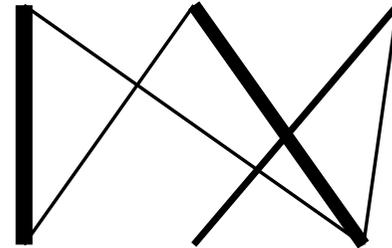

... la maison ... la maison blue ... la fleur ...
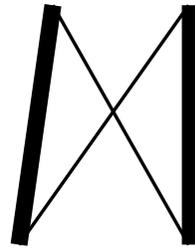
... the house ... the blue house ... the flower ...

- After one iteration

- Alignments, e.g., between la and the are more likely

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

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

- After another iteration

- It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)

EM Algorithm

... la maison ... la maison bleu ... la fleur ...

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

- Convergence

- Inherent hidden structure revealed by EM

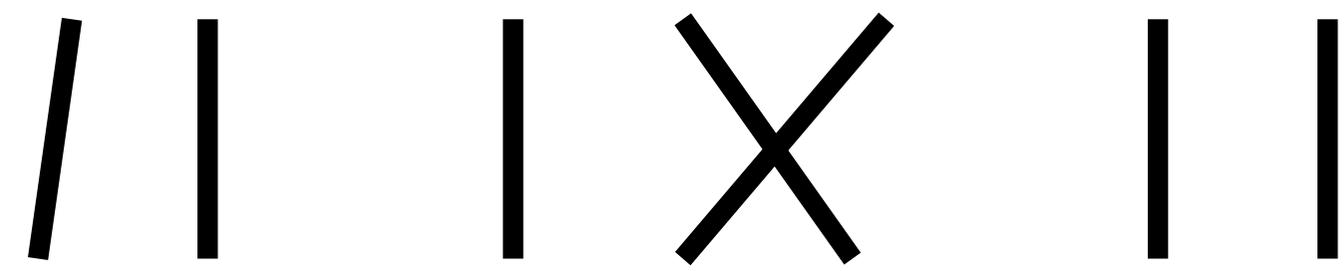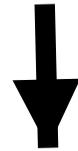# EM Algorithm

... la maison ... la maison bleu ... la fleur ...

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

$$p(\text{la}|\text{the}) = 0.453$$
$$p(\text{le}|\text{the}) = 0.334$$
$$p(\text{maison}|\text{house}) = 0.876$$
$$p(\text{bleu}|\text{blue}) = 0.563$$
$$...$$

• Parameter estimation from the aligned corpus

IBM Model 1 and EM

- EM Algorithm consists of two steps

- Expectation-Step: Apply model to the data to infer the hidden variables.

  - alignments are missing/hidden.
  - using the translation model $t(e|f)$, assign probabilities to each possible alignment.

- Maximization-Step: Estimate model from data

  - collect expected counts i.e. weighted by probabilities
  - estimate model from counts

- Iterate these steps until convergence
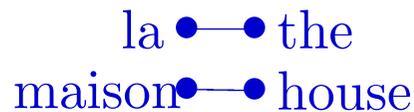
IBM Model 1 and EM

• We need to be able to compute:

– Expectation-Step: probability of alignments

– Maximization-Step: count collection

IBM Model 1 and EM

- **Probabilities**

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

- **Alignments**
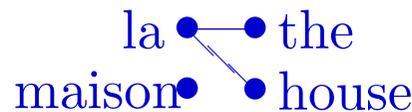
$$p(\mathbf{e}, a|\mathbf{f}) = 0.56 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.035 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.08 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.005$$
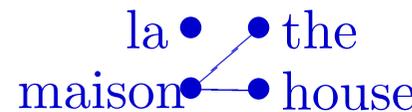
$$p(a|\mathbf{e}, \mathbf{f}) = 0.824 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.052 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.118 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.007$$
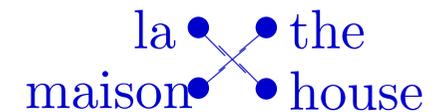
- **Counts**

