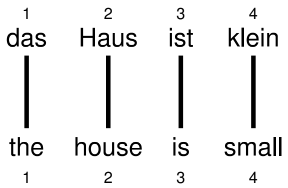
1 Word-based Models

Note: We always translate *from* a foreign ($\mathbf{f}$) language *into* English ($\mathbf{e}$).

1.1 Alignment function

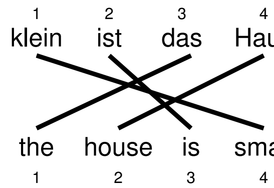
Mapping an English target word at position j to a German source word at position i with a function $a : j \to i$. Function a must be fully defined on the English (target) side. The reason for this is the noisy channel model (next lecture), where the output sentence is the *code* that gets transmitted and distorted, so we must account for every e.

Example 1: Alignment



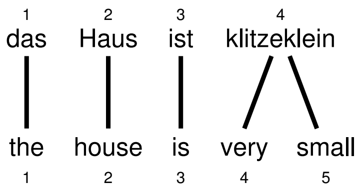
$a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 4\}$

Example 2: Reordering



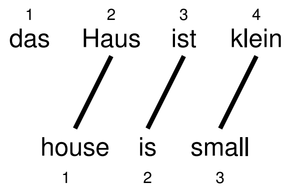
$a : \{1 \to 3, 2 \to 4, 3 \to 2, 4 \to 1\}$

Example 3: One-to-Many Translation



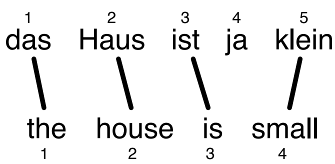
$a : \{1 \to 1, 2 \to 2, 3 \to 3, 4 \to 4, 5 \to 4\}$

Example 4: Dropping Words



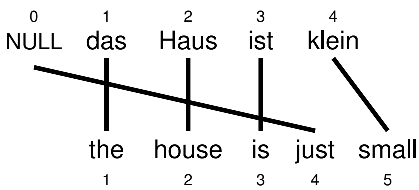
$a : \{1 \to 2, 2 \to 3, 3 \to 4\}$

Example 5: Dropping Words



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

Example 6: Inserting Words



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

1.2 Basics

Take now a simplified look at a parallel corpus (parallelism on sentence level) **with given alignments** and imaging having observed the following alignments possibilities for the word *Haus*:

Translation of *Haus*	**Count**
house	8,000
building	1,600
home	200
household	150
shell	50
total	**10,000**

We want to estimate the **lexical** translation probabilities from corpus statistics, i.e. the probability of foreign word f being translated as English translation e:

$$p_f : e \mapsto p_f(e).$$

It should be a probability function with usual properties of a probability distribution: $0 \leq p_f(e) \leq 1, \sum_e p_f(e) = 1, \forall f$.

1.2.1 Maximum Likelihood Estimation

How do we estimate $p_f(e)$ for $e = house$ and $f = Haus$?

$$p_{Haus}(house) \equiv p(house|Haus) = \frac{count(Haus \rightarrow house)}{count(Haus \rightarrow .)} = \frac{8,000}{10,000} = 0.8$$

For all translations of *Haus*, we get

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

Estimation based on ratios of counts is called **'maximum likelihood estimation'**.

1.2.2 Small Recap

Why the name "maximum likelihood"?

Suppose we observe data $\mathcal{D} = \{(e_i, f_i); i = \dots N\}$ in a form of aligned word pairs, assumed to be generated independently with probabilities $\theta(e_i, f_j) = \theta_{ij}$. Data likelihood:

$$\ell(\mathcal{D};\theta) = \prod_{i=1}^{N} \theta(e_i, f_i) = \prod_{(e,f)} \theta(e,f)^{n_{ef}} = \prod_{(e,f)} \theta_{e,f}^{n_{ef}},$$

where n_{ef} are counts of a particular aligned pair (e, f) in the data set.

Maximum likelihood estimation of a set of parameters $\{\theta_{ef}\}$

$$\{\theta^*(e,f)\} = \arg\max_{\theta_{ef}} \ell(\mathcal{D};\theta) = \arg\max_{\theta_{ef}} \log \ell(\mathcal{D};\theta),$$

under constrains $\sum_e \theta_{ef} = 1, \forall f$.

