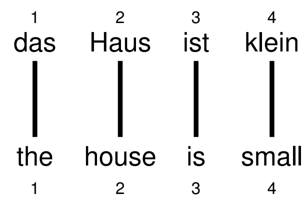
Alignment

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



- Word positions are numbered 1–4

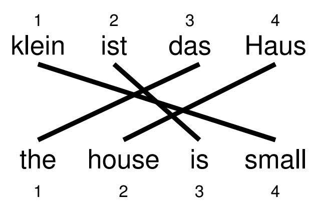
Alignment Function

- 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 $a : i \to j$
- Example

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

Reordering

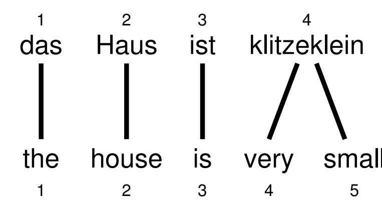
Words may be reordered during translation



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

One-to-Many Translation

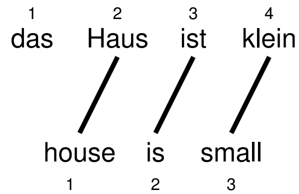
A source word may translate into multiple target words



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

Dropping Words

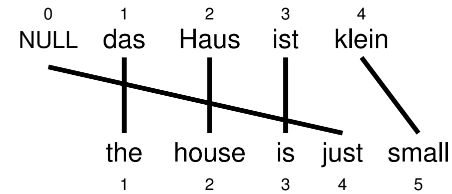
Words may be dropped when translated
(German article das is dropped)



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

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



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

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

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

$$\begin{aligned} p(e, a|f) &= \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\ &= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\ &= 0.0028\epsilon \end{aligned}$$

em algorithm

EM Algorithm

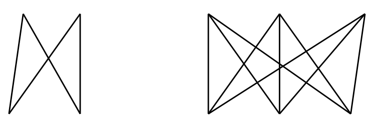
- Incomplete data
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

- 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 ...`



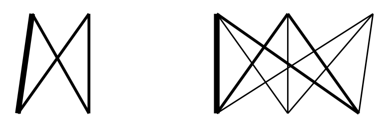
`... the house ... the blue house ... the flower ...`

- 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 ...`



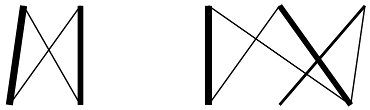
`... the house ... the blue house ... the flower ...`

- After one iteration

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

EM Algorithm

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

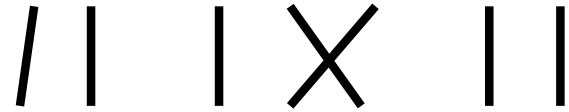


`... the house ... the blue house ... the flower ...`

- 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 ...`

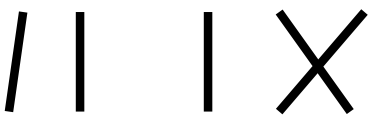


`... the house ... the blue house ... the flower ...`

- Convergence
- Inherent hidden structure revealed by EM

EM Algorithm

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



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

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

- Parameter estimation from the aligned corpus

IBM Model 1 and EM

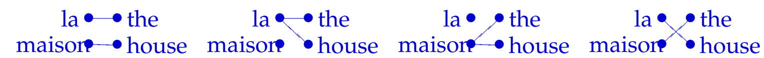
- EM Algorithm consists of two steps
- Expectation-Step: Apply model to the data
  - parts of the model are hidden (here: alignments)
  - using the model, assign probabilities to possible values
- Maximization-Step: Estimate model from data
  - take assign values as fact
  - collect counts (weighted 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** $p(\text{the}|\text{la}) = 0.7$ $p(\text{house}|\text{la}) = 0.05$ $p(\text{the}|\text{maison}) = 0.1$ $p(\text{house}|\text{maison}) = 0.8$
- **Alignments**



