CA4012

Statistical Machine Translation

Week 6: IBM Models

Lecturer: Dimitar Shterionov

E-mail: (dimitar.shterionov@adaptcentre.ie)

Lab Tutors: Eva Vanmassenhove, Alberto Poncelas

2nd Semester, 2018-2019 Academic Year

Content

IBM Model 1

Higher IBM Models

Word Alignment

Exercises

Recap & Quiz

- What's the purpose of a translation model?
- The translation model $p(\boldsymbol{e}|\boldsymbol{f})$ is used to reflect how likely an English sentence ***e*** could be the translation of a foreign sentence ***f***, and it evaluates the adequacy of a translation given a source sentence.

Recap & Quiz

- When we build a translation model, why do we calculate lexical translation probabilities rather than directly computing sentence-level probabilities?
- Limited by the size of the parallel corpus, it is not feasible to directly estimate the translation probability for a sentence pair, and it is not reasonable to give zero probability to a sentence pair that is never seen in the corpus. However, it is more feasible to decompose the sentences into words to calculate lexical translation probabilities, and then obtain translation probabilities at the sentence-level.

Recap & Quiz

- What's the EM algorithm? How do we use it in the task of word alignment?
- EM is a parameter estimation technique for incomplete data (hidden variable and unknown parameters), which includes three steps: initialization, E-step and M-step.
- For word alignment task, we first initialize model parameters to fill in the gap in the incomplete data, and then apply the parameters to the hidden data to obtain the possible alignments, and then collects counts and estimate new model parameters. Finally we repeat step 2 and 3 until convergence.

Concept of IBM Models

- Motivation:
  - SMT is based on Bayes Rule and noisy channel model: $p(\mathbf{e}|\mathbf{f}) \rightarrow p(\mathbf{e})p(\mathbf{f}|\mathbf{e})$
  - Sentence probabilities are products of word probabilities
  - Word probabilities are estimated from the incomplete data
- IBM Model:
  - A statistical model that generates a number of different translations for a sentence,
  - Each sentence probability is based on lexical translation probabilities and alignments.

IBM Models

- Five models of increasing complexity were proposed in the original work on statistical machine translation at IBM.
  - IBM Model 1: lexical translation probability distribution

IBM Models

- Five models of increasing complexity were proposed in the original work on statistical machine translation at IBM.
  - IBM Model 1: lexical translation probability distribution
  - IBM Model 2: adds absolute alignment model;

IBM Models

- Five models of increasing complexity were proposed in the original work on statistical machine translation at IBM.
  - IBM Model 1: lexical translation probability distribution
  - IBM Model 2: adds absolute alignment model;
  - IBM Model 3: adds fertility model;

IBM Models

- Five models of increasing complexity were proposed in the original work on statistical machine translation at IBM.
  - IBM Model 1: lexical translation probability distribution
  - IBM Model 2: adds absolute alignment model;
  - IBM Model 3: adds fertility model;
  - IBM Model 4: adds relative alignment model;

IBM Models

- Five models of increasing complexity were proposed in the original work on statistical machine translation at IBM.
  - IBM Model 1: lexical translation probability distribution
  - IBM Model 2: adds absolute alignment model;
  - IBM Model 3: adds fertility model;
  - IBM Model 4: adds relative alignment model;
  - IBM Model 5: fixes deficiency.

Recall: Some Notations

Given a sentence-aligned text, we have the following notations:

Source **f**: Foreign (e.g. German)

Target **e**: English

f: a word in **f**

e: a word in **e**

$l_{\mathbf{f}}$: length of **f** (i=1…)