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

IBM Model 1 and EM: Expectation Step

- We need to compute $p(a|\mathbf{e}, \mathbf{f})$

- Applying the chain rule:

$$p(a|\mathbf{e}, \mathbf{f}) = \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})}$$

- We already have the formula for $p(\mathbf{e}, \mathbf{a}|\mathbf{f})$ (definition of Model 1)

IBM Model 1 and EM: Expectation Step

- We need to compute $p(\mathbf{e}|\mathbf{f})$

$$p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, a|\mathbf{f})$$

$$= \sum_{a(1)=0}^{l_f} ... \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f})$$

$$= \sum_{a(1)=0}^{l_f} ... \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

IBM Model 1 and EM: Expectation Step

$$p(\mathbf{e}|\mathbf{f}) = \sum_{a(1)=0}^{l_f} ... \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

$$= \frac{\epsilon}{(l_f + 1)^{l_e}} \sum_{a(1)=0}^{l_f} ... \sum_{a(l_e)=0}^{l_f} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

$$= \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

- Note the trick in the last line

  − removes the need for an exponential number of products
  $\rightarrow$ this makes IBM Model 1 estimation tractable

The Trick

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} = \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j | f_{a(j)}) =$$

$$= t(e_1|f_0)\, t(e_2|f_0) + t(e_1|f_0)\, t(e_2|f_1) + t(e_1|f_0)\, t(e_2|f_2) +$$

$$+ t(e_1|f_1)\, t(e_2|f_0) + t(e_1|f_1)\, t(e_2|f_1) + t(e_1|f_1)\, t(e_2|f_2) +$$

$$+ t(e_1|f_2)\, t(e_2|f_0) + t(e_1|f_2)\, t(e_2|f_1) + t(e_1|f_2)\, t(e_2|f_2) =$$

$$= t(e_1|f_0)\, (t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) +$$

$$+ t(e_1|f_1)\, (t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) +$$

$$+ t(e_1|f_2)\, (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) =$$

$$= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\, (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))$$

IBM Model 1 and EM: Expectation Step

- Combine what we have:

$$p(\mathbf{a}|\mathbf{e}, \mathbf{f}) = p(\mathbf{e}, \mathbf{a}|\mathbf{f})/p(\mathbf{e}|\mathbf{f})$$

$$= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)}$$

$$= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}$$

IBM Model 1 and EM: Maximization Step

- Now we have to collect counts

- Evidence from a sentence pair **e,f** that word e is a translation of word f:

$$c(e|f; \mathbf{e}, \mathbf{f}) = \sum_a p(a|\mathbf{e}, \mathbf{f}) \sum_{j=1}^{l_e} \delta(e, e_j) \delta(f, f_{a(j)})$$

- With the same simplication as before:

$$c(e|f; \mathbf{e}, \mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i)$$

IBM Model 1 and EM

- **Probabilities**

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

- **Alignments**

$$p(\mathbf{e}, a|\mathbf{f}) = 0.56 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.035 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.08 \qquad p(\mathbf{e}, a|\mathbf{f}) = 0.005$$

$$p(a|\mathbf{e}, \mathbf{f}) = 0.824 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.052 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.118 \quad p(a|\mathbf{e}, \mathbf{f}) = 0.007$$

- **Counts**

$$c(\text{the}|\text{la}) = 0.824 + 0.052 = \frac{0.7}{0.7+0.1}$$
$$c(\text{house}|\text{la}) = 0.052 + 0.007 = \frac{0.05}{0.05+0.8}$$
$$c(\text{the}|\text{maison}) = 0.118 + 0.007 = ?$$
$$c(\text{house}|\text{maison}) = 0.824 + 0.118 = ?$$

IBM Model 1 and EM: Maximization Step

- After collecting these counts over a corpus, we can estimate the model:

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

- This is much faster than directly compute $p(a|\mathbf{e}, \mathbf{f})$ for each sentence pair.