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

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

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

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 $p(\mathbf{e}|\mathbf{f})$

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

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

→ this makes IBM Model 1 estimation tractable

The Trick

(case $l_e = l_f = 2$)

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

$$\begin{aligned}
&= t(e_1|f_0)\, t(e_2|f_0) + t(e_1|f_0)\, t(e_2|f_1) + t(e_1|f_0)\, t(e_2|f_2) + \\
&\quad + t(e_1|f_1)\, t(e_2|f_0) + t(e_1|f_1)\, t(e_2|f_1) + t(e_1|f_1)\, t(e_2|f_2) + \\
&\quad + t(e_1|f_2)\, t(e_2|f_0) + t(e_1|f_2)\, t(e_2|f_1) + t(e_1|f_2)\, t(e_2|f_2) = \\
&= t(e_1|f_0)\, (t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_1)\, (t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_2)\, (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) = \\
&= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\, (t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\end{aligned}$$

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: Maximization Step

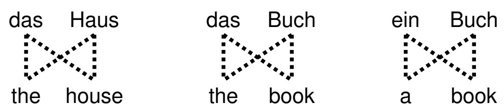
After collecting these counts over a corpus, we can estimate the model:

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

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
```


Convergence



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 (ϵ=1)

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

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

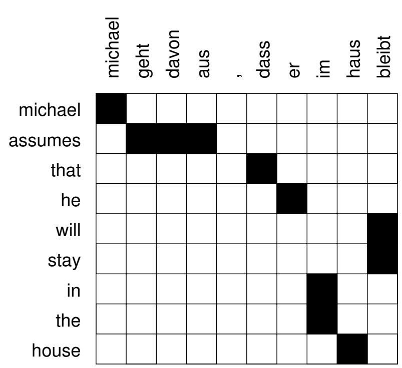
- Only IBM Model 1 has global maximum
  - training of a higher IBM model builds on previous model

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

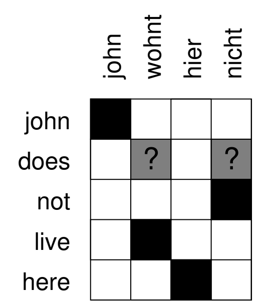
word alignment

Word Alignment

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

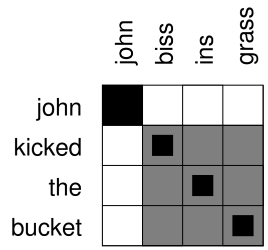


Word Alignment?



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)

$$\text{AER}(S, P; A) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

- AER = 0: alignment A matches all sure, any possible alignment points
- However: different applications require different precision/recall trade-offs

symmetrization

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

Symmetrization

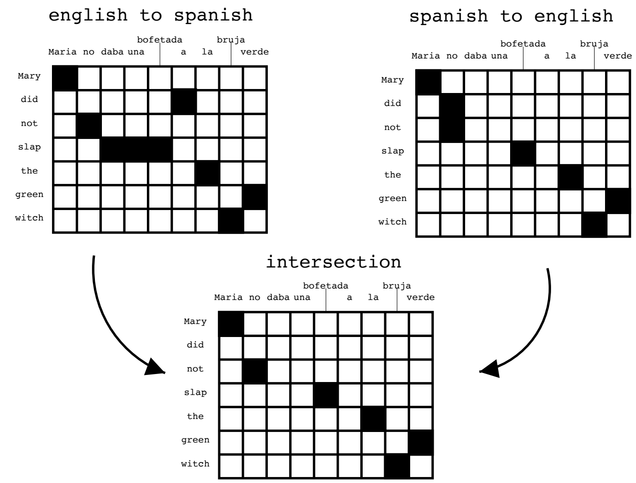
- Run IBM Model training in both directions

→ two sets of word alignment points

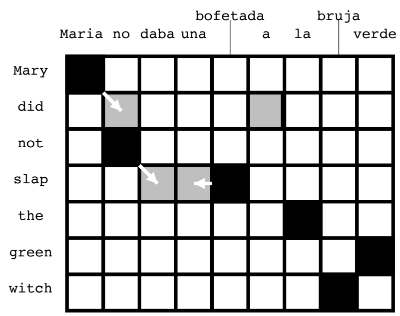
- Intersection: high precision alignment points
- Union: high recall alignment points

- Refinement methods explore the sets between intersection and union

Example



Growing Heuristics



black: intersection **grey**: additional points in union

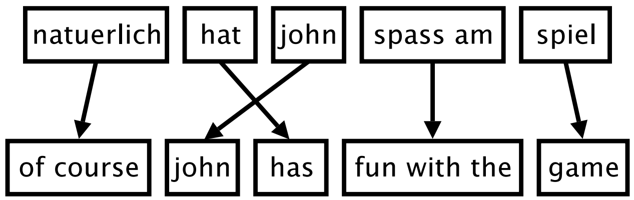
- Add alignment points from union based on heuristics:
  - directly/diagonally neighboring points
  - finally, add alignments that connect unaligned words in source and/or target
- Popular method: grow-diag-final-and

Phrase-Based Models

Philipp Koehn

18 September 2018

Phrase-Based Model



