# Machine Translation

Computational Linguistics: Jordan Boyd-Graber
University of Maryland

WORD-BASED MODELS

Adapted from material by Philipp Koehn

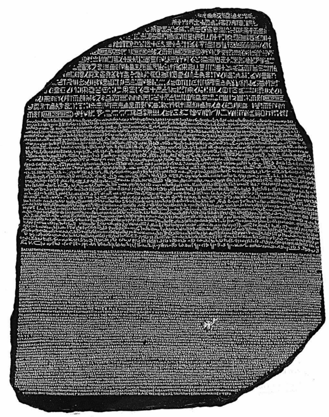
## Roadmap

- Introduction to MT
- Components of MT system
- Word-based models
- Beyond word-based models

## Roadmap

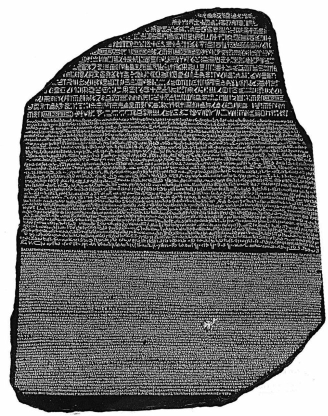
- Introduction to MT
- Components of MT system
- Word-based models
- Beyond word-based models: phrase-based and neural

## What unlocks translations?



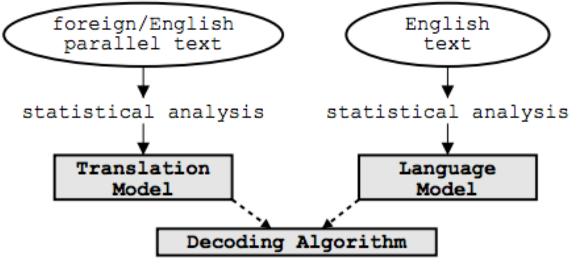
- Humans need parallel text to understand new languages when no speakers are round
- Rosetta stone: allowed us understand to Egyptian
- Computers need the same information

## What unlocks translations?



- Humans need parallel text to understand new languages when no speakers are round
- Rosetta stone: allowed us understand to Egyptian
- Computers need the same information
- Where do we get them?
  - Some governments require translations (Canada, EU, Hong Kong)
  - Newspapers
  - Internet

## Pieces of Machine Translation System



## Terminology

- Source language: **f** (foreign)
- Target language: **e** (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 |

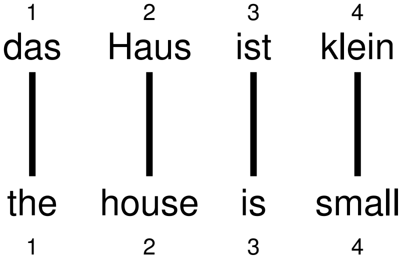
## Estimate Translation Probabilities

Maximum likelihood estimation

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

## Alignment

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



- Word positions are numbered 1–4

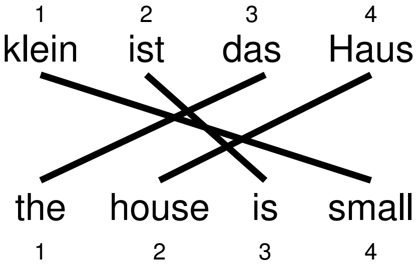
## Alignment Function

- 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 $a : i \longrightarrow j$
- Example

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

## Reordering

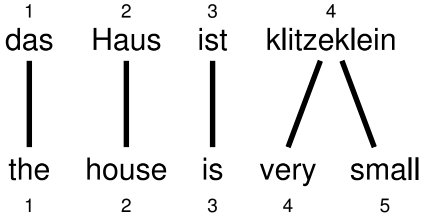
Words may be reordered during translation



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

## One-to-Many Translation

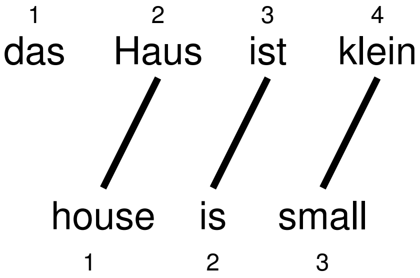
A source word may translate into multiple target words



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

## Dropping Words

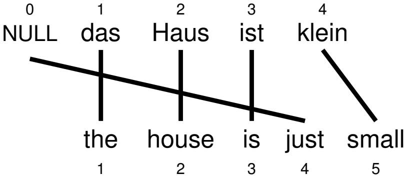
Words may be dropped when translated
(German article das is dropped)