IBM Model 1 and EM: Pseudocode

**Input:** set of sentence pairs $(\mathbf{e}, \mathbf{f})$

Output: translation prob. $t(e|f)$

  1: initialize $t(e|f)$ uniformly

  2: **while** not converged **do**

  3:     *// initialize*

  4:     count$(e|f) = 0$ **for all** e, f

  5:     total$(f) = 0$ **for all** f

  6:     **for all** sentence pairs $(\mathbf{e},\mathbf{f})$ **do**

  7:       *// compute normalization*

  8:       **for all** words e in $\mathbf{e}$ **do**

  9:         s-total$(e) = 0$

10:         **for all** words f in $\mathbf{f}$ **do**

11:           s-total(e) += $t(e|f)$

12:         **end for**

13:       **end for**

14:       *// collect counts*

15:       **for all** words e in $\mathbf{e}$ **do**

16:         **for all** words f in $\mathbf{f}$ **do**

17:           count$(e|f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$

18:           total(f) += $\frac{t(e|f)}{\text{s-total}(e)}$

19:         **end for**

20:       **end for**

21:     **end for**

22:     *// estimate probabilities*

23:     **for all** foreign words f **do**

24:       **for all** English words e **do**

25:         $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$

26:       **end for**

27:     **end for**

28: **end while**

Convergence

das   Haus   das   Buch   ein   Buch
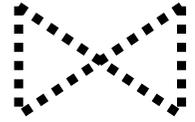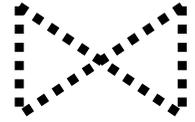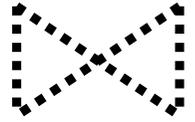
the   house   the   book   a   book

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

$$\log_2 PP = -\sum_s \log_2 p(\mathbf{e}_s | \mathbf{f}_s)$$

- Example ($\epsilon{=}1$)

	initial	1st it.	2nd it.	3rd it.	...	final
$p(\text{the haus}|\text{das haus})$	0.0625	0.1875	0.1905	0.1913	...	0.1875
$p(\text{the book}|\text{das buch})$	0.0625	0.1406	0.1790	0.2075	...	0.25
$p(\text{a book}|\text{ein buch})$	0.0625	0.1875	0.1907	0.1913	...	0.1875
perplexity	4095	202.3	153.6	131.6	...	113.8

Ensuring Fluent Output

- Our 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

- Language model

  - estimate how likely a string is English
  - based on n-gram statistics
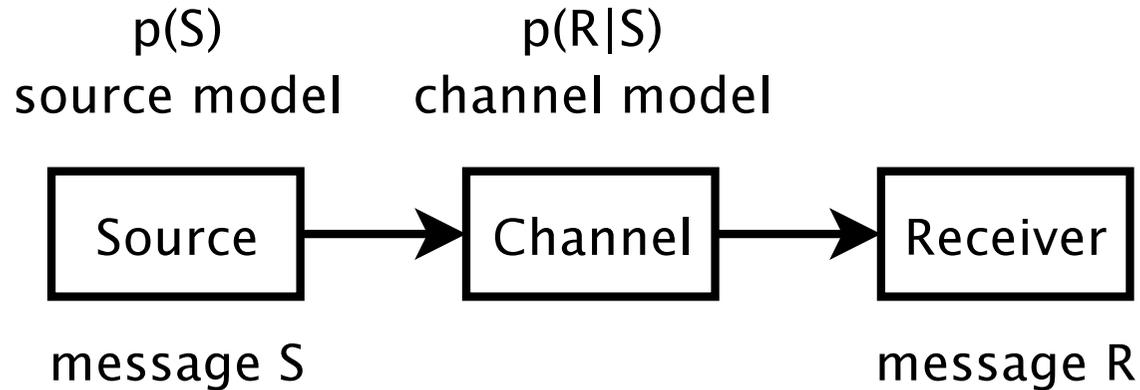
$$p(\mathbf{e}) = p(e_1, e_2, ..., e_n)$$
$$= p(e_1)p(e_2|e_1)...p(e_n|e_1, e_2, ..., e_{n-1})$$
$$\simeq p(e_1)p(e_2|e_1)...p(e_n|e_{n-2}, e_{n-1})$$