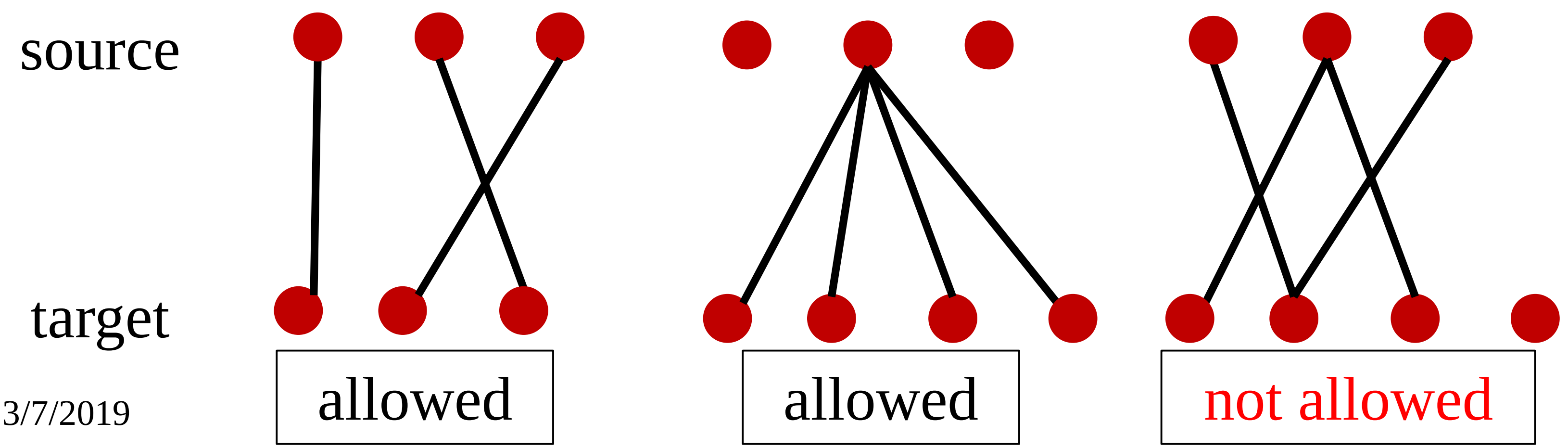
$l_{\mathbf{e}}$: length of **e** (j=1…)

$t(e|f;\ \mathbf{e},\mathbf{f})$: lexical translation probability

Alignment: $a: j \rightarrow i$ or $a_j = i$

IBM Model 1

- Alignments:
  - one-to-one alignments are allowed
  - many-to-one alignments are allowed
  - However, one-to-many alignments are still not allowed



IBM Model 1

- Given:
  - Foreign sentence **f**
  - English sentence **e**
  - the alignment function *a*, with the probability *t*
  - a normalization factor ε (required to turn the formula into a proper probability function)
- Do:
  - Compute the product of probabilities for each English word e_j is generated by a foreign word $f_{a(j)}$
  - Normalize using ε

IBM Model 1

$$p^*(\mathbf{e}, a|\mathbf{f})$$
$$= t(e_1|f_{a(1)}) * t(e_2|f_{a(2)}) * \cdots * t\left(e_{l_\mathbf{e}}|f_{a(l_\mathbf{e})}\right)$$
$$= \prod_{j=1}^{l_\mathbf{e}} t(e_j|f_{a(j)})$$

IBM Model 1

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\varepsilon}{(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}}} * p^*(\mathbf{e}, a|\mathbf{f})$$

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

EM and IBM Model 1

Expectation-Step:

- Apply model to the data,
- Then we need to compute the probabilities of different alignments given a sentence pair.

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

- Compute:

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

- We need to derive the denominator $p(\mathbf{e}|\mathbf{f})$.

IBM Model 1

$$p(\mathbf{e}|\mathbf{f}) = \sum_{a} p(\mathbf{e}, a|\mathbf{f}) = \sum_{a} \frac{\varepsilon}{(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}}} \prod_{j=1}^{l_{\mathbf{e}}} t(e_j | f_{a_j})$$

Because: $\forall j \in \{1, \dots, l_{\mathbf{e}}\}, a_j \in \{0, \dots, l_{\mathbf{f}}\}$

$$p(\mathbf{e}|\mathbf{f}) = \sum_{a_{1=0}}^{l_{\mathbf{f}}} \sum_{a_{2=0}}^{l_{\mathbf{f}}} \dots \sum_{a_{l_e=0}}^{l_{\mathbf{f}}} \frac{\varepsilon}{(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}}} \prod_{j=1}^{l_{\mathbf{e}}} t(e_j | f_{a_j})$$

IBM Model 1

$$p(\mathbf{e}|\mathbf{f}) = \sum_{a} p(\mathbf{e}, a|\mathbf{f}) = \sum_{a} \frac{\varepsilon}{(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}}} \prod_{j=1}^{l_{\mathbf{e}}} t(e_j|f_{a_j})$$

Because: $\forall j \in \{1, \dots, l_{\mathbf{e}}\}, a_j \in \{0, \dots, l_{\mathbf{f}}\}$

$$p(\mathbf{e}|\mathbf{f}) = \underbrace{\sum_{a_1=0}^{l_{\mathbf{f}}} \sum_{a_2=0}^{l_{\mathbf{f}}} \dots \sum_{a_{l_e}=0}^{l_{\mathbf{f}}} \frac{\varepsilon}{(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}}}}_{} \prod_{j=1}^{l_{\mathbf{e}}} t(e_j|f_{a_j})$$