Lagrangian function:

$$\begin{aligned}
\mathcal{L} &= \log \ell(\mathcal{D}) - \sum_f \lambda_f (\sum_e \theta_{ef} - 1) \\
&= \sum_{(e,f)} n_{ef} \log \theta_{ef} - \sum_f \lambda_f (\sum_e \theta_{ef} - 1). \\
\frac{\partial \mathcal{L}}{\partial \theta_{e'f'}} &= n_{e'f'}/\theta_{e'f'} - \lambda_{f'}. \\
&\text{set derivatives to zero} \\
&\Rightarrow \theta_{e'f'} = n_{e'f'}/\lambda_{f'} \\
\text{using the constraint } \sum_e \theta_{ef} &= 1, \text{ we get} \\
\sum_{e'} n_{e'f'} &= \lambda_{f'} \\
&\text{solving for } \theta\text{'s} \\
\theta^*_{ef} &= \frac{n_{ef}}{\sum_{e'} n_{e'f'}}
\end{aligned}$$

1.3 IBM Model 1

IBM Models in general:
Generative models, which break up the translation process into smaller steps and achieve better statistics with simpler models.

IBM Model 1 uses only lexical translation. Ignores any position information (order), resulting in translating multisets of words into multisets of words.

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
- translation probability $t(e|f) \equiv p(e|f)$ (t-tables)
- with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \to i$

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

$\prod_{j=1}^{l_e} t(e_j|f_{a(j)})$ is the product over the lexical translation probabilities for all l_e generated target words. We use the product, since we assume that the lexical translation probabilities are independant.
ϵ is a normalization constant, s.t. $\sum_{\mathbf{e},a} p(\mathbf{e}, a|\mathbf{f}) = 1$. or a distribution of lengths $\epsilon(l_e|l_f)$.
$(l_f + 1)^{l_e}$ is the number of alignments of l_f+NULL input words with l_e output words: the uniform probabilities over alignments.

Can also be defined the reverse direction: $p(\mathbf{f}, a|\mathbf{e})$ (original IBM1).

Generative story for IBM translation Model 1:

1. pick a length l_e for $\mathbf{e}$ according to distribution $\epsilon(l_e|l_f)$
2. for each $j = 1, \ldots, l_e$ choose avalue for a_j from $0, 1, \ldots, l_e$ according to uniform distribution
3. for each $j = 1, \ldots, l_e$ choose a output word e_j according to $t(e_j|f_{a_j})$

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

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

1.4 Learning Lexical Translation Models

We would like to estimate the lexical translation probabilities $t(e|f)$ (and $t(f|e)$) from a corpus of parallel translations.
Problem: We don't have the alignments, only parallel sentences (i.e., sentences in source language, paired with sentences that are translations in target language).

Chicken-and-egg problem caused by **incomplete** data:

machine translation	**machine learning**
If we had alignments, we could estimate $t(e\|f)$ by relative frequency count.	If we had complete data, we could estimate the model by Maximum Likelihood Estimation.
If we had the model $t(e\|f)$, we could assign most probable alignments.	If we had the model, we could complete our data by most probable predictions.

1.4.1 Concave functions

Let the domain of a function g be denoted by $\mathbf{dom}(g)$.

Definition A function $g : \mathbb{R}^n \to \mathbb{R}$ is concave if and only if $\mathbf{dom}(g)$ is a convex set and for all $x, y \in \mathbf{dom}(g)$ and $0 \leq \alpha \leq 1$

$$g(\alpha x + (1-\alpha)y) \geq \alpha g(x) + (1-\alpha)g(y).$$

Strict concativity differs by requiring $x \neq y$ and strict inequality in the last expression.

Any local maximum of a concave function is also a global maximum. A strictly concave function will have at most one global maximum.

Without knowing the alignment function, the probability of a translation should account for all possible alignments (marginalize alignments out): $p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, a|\mathbf{f})$. Then the probability (1) rewrites:

$$p(\mathbf{e}|\mathbf{f}) = \sum_a \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a_j}) = \text{{\tiny (see next lecture why)}} = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_a t(e_j|f_{a_j}),$$

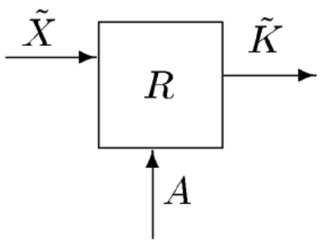
where $a_j = a(j)$. The sum over all possible alignments ($\sum_a$) can be rewriten as the sum over all possible values of a_j (all positions in the $\mathbf{f}$): $\sum_{a_j=0}^{l_f}$.