Noisy Channel Model

- We would like to integrate a language model

- Bayes rule

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

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

Noisy Channel Model

p(S)
source model

p(R|S)
channel model

Source → Channel → Receiver

message S

message R

- Applying Bayes rule also called noisy channel model

  – we observe a distorted message R (here: a foreign string **f**)
  – we have a model on how the message is distorted (here: translation model)
  – we have a model on what messages are probably (here: language model)
  – we want to recover the original message S (here: an English string **e**)

Higher IBM Models

IBM Model 1	lexical translation
IBM Model 2	adds absolute reordering model
IBM Model 3	adds fertility model
IBM Model 4	relative reordering model
IBM Model 5	fixes deficiency

- Only IBM Model 1 has global maximum
  - training of a higher IBM model builds on previous model

- Compuationally biggest change in Model 3
  - trick to simplify estimation does not work anymore
  $\rightarrow$ exhaustive count collection becomes computationally too expensive
  - sampling over high probability alignments is used instead

Reminder: IBM Model 1

- Generative model: break up translation process into smaller steps
  - IBM Model 1 only uses lexical translation

- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

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

  - parameter ϵ is a normalization constant

IBM Model 2

Adding a model of alignment
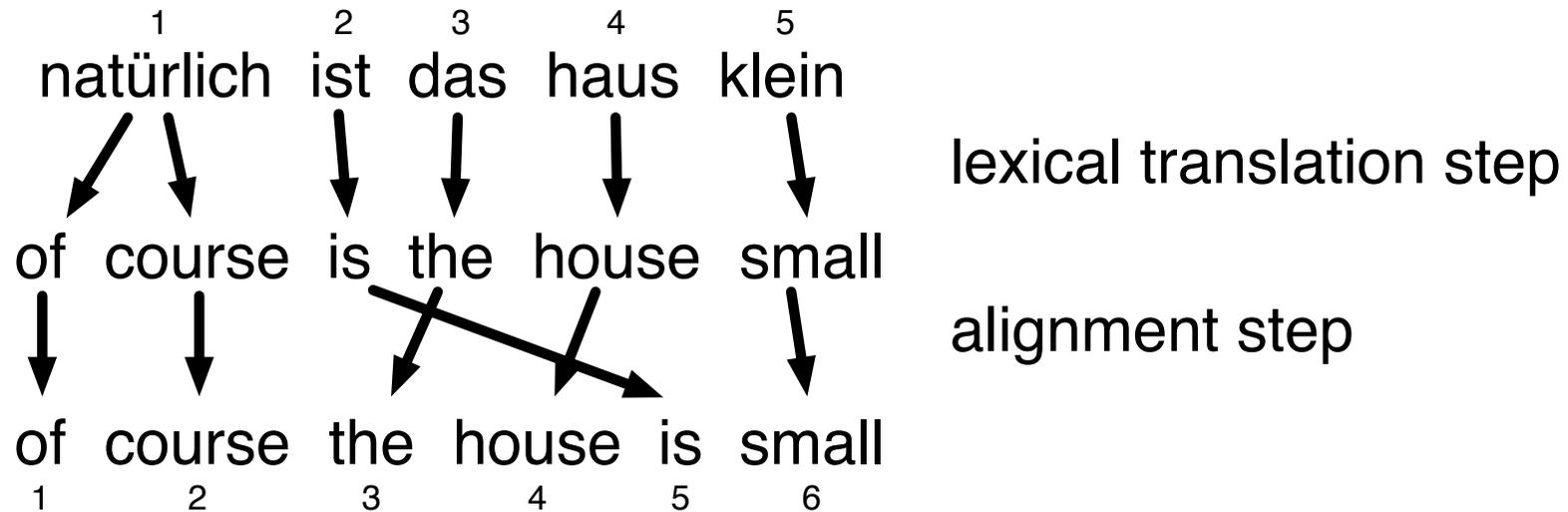
IBM Model 2

- Modeling alignment with an alignment probability distribution

- Translating foreign word at position i to English word at position j:

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

- Putting everything together

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

- EM training of this model works the same way as IBM Model 1

Interlude: HMM Model

- Words do not move independently of each other

  - they often move in groups
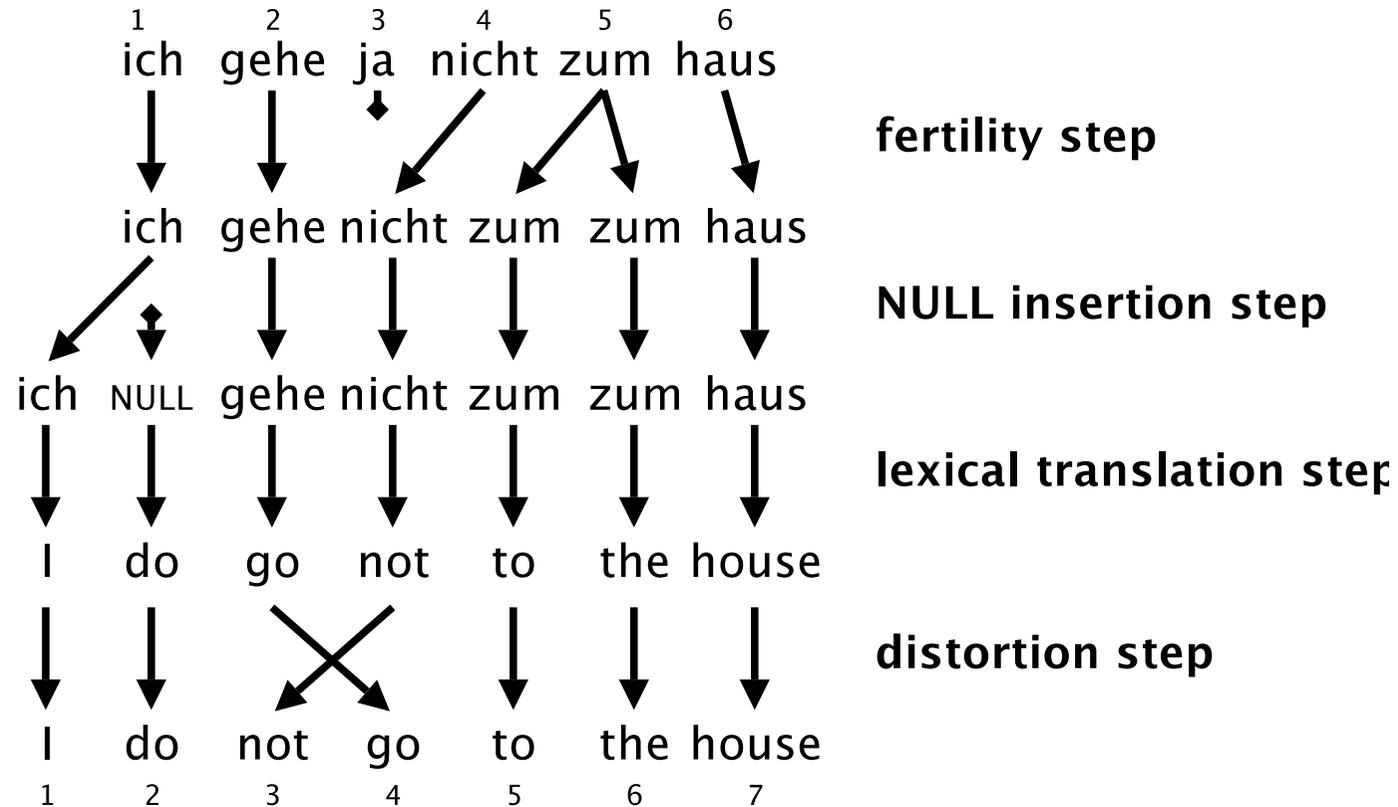 $\rightarrow$ condition word movements on previous word