Complexity: $(l + 1)^{l_{\mathbf{e}}} \times l_{\mathbf{e}}$: intractable

IBM Model 1

However, we can derive the formula as:

$$p(\boldsymbol{e}|\boldsymbol{f}) = \sum_{a_{1=0}}^{l_f} \sum_{a_{2=0}}^{l_f} \dots \sum_{a_{l_e=0}}^{l_f} \frac{\varepsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a_j})$$

$$= \frac{\varepsilon}{(l_f+1)^{l_e}} \boxed{\sum_{a_{1=0}}^{l_f} \sum_{a_{2=0}}^{l_f} \dots \sum_{a_{l_e=0}}^{l_f}} \prod_{j=1}^{l_e} t(e_j|f_{\boxed{a}_j})$$

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

Complexity: $(l+1) \times m$: tractable

IBM Model 1

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

Trick: Factoring Out

Given the sentence pair:

the house – la maison

Calculate $p(\mathbf{e}|\mathbf{f})$ for all alignments and then perform the factoring out trick.

Trick: Factoring Out

- $p(\mathbf{e}|\mathbf{f})$ for all alignments:

$$\sum_{a_1=0}^{l_\mathbf{f}} \sum_{a_2=0}^{l_\mathbf{f}} \cdots \sum_{a_{l_\mathbf{e}}=0}^{l_\mathbf{f}} \frac{\varepsilon}{(l_\mathbf{f}+1)^{l_\mathbf{e}}} \prod_{j=1}^{l_\mathbf{e}} t\left(e_j \middle| f_{a_j}\right)$$