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

## 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



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

## A family of lexical translation models

- A family translation models
- Uncreatively named: Model 1, Model 2, . . .
- Foundation of all modern translation algorithms
- First up: Model 1

## 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 $\epsilon$ is a normalization constant

# 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 $\epsilon$ is a normalization constant

## 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 $\epsilon$ is a normalization constant

## 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 $a : j \to i$

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

  - parameter $\epsilon$ is a normalization constant

## 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 $\epsilon$ is a normalization constant

## 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 $\epsilon$ is a normalization constant

## 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 $\epsilon$ is a normalization constant

## 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 $\epsilon$ 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 |
| family | 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 |

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

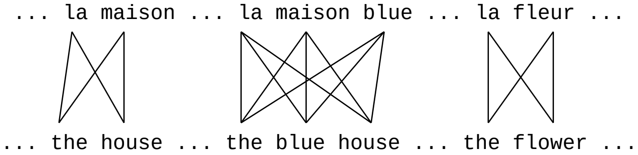
## Learning Lexical Translation Models

- 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,
    ⟶ we could estimate the parameters of our generative model
  - □ if we had the parameters,
    ⟶ we could estimate the alignments

## EM Algorithm

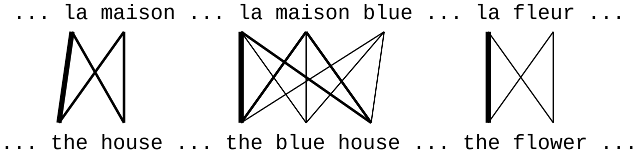
- Incomplete data
  - if we had complete data, would could estimate model
  - if we had model, we could fill in the gaps in the data
- Expectation Maximization (EM) in a nutshell
  1. initialize model parameters (e.g. uniform)
  2. assign probabilities to the missing data
  3. estimate model parameters 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

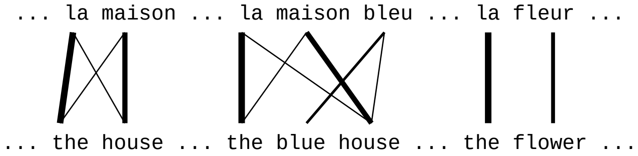


... la maison ... la maison blue ... la fleur ...

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

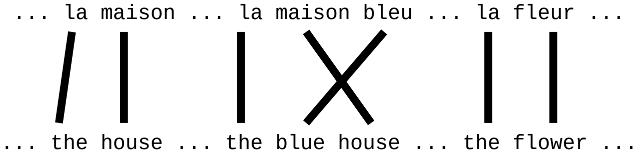
- After one iteration
- Alignments, e.g., between la and the are more likely

## EM Algorithm


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

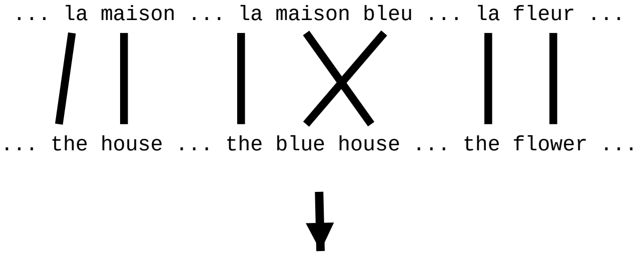
- After another iteration
- It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)

## 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(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

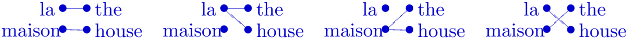
- We need to be able to compute:
  - Expectation-Step: probability of alignments
  - Maximization-Step: count collection

## IBM Model 1 and EM

- **Probabilities**

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

- **Alignments**



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

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

- **Counts**

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

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

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

## IBM Model 1 and EM: Expectation Step

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

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

## IBM Model 1 and EM: Expectation Step

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

$$
\begin{aligned}
p(\mathbf{e}|\mathbf{f}) &= \sum_{a} p(\mathbf{e}, a|\mathbf{f}) \\
&= \sum_{a(1)=0}^{l_f} \cdots \sum_{a(l_e)=0}^{l_f}
\end{aligned}
$$