- HMM alignment model:

$$p(a(j)|a(j-1), l_e)$$

- EM algorithm application harder, requires dynamic programming

- IBM Model 4 is similar, also conditions on word classes

IBM Model 3

Adding a model of fertilty

IBM Model 3: Fertility

- Fertility: number of English words generated by a foreign word

- Modelled by distribution $n(\phi|f)$

- Example:

$$n(1|\text{haus}) \simeq 1$$
$$n(2|\text{zum}) \simeq 1$$
$$n(0|\text{ja}) \simeq 1$$

Sampling the Alignment Space

- Training IBM Model 3 with the EM algorithm

  - The trick that reduces exponential complexity does not work anymore
  $\rightarrow$ Not possible to exhaustively consider all alignments

- Finding the most probable alignment by hillclimbing

  - start with initial alignment
  - change alignments for individual words
  - keep change if it has higher probability
  - continue until convergence

- Sampling: collecting variations to collect statistics

  - all alignments found during hillclimbing
  - neighboring alignments that differ by a move or a swap

IBM Model 4

- Better reordering model

- Reordering in IBM Model 2 and 3

  - recall: $d(j||_i, l_e, lf)$
  - for large sentences (large l_f and l_e), sparse and unreliable statistics
  - phrases tend to move together

- Relative reordering model: relative to previously translated words (cepts)

IBM Model 5

- IBM Models 1–4 are *deficient*

  - some impossible translations have positive probability
  - multiple output words may be placed in the same position
  $\rightarrow$ probability mass is wasted

- IBM Model 5 fixes deficiency by keeping track of vacancies (available positions)

Conclusion

- 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

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

Word Alignment?

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

Measuring Word Alignment Quality

- Manually align corpus with *sure* (S) and *possible* (P) alignment points ($S \subseteq P$)

- Common metric for evaluation word alignments: Alignment Error Rate (AER)

$$\mathsf{AER}(S, P; A) = \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

- 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)

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

Symmetrizing Word Alignments

English to German

German to English

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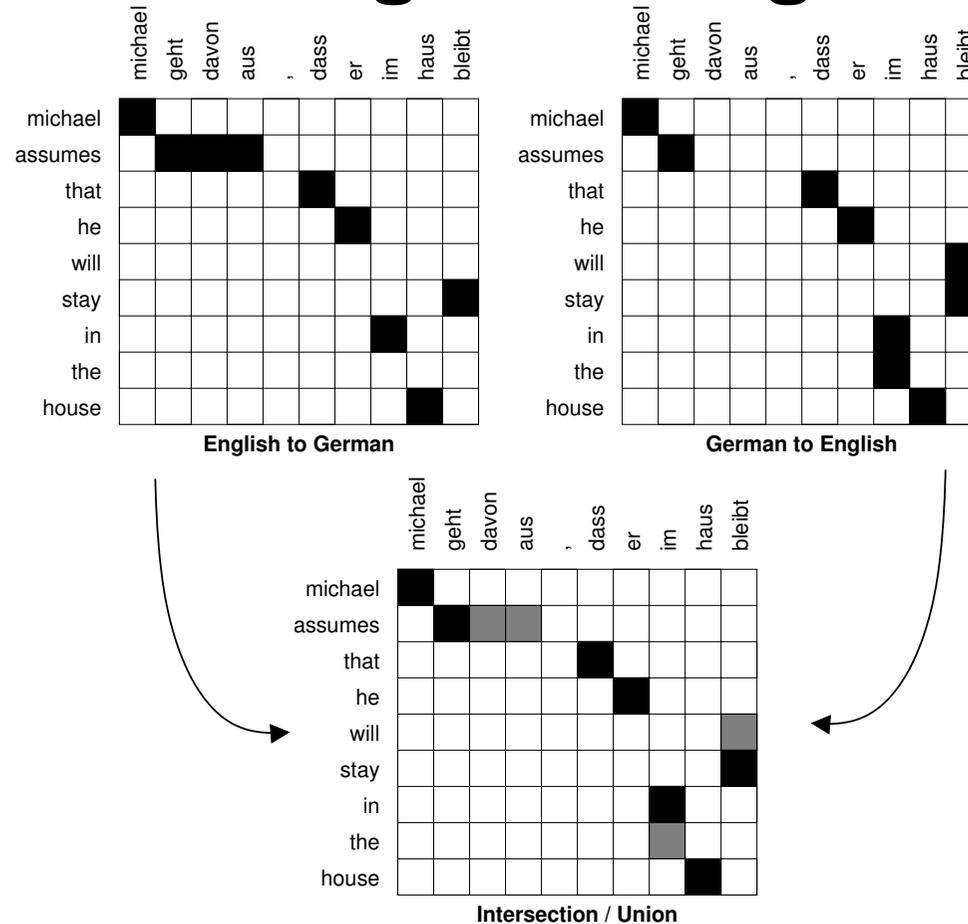
- Intersection of GIZA++ bidirectional alignments
- Grow additional alignment points [Och and Ney, CompLing2003]