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

Phrase Translation Table

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

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

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:

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

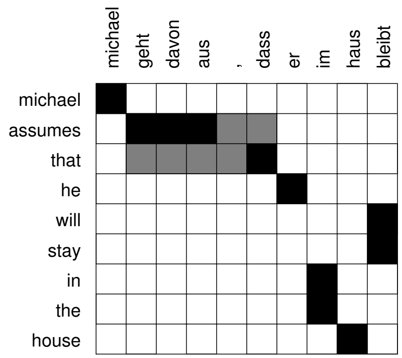
Real Example

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

English	$\phi(\bar{e}\vert f)$	English	$\phi(\bar{e}\vert f)$
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)

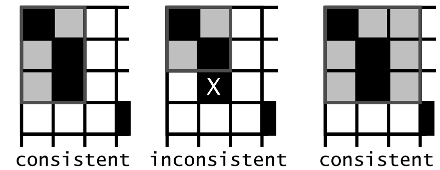
Extracting Phrase Pairs



extract phrase pair consistent with word alignment:

assumes that / geht davon aus , dass

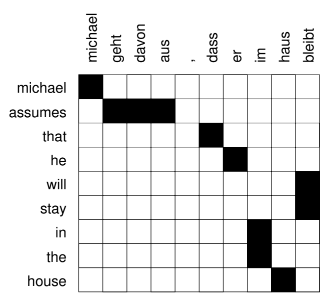
Consistent



ok	**violated** one alignment point outside	**ok** unaligned word is fine

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

Phrase Pair Extraction

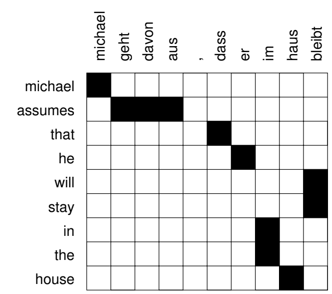


Smallest phrase pairs:

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

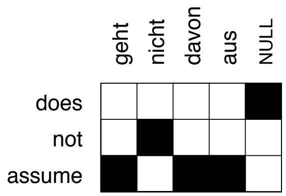
Larger Phrase Pairs



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

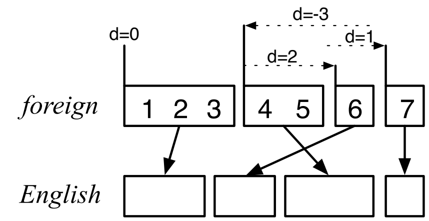
More Feature Functions

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



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

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

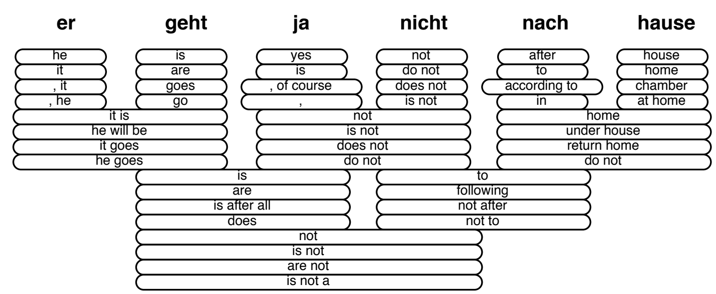
Scoring function: $d(x) = \alpha^{|x|}$ — exponential with distance

Decoding

Philipp Koehn

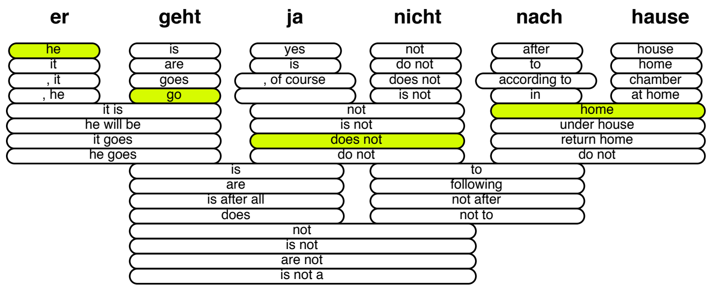
20 September 2018