## IBM Model 1 and EM: Expectation Step

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

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

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

## IBM Model 1 and EM: Expectation Step

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

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

## IBM Model 1 and EM: Expectation Step

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

## IBM Model 1 and EM: Expectation Step

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

## IBM Model 1 and EM: Expectation Step

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

- Note the algebra trick in the last line
  - removes the need for an exponential number of products
  - this makes IBM Model 1 estimation tractable

## The Trick

$$(\text{case} \quad l_e = l_f = 2)$$

$$\begin{aligned}
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} &= \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) = \\
&= t(e_1|f_0)\,t(e_2|f_0) + t(e_1|f_0)\,t(e_2|f_1) + t(e_1|f_0)\,t(e_2|f_2) + \\
&\quad + t(e_1|f_1)\,t(e_2|f_0) + t(e_1|f_1)\,t(e_2|f_1) + t(e_1|f_1)\,t(e_2|f_2) + \\
&\quad + t(e_1|f_2)\,t(e_2|f_0) + t(e_1|f_2)\,t(e_2|f_1) + t(e_1|f_2)\,t(e_2|f_2) = \\
&= t(e_1|f_0)\,(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_1)\,(t(e_2|f_1) + t(e_2|f_1) + t(e_2|f_2)) + \\
&\quad + t(e_1|f_2)\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2)) = \\
&= (t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))\,(t(e_2|f_2) + t(e_2|f_1) + t(e_2|f_2))
\end{aligned}$$

## 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

- 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

- 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

- 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

- 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:

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

## IBM Model 1 and EM: Pseudocode

```
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 sentence pairs ($\mathbf{e}$,$\mathbf{f}$) do
                                                    ▷ compute normalization
        for words $e$ in $\mathbf{e}$ do
            s-total($e$) = 0
            for words $f$ in $\mathbf{f}$ do
                s-total($e$) += $t(e|f)$
                                                    ▷ collect counts
        for words $e$ in $\mathbf{e}$ do
            for words $f$ in $\mathbf{f}$ do
                count($e|f$) += $\frac{t(e|f)}{\text{s-total}(e)}$
                total($f$) += $\frac{t(e|f)}{\text{s-total}(e)}$
```

```
while not converged (cont.) do
                                                    ▷ estimate probabilities
    for foreign words $f$ do
        for English words $e$ do
            $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
```

# IBM Model 1 and EM: Pseudocode

```
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 sentence pairs (e,f) do
                                                    ▷ compute normalization
        for words $e$ in e do
            s-total($e$) = 0
            for words $f$ in f do
                s-total($e$) += $t(e|f)$
                                                    ▷ collect counts
        for words $e$ in e do
            for words $f$ in f do
                count($e|f$) += $\frac{t(e|f)}{\text{s-total}(e)}$
                total($f$) += $\frac{t(e|f)}{\text{s-total}(e)}$
```

```
while not converged (cont.) do
                                    ▷ estimate probabilities
    for foreign words $f$ do
        for English words $e$ do
            $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
```

## IBM Model 1 and EM: Pseudocode

```
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 sentence pairs (e,f) do
                                        ▷ compute normalization
        for words $e$ in e do
            s-total$(e) = 0$
            for words $f$ in f do
                s-total$(e)$ += $t(e|f)$
                                        ▷ collect counts
        for words $e$ in e do
            for words $f$ in f do
                count$(e|f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
                total$(f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
```

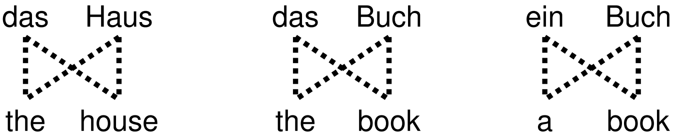
```
while not converged (cont.) do
                                        ▷ estimate probabilities
    for foreign words $f$ do
        for English words $e$ do
            $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
```

## IBM Model 1 and EM: Pseudocode

```
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 sentence pairs (e,f) do
                                                ▷ compute normalization
        for words $e$ in e do
            s-total$(e) = 0$
            for words $f$ in f do
                s-total$(e)$ += $t(e|f)$
                                                ▷ collect counts
        for words $e$ in e do
            for words $f$ in f do
                count$(e|f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
                total$(f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
```

```
while not converged (cont.) do
                                                ▷ estimate probabilities
    for foreign words $f$ do
        for English words $e$ do
            $t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
```