Summary

- Lexical translation

- Alignment

- Expectation Maximization (EM) Algorithm

- Noisy Channel Model

- IBM Models 1–5

    - IBM Model 1: lexical translation
    - IBM Model 2: alignment model
    - IBM Model 3: fertility
    - IBM Model 4: relative alignment model
    - IBM Model 5: deficiency

- Word Alignment

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

Phrase-Based Model

natuerlich	hat	john	spass am	spiel

of course	john	has	fun with the	game
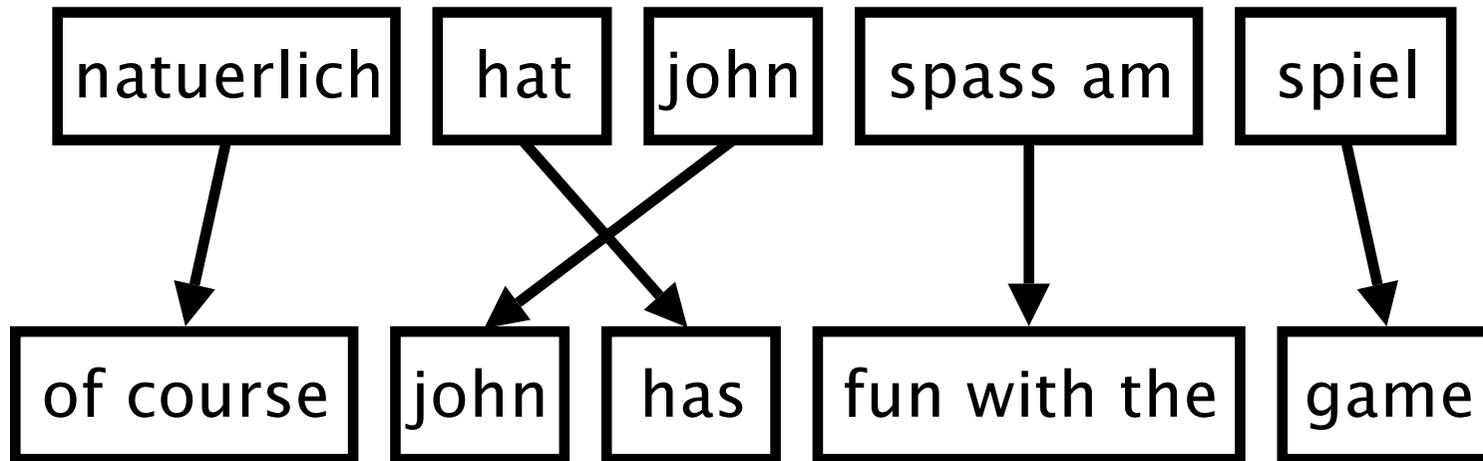
- Foreign input is segmented in phrases

- Each phrase is translated into English

- Phrases are reordered

Phrase Translation Table

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

- Example: phrase translations for natuerlich

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

Real Example

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

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

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

Linguistic Phrases?

- Model is not limited to linguistic phrases
  (noun phrases, verb phrases, prepositional phrases, ...)