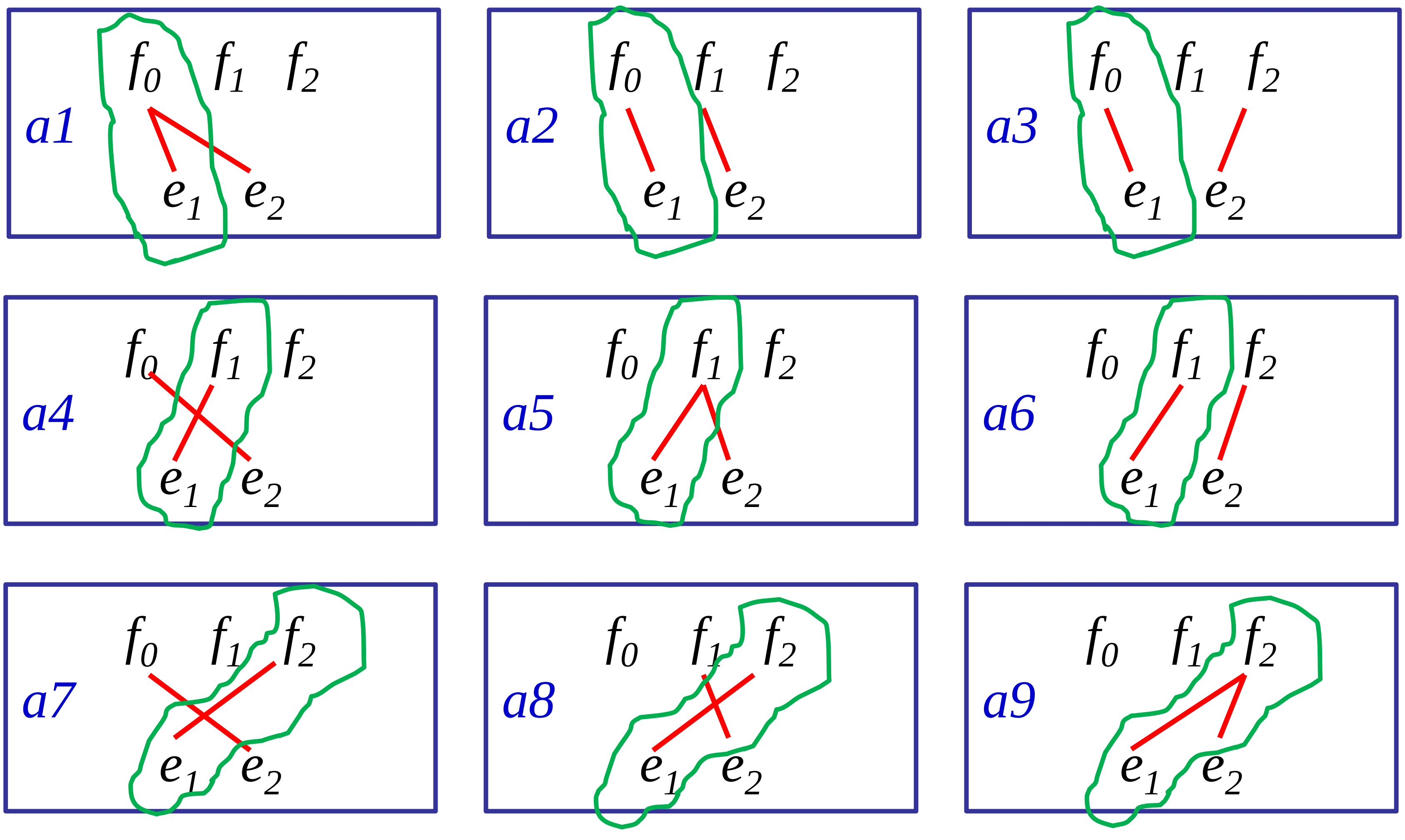
Trick: Factoring Out

- List all possible alignments for

the house – NULL la maison

- Calculate how many possible alignments for this sentence pair?
  - $(l_{\mathbf{f}} + 1)^{l_{\mathbf{e}}} = (2 + 1)^2 = 9$

Trick: Factoring Out



Trick: Factoring Out

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

$$t(e_1/f_0)*t(e_2/f_0)+t(e_1/f_0)*t(e_2/f_1)+t(e_1/f_0)*t(e_2/f_2)$$
$$+\ t(e_1/f_1)*t(e_2/f_0)+t(e_1/f_1)*t(e_2/f_1)+t(e_1/f_1)*t(e_2/f_2)$$
$$+\ t(e_1/f_2)*t(e_2/f_0)+t(e_1/f_2)*t(e_2/f_1)+t(e_1/f_2)*t(e_2/f_2)$$

Trick: Factoring Out

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

$$t(e_1/f_0)*t(e_2/f_0)+t(e_1/f_0)*t(e_2/f_1)+t(e_1/f_0)*t(e_2/f_2)$$
$$+\ t(e_1/f_1)*t(e_2/f_0)+t(e_1/f_1)*t(e_2/f_1)+t(e_1/f_1)*t(e_2/f_2)$$
$$+\ t(e_1/f_2)*t(e_2/f_0)+t(e_1/f_2)*t(e_2/f_1)+t(e_1/f_2)*t(e_2/f_2)$$

Trick: Factoring Out

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

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

Trick: Factoring Out

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

$(t(e_1/f_0) + t(e_1/f_1) + t(e_1/f_2)) * (t(e_2/f_0) + t(e_2/f_1) + t(e_2/f_2))$

Trick: Factoring Out

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

$$(t(e_1/f_0) + t(e_1/f_1) + t(e_1/f_2)) * (t(e_2/f_0)+t(e_2/f_1)+t(e_2/f_2))$$

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

Trick: Factoring Out

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

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

$$= \frac{\epsilon}{3^2}\prod_{j=1}^{l_e}\sum_{i=0}^{l_f} t(e_j|f_i)$$

Trick: Factoring Out

- Combine what we have

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

$$= \frac{\frac{\varepsilon}{(l_{\mathbf{f}}+1)^{l_{\mathbf{e}}}} \prod_{j=1}^{l_{\mathbf{e}}} t(e_j|f_{a(j)})}{\frac{\varepsilon}{(l_{\mathbf{f}}+1)^{l_{\mathbf{e}}}} \prod_{j=1}^{l_{\mathbf{e}}} \sum_{i=0}^{l_{\mathbf{f}}} t(e_j|f_i)}$$

$$= \prod_{j=1}^{l_{\mathbf{e}}} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_{\mathbf{f}}} t(e_j|f_i)}$$

More Simplification for IBM Model 1

- M-Step: collect fractional counts

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

More Simplification for IBM Model 1

- With the same simplification (Trick) as before:

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

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

Exercise: Derive the last line.

Simplified EM for IBM Model 1

Step 1: Initialize model parameters $p(\mathbf{e}|\mathbf{f})$

Step 2: Collect counts for word pair (e, f)

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

Step 3: Estimate new model parameters

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

Iterate untill convergence.

IBM Model 1 and EM: Pseudocode

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

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
```

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
```

IBM Model 1 and EM: Pseudocode

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

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
```

IBM Model 1 and EM: Pseudocode

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

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
```

IBM Model 1 and EM: Pseudocode

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

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
```

IBM Model 1 and EM: Pseudocode

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

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
```