## Convergence



| *e* | *f* | initial | 1st it. | 2nd it. | ... | final |
|---|---|---|---|---|---|---|
| the | das | 0.25 | 0.5 | 0.6364 | ... | 1 |
| book | das | 0.25 | 0.25 | 0.1818 | ... | 0 |
| house | das | 0.25 | 0.25 | 0.1818 | ... | 0 |
| the | buch | 0.25 | 0.25 | 0.1818 | ... | 0 |
| book | buch | 0.25 | 0.5 | 0.6364 | ... | 1 |
| a | buch | 0.25 | 0.25 | 0.1818 | ... | 0 |
| book | ein | 0.25 | 0.5 | 0.4286 | ... | 0 |
| a | ein | 0.25 | 0.5 | 0.5714 | ... | 1 |
| the | haus | 0.25 | 0.5 | 0.4286 | ... | 0 |
| house | haus | 0.25 | 0.5 | 0.5714 | ... | 1 |

## 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

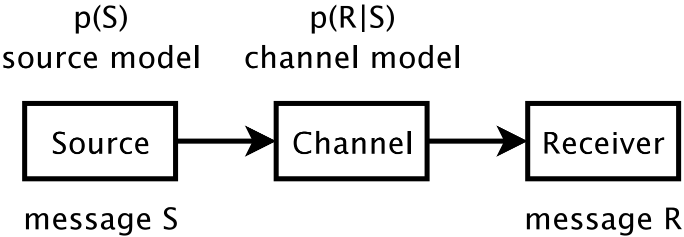
$$
\begin{aligned}
p(\mathbf{e}) &= p(e_1, e_2, \ldots, e_n) \\
&= p(e_1)p(e_2|e_1)\ldots p(e_n|e_1, e_2, \ldots, e_{n-1}) \\
&\simeq p(e_1)p(e_2|e_1)\ldots p(e_n|e_{n-2}, e_{n-1})
\end{aligned}
$$

## Noisy Channel Model

- We would like to integrate a language model
- Bayes rule

$$\begin{aligned}\operatorname{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) &= \operatorname{argmax}_{\mathbf{e}} \frac{p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e})}{p(\mathbf{f})} \\ &= \operatorname{argmax}_{\mathbf{e}} p(\mathbf{f}|\mathbf{e})\, p(\mathbf{e})\end{aligned}$$

## Noisy Channel Model



- 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
- Computationally 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

## Legacy

- 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?

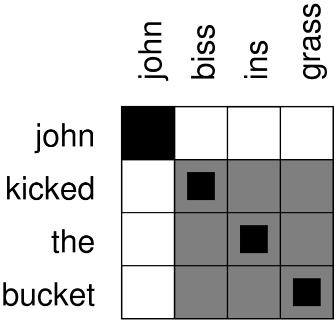
| | michael | geht | davon | aus | , | dass | er | im | haus | bleibt |
|---|---|---|---|---|---|---|---|---|---|---|
| michael | ■ | | | | | | | | | |
| assumes | | ■ | ■ | ■ | | | | | | |
| that | | | | | | ■ | | | | |
| he | | | | | | | ■ | | | |
| will | | | | | | | | | | ■ |
| stay | | | | | | |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

## Word Alignment?

| | john | wohnt | hier | nicht |
|---|---|---|---|---|
| john | ■ | | | |
| does | | ? | | ? |
| not | | | | ■ |
| live | | ■ | | |
| here | | | ■ | |

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

## Summary

- Lexical translation
- Alignment
- Expectation Maximization (EM) Algorithm
- Noisy Channel Model
- IBM Models
- Word Alignment

## Summary

- Lexical translation
- Alignment
- Expectation Maximization (EM) Algorithm
- Noisy Channel Model
- IBM Models
- Word Alignment
- Alternate models next

# Example

## Example

$$\frac{1.0}{(4+1)^4} .7 \cdot .8 \cdot .8 \cdot .4$$

## Example

$$\frac{1.0}{(4+1)^4} .7 \cdot .8 \cdot .8 \cdot .4 = 0.000287 \tag{1}$$

## Example

$$\frac{1.0}{(4+1)^4}.7 \cdot .8 \cdot .8 \cdot .4 = 0.000287 \tag{1}$$

$$\frac{1.0}{(4+1)}.7 \cdot .8 \cdot .8 \cdot .4$$

## Example

$$\frac{1.0}{(4+1)^4}.7 \cdot .8 \cdot .8 \cdot .4 = 0.000287 \tag{1}$$

$$\frac{1.0}{(4+1)}.7 \cdot .8 \cdot .8 \cdot .4 = 0.14 \tag{2}$$