- Example non-linguistic phrase pair

$$\text{spass am} \rightarrow \text{fun with the}$$

- Prior noun often helps with translation of preposition

- Experiments show that limitation to linguistic phrases hurts quality

Probabilistic Model

- Bayes rule

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

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

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

- Decomposition of the translation model

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

  – phrase translation probability ϕ
  – reordering probability d

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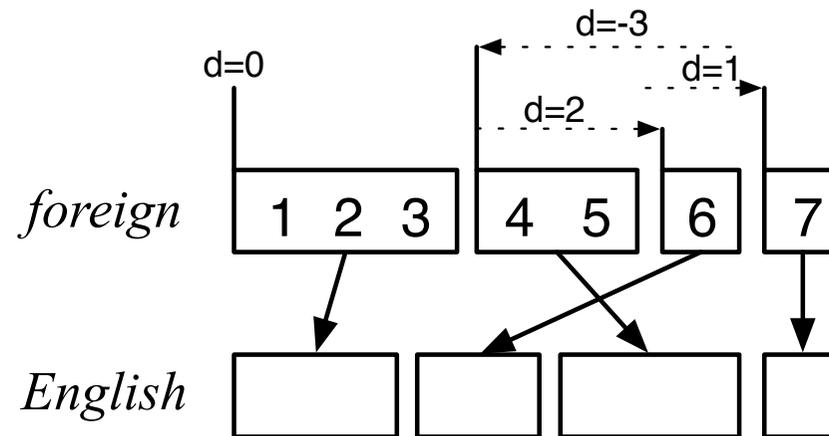
phrase	translates	movement	distance
1	1–3	start at beginning	0
2	6	skip over 4–5	+2
3	4–5	move back over 4–6	-3
4	7	skip over 6	+1

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

Learning a Phrase Translation Table

- Task: learn the model from a parallel corpus

- Three stages:

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

Word Alignment

Extracting Phrase Pairs

extract phrase pair consistent with word alignment:

assumes that / geht davon aus , dass

Consistent
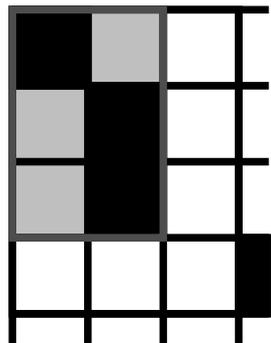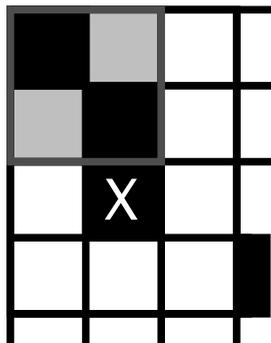
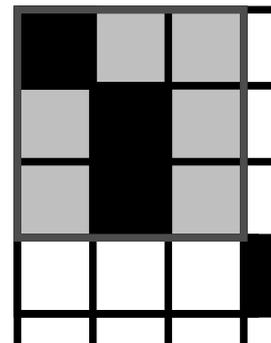| consistent	inconsistent	consistent
ok	**violated**	**ok**
	one alignment	unaligned
	point outside	word is fine

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

Phrase Pair Extraction

Smallest phrase pairs:

michael — michael

assumes — geht davon aus / geht davon aus ,

that — dass / , dass

he — er

will stay — bleibt

in the — im

house — haus

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

Larger Phrase Pairs

michael assumes — michael geht davon aus / michael geht davon aus ,

assumes that — geht davon aus , dass   ;   assumes that he — geht davon aus , dass er

that he — dass er / , dass er   ;   in the house — im haus

michael assumes that — michael geht davon aus , dass

michael assumes that he — michael geht davon aus , dass er

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

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

that he will stay in the house — dass er im haus bleibt   ;   dass er im haus bleibt ,

he will stay in the house — er im haus bleibt   ;   will stay in the house — im haus bleibt

Scoring Phrase Translations

- Phrase pair extraction: collect all phrase pairs from the data

- Phrase pair scoring: assign probabilities to phrase translations

- Score by relative frequency:

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

Size of the Phrase Table

- Phrase translation table typically bigger than corpus

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

$\rightarrow$ 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

Summary

- Phrase Model

- Training the model

  - word alignment
  - phrase pair extraction
  - phrase pair scoring