IBM Model 1 and EM: Pseudocode

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

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
```

IBM Model 1 and EM: Pseudocode

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

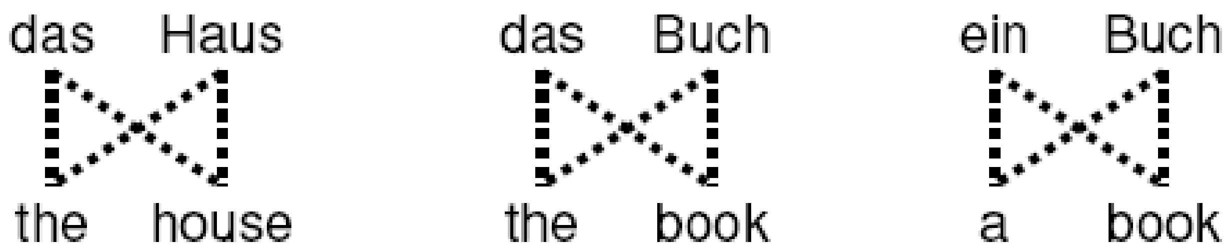
Output: translation prob. $t(e|f)$

```
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

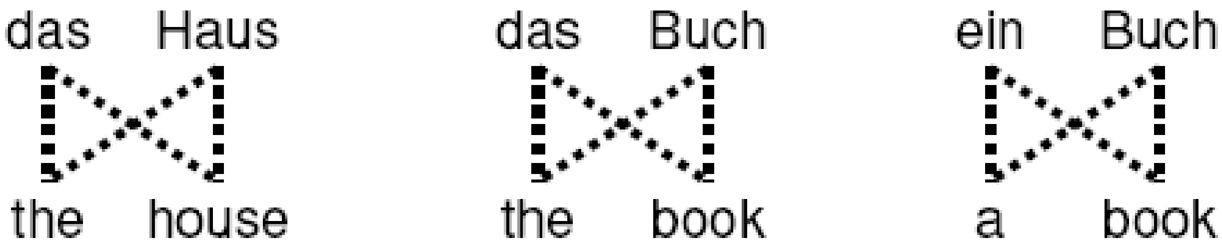
- Goal of EM:
  - Find a model ($p(\mathbf{e}|\mathbf{f})$) that best fits the data.
- How can we measure whether we have accomplished this goal?
  - its quality will ultimately be measured by how well it translates new, previously unseen sentences.
  - however, we have the training data and we can measure how well our model translates this data.

Convergence



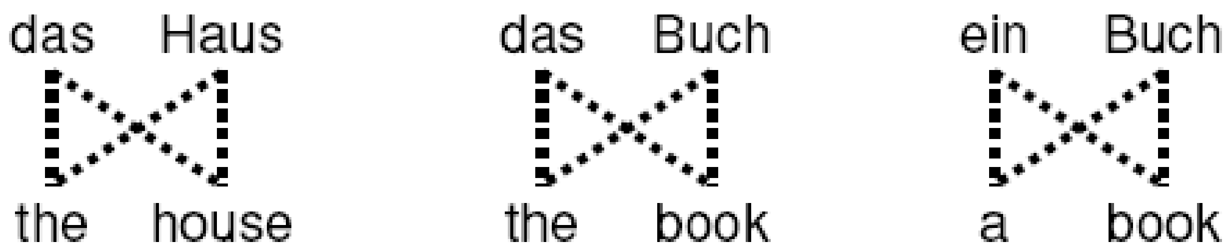
e	f	initial	1st it.	2nd it.	3rd it.	...	final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	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

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

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

Convergence

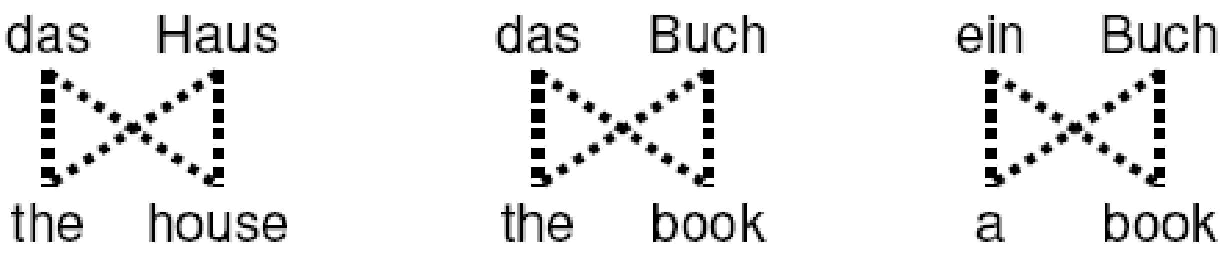
- Calculate the translation probability for the sentence pair:

$$p(\text{the book}|\text{das buch})$$

- At initial step:
  - uniformly distributed
  - Sum over the length of **f**
  - Multiply over the length of **e**
  - Normalize