Translation Options



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

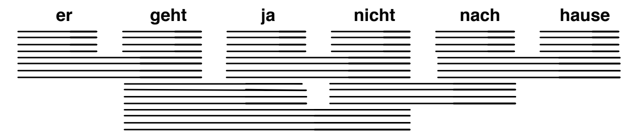
Translation Options



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

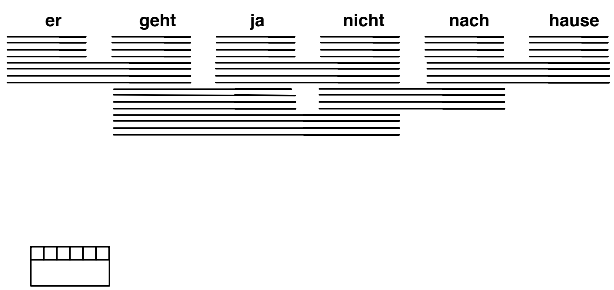
→ Search problem solved by heuristic beam search

Decoding: Precompute Translation Options



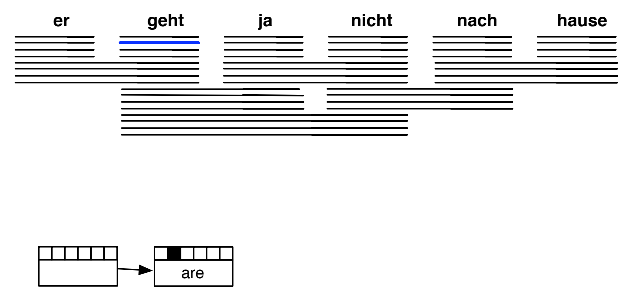
consult phrase translation table for all input phrases

Decoding: Start with Initial Hypothesis



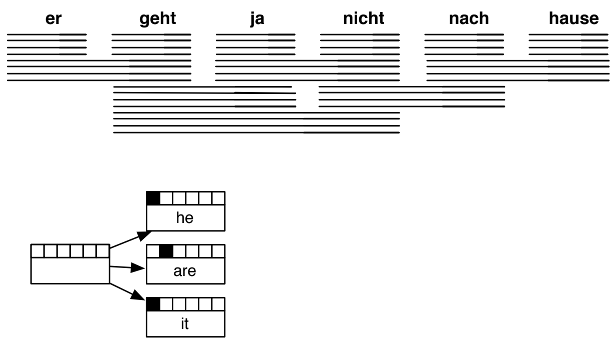
initial hypothesis: no input words covered, no output produced

Decoding: Hypothesis Expansion



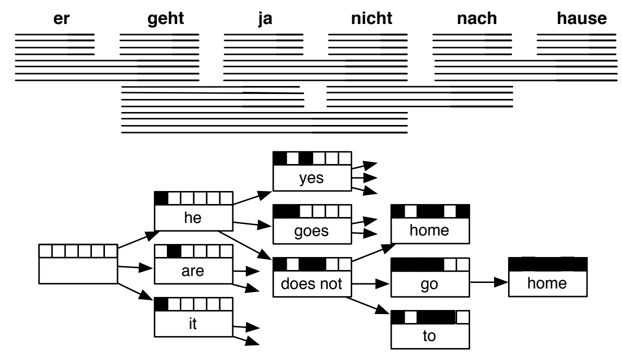
pick any translation option, create new hypothesis

Decoding: Hypothesis Expansion



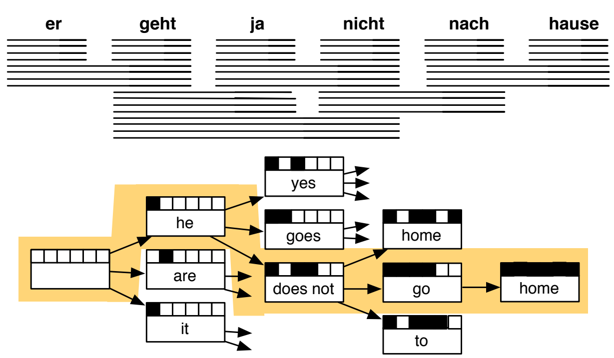
create hypotheses for all other translation options

Decoding: Hypothesis Expansion



also create hypotheses from created partial hypothesis

Decoding: Find Best Path



backtrack from highest scoring complete hypothesis

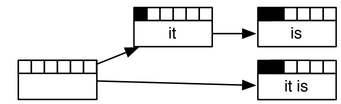
dynamic programming

Computational Complexity

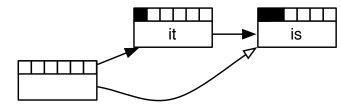
- The suggested process creates exponential number of hypothesis
- Machine translation decoding is NP-complete
- Reduction of search space:
  - recombination (risk-free)
  - pruning (risky)

Recombination

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

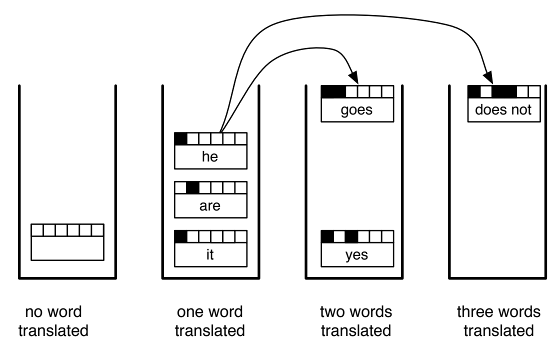


- Worse hypothesis is dropped



pruning

Stacks



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

Stack Decoding Algorithm