Now we can write the log-likelihood under IBM Model 1:

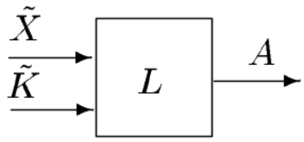
$$\log \ell(\mathcal{D}; t) = \log \prod_{\mathbf{e},\mathbf{f} \in \mathcal{D}} p(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{e},\mathbf{f} \in \mathcal{D}} \sum_{j=1}^{l_e} \log \sum_{i=0}^{l_f} t(e_j|f_i) + \text{const}$$

We see that the above likelihood is concave (prove it). However, it is not strictly concave (important for convergence of the EM algorithm to a unique maximum).

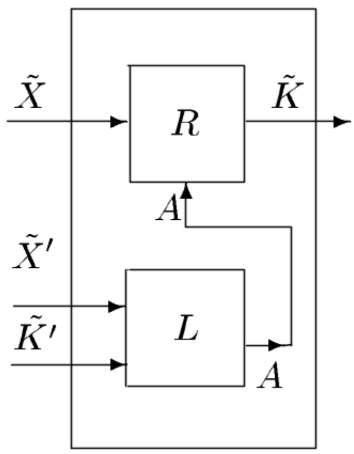
1.4.2 Unsupervised learning idea



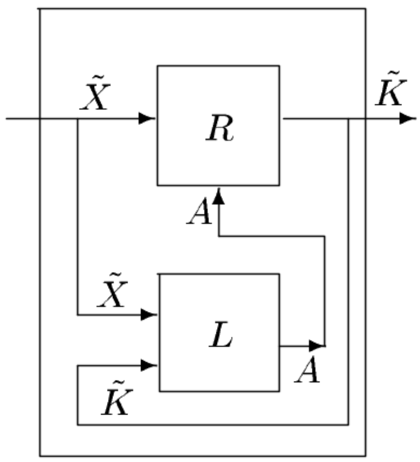
(a) Varying classifier.



(b) Learning as an estimate of parameter A.



(c) Learning classifier.

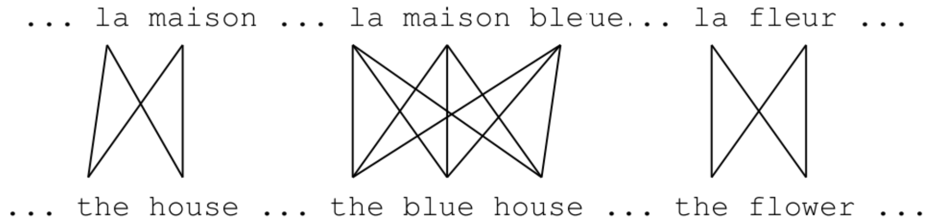


(d) Classifier based on unsupervised learning.

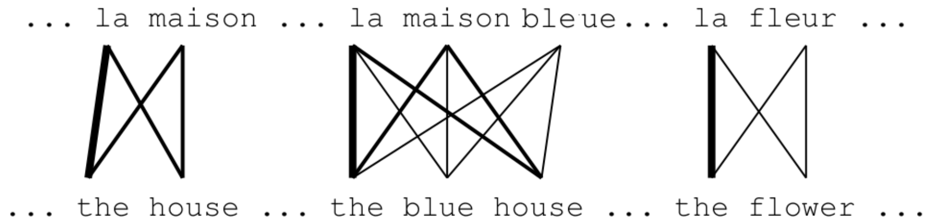
1.4.3 EM Algorithm

EM (Expectation Maximization) in a nutshell: 1. Initialize model parameters, e.g., uniform. 2. Assign probabilities to missing data. 3. Estimate model parameters from completed/manufactured/expected data. 4. Iterate step 2 - 3 until convergence.

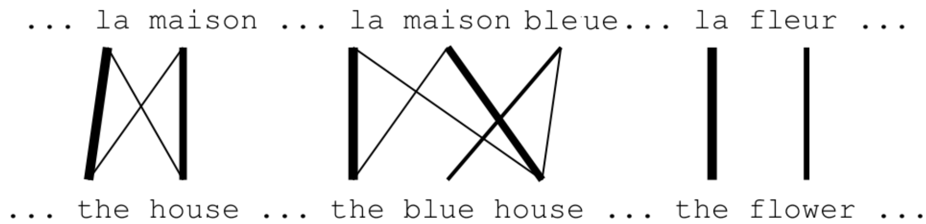
Initial step:
All alignments are equally likely. The Model learns that, e.g., "la" is often aligned with "the".



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



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



Convergence:
Inherent hidden structure revealed by EM.

$$\begin{aligned} p(\text{la}|\text{the}) &= 0.453 \\ p(\text{le}|\text{the}) &= 0.334 \\ p(\text{maison}|\text{house}) &= 0.876 \\ p(\text{bleue}|\text{blue}) &= 0.563 \end{aligned}$$