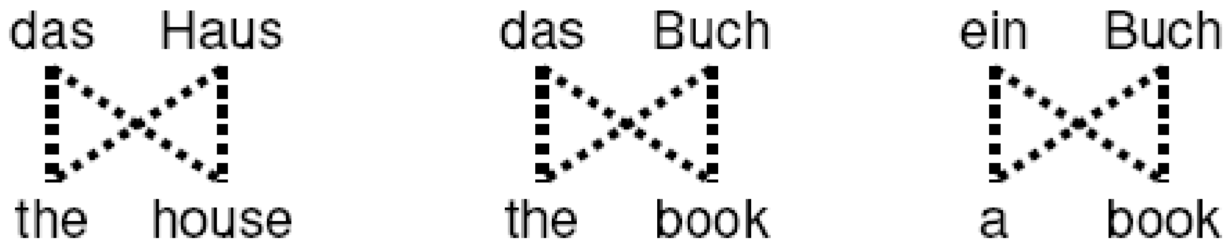
at the initial step and at the final iteration.

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

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

Convergence

$$p(\mathbf{e}|\mathbf{f}) = \frac{\varepsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

- At the initial, we have

$$p(\text{the book}|\text{das buch}) \\ = \frac{\epsilon}{2^2}(0.25 + 0.25)(0.25 + 0.25) \\ = 0.0625\,\epsilon$$

Convergence

- At the final, we have

$$p(\text{the book}|\text{das buch}) = \frac{\epsilon}{2^2}(1+0)(1+0) = 0.25\,\epsilon$$

- We see a steady improvement of the likelihood of the output, given the input side and our model.

Convergence

- We can measure this progress by the perplexity of the model:

$$PP = 2^{-\sum_s \log_2 p(\mathbf{e}_s|\mathbf{f}_s)}$$

- Theoretically, the perplexity is guaranteed to decrease or stay the same at each iteration.
- This also guarantees that EM training converges to a global minimum for IBM Model 1.

Convergence

$$PP = 2^{-\sum_s \log_2 p(\mathbf{e}_s|\mathbf{f}_s)}$$

	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 (ϵ=1)	4096	202.3	153.6	131.6	...	113.8

Content

- IBM Model 1
- **Higher IBM Models**
- Word Alignment
- Exercises

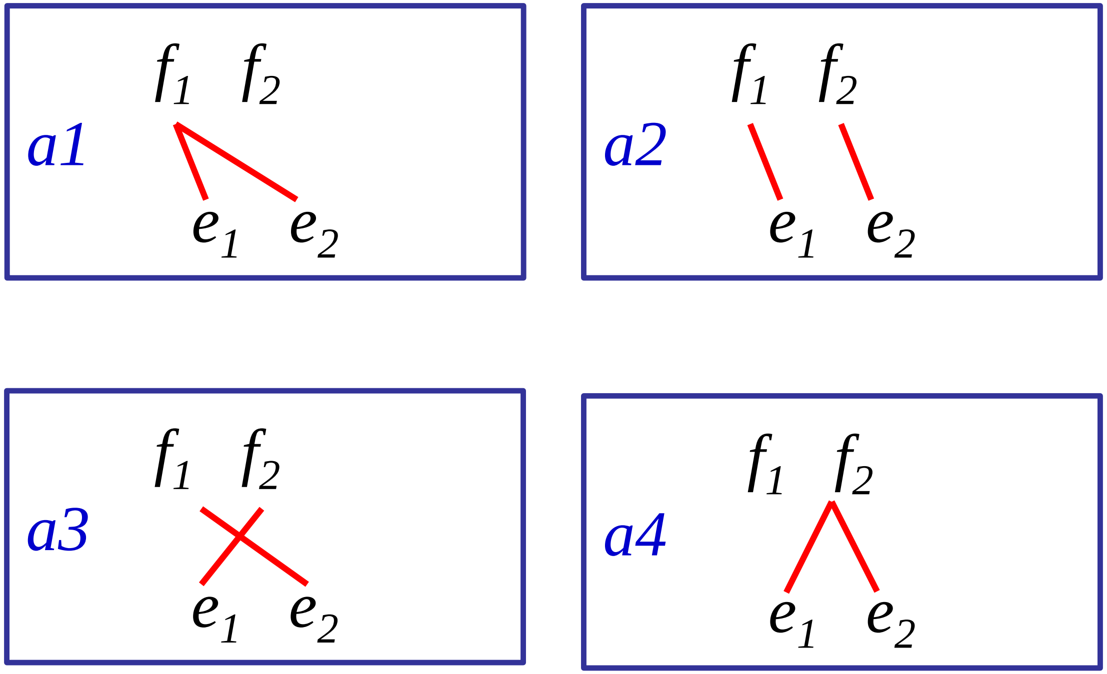
Exercise

- Draw all possible alignments for the following sentence:

keku xuexi ---- study hard

- The source side is Chinese and the target side is English. One target word is only allowed to align to one source word.
- The *NULL* word in the source side is ignored.