```
place empty hypothesis into stack 0
for all stacks 0...n − 1 do
  for all hypotheses in stack do
    for all translation options do
      if applicable then
        create new hypothesis
        place in stack
        recombine with existing hypothesis if possible
        prune stack if too big
      end if
    end for
  end for
end for
```


Pruning

- Pruning strategies
  - histogram pruning: keep at most k hypotheses in each stack
  - stack pruning: keep hypothesis with score $\alpha \times$ best score ($\alpha < 1$)

- Computational time complexity of decoding with histogram pruning

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

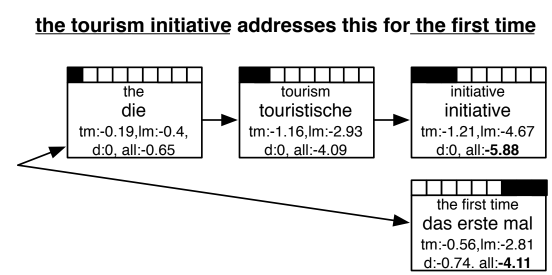
- Number of translation options is linear with sentence length, hence:

$$O(\text{max stack size} \times \text{sentence length}^2)$$

- Quadratic complexity

future cost estimation

Translating the Easy Part First?

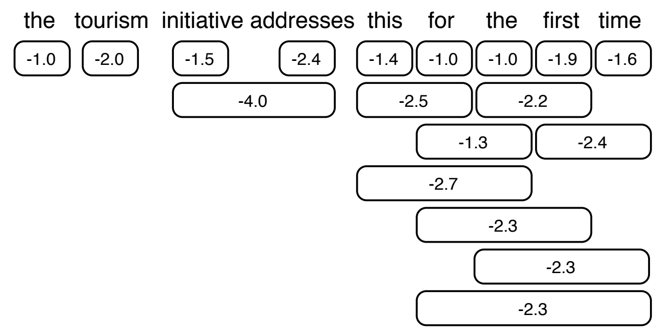


both hypotheses translate 3 words
worse hypothesis has better score

Estimating Future Cost

- Future cost estimate: how expensive is translation of rest of sentence?
- Optimistic: choose cheapest translation options
- Cost for each translation option
  - **translation model**: cost known
  - **language model:** output words known, but not context
    → estimate without context
  - **reordering model:** unknown, ignored for future cost estimation

Cost Estimates from Translation Options



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

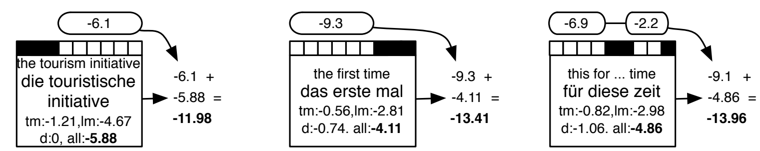
Cost Estimates for all Spans

- Compute cost estimate for all contiguous spans by combining cheapest options

first word	future cost estimate for n words (from first)								
	1	2	3	4	5	6	7	8	9
the	-1.0	-3.0	-4.5	-6.9	-8.3	-9.3	-9.6	-10.6	-10.6
tourism	-2.0	-3.5	-5.9	-7.3	-8.3	-8.6	-9.6	-9.6	
initiative	-1.5	-3.9	-5.3	-6.3	-6.6	-7.6	-7.6		
addresses	-2.4	-3.8	-4.8	-5.1	-6.1	-6.1			
this	-1.4	-2.4	-2.7	-3.7	-3.7				
for	-1.0	-1.3	-2.3	-2.3					
the	-1.0	-2.2	-2.3						
first	-1.9	-2.4							
time	-1.6								

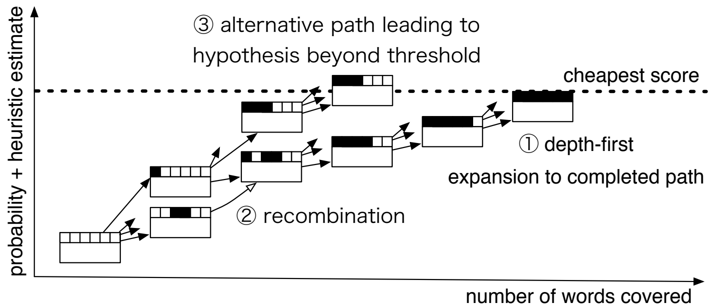
- Function words cheaper (the: -1.0) than content words (tourism -2.0)
- Common phrases cheaper (for the first time: -2.3)
  than unusual ones (tourism initiative addresses: -5.9)

Combining Score and Future Cost



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

A* Search



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