Solution

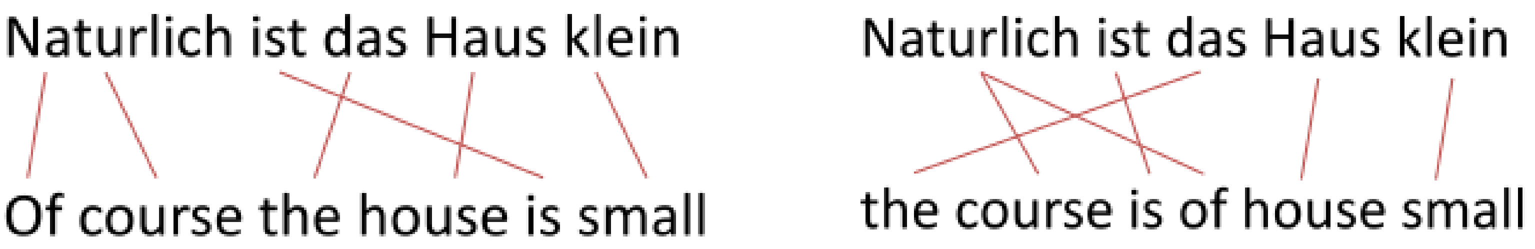


Motivation

- Is IBM Model 1 good enough?
- Can you see the potential problem?

Deficiencies of IBM Model 1

- With IBM Model 1, both the following translations would be given the same probability:

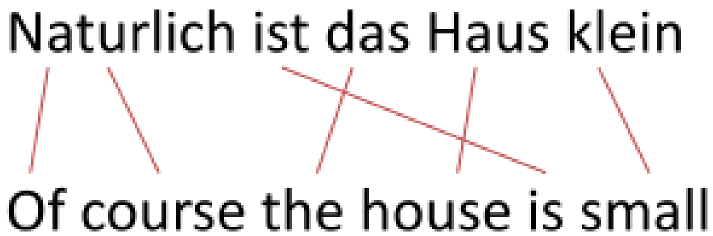


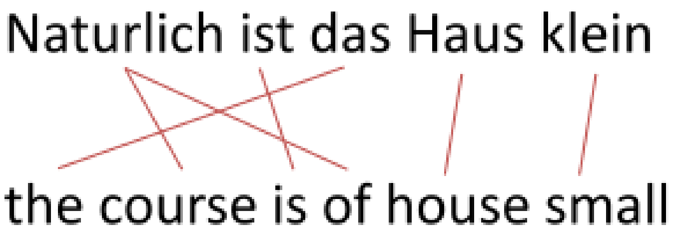
Deficiencies of IBM Model 1

- With IBM Model 1, both the following translations would be given the same probability:

WHY?

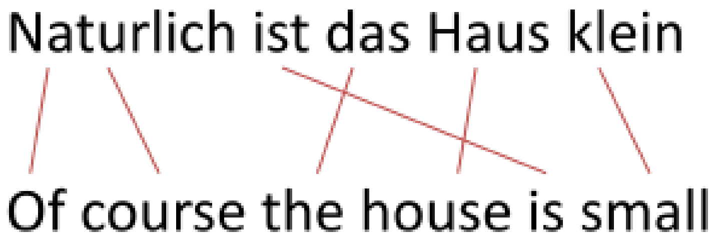
Deficiencies of IBM Model 1

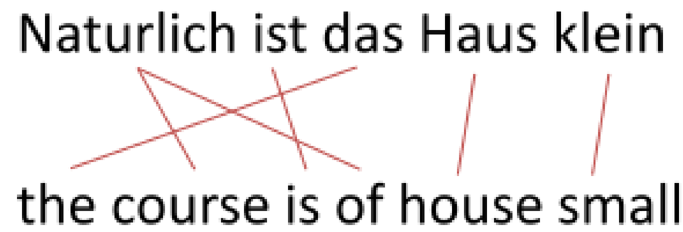




- List all lexical translation pairs for each alignment
  - t(of|Naturlich)
  - t(course|Naturlich)
  - t(is|ist)
  - t(the|das)
  - t(house|Haus)
  - t(small|klein)

Deficiencies of IBM Model 1



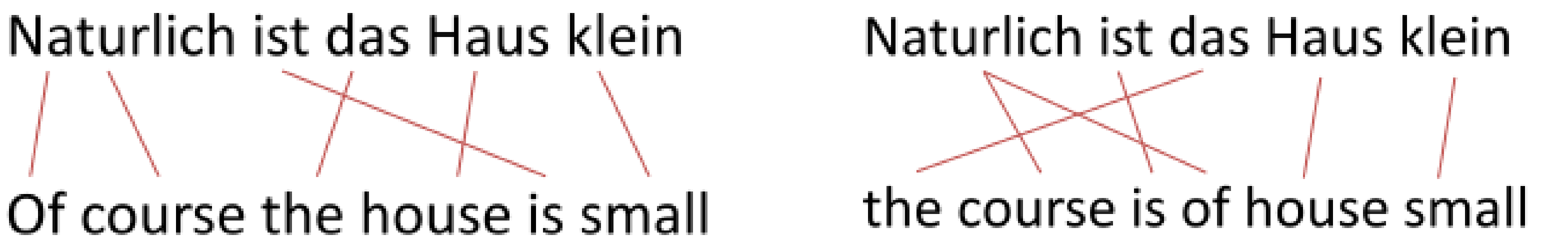


- For each alignment, we have

$$p(\mathbf{e},a|\mathbf{f}) = \mathrm{t}(\mathrm{of}|\mathrm{Naturlich}) * \mathrm{t}(\mathrm{course}|\mathrm{Naturlich}) * \mathrm{t}(\mathrm{is}|\mathrm{ist}) * \mathrm{t}(\mathrm{the}|\mathrm{das}) * \mathrm{t}(\mathrm{house}|\mathrm{Haus}) * \mathrm{t}(\mathrm{small}|\mathrm{klein})$$

The results are the same!

Deficiencies of IBM Model 1



However,

the course is of house small

is not a meaningful sentence!

Deficiencies of IBM Model 1

- IBM Model 1 is very weak:
  - in terms of reordering, because it regards all possible reorderings as equally likely.
  - adding and dropping words.

Facts for Natural Languages

- Word order
  - For many languages, words that follow each other in one language have translations that follow each other in the output language.
- Fertility
  - One word in the input language translates into one single word in the output language.
  - some words produce multiple words or get dropped (producing zero words).

Higher Models

- Advances of 5 IBM Models
  - IBM Model 1: lexical translation;
  - IBM Model 2: adds absolute alignment model;
  - IBM Model 3: adds fertility model;
  - IBM Model 4: adds relative alignment model;
  - IBM Model 5: fixes deficiency.
- Only IBM Model 1 has global maximum
  - training of a higher IBM model builds on previous model
  - In GIZA++, we generally run

IBM Model 1 3~5 times -> HMM 3~5 times -> IBM 3 3~5 times -> IBM4 3~5 times

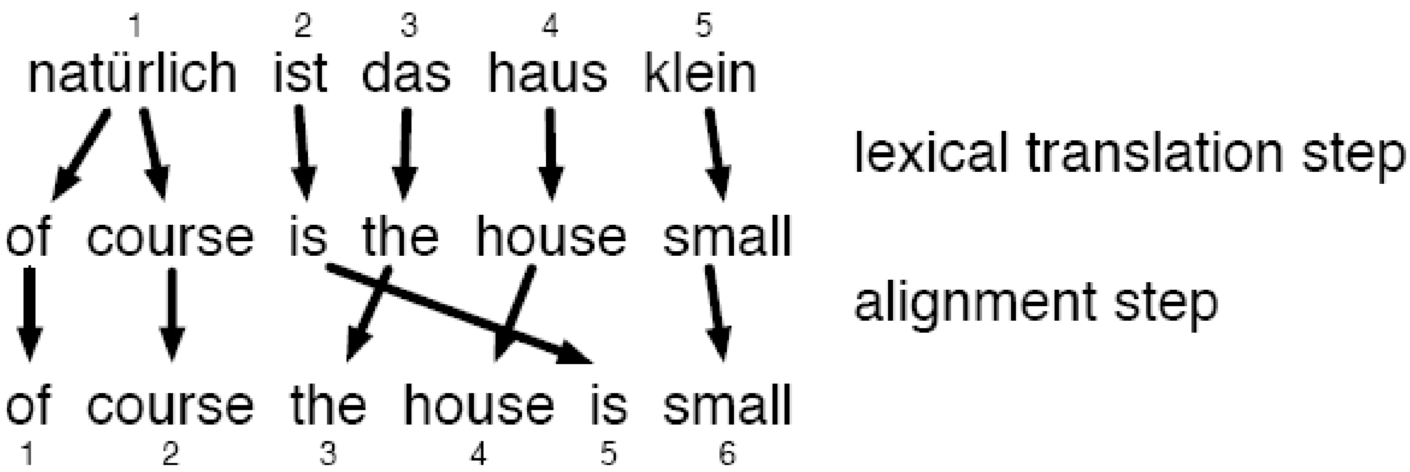
IBM Model 2

- Adding an explicit model for alignment
  - based on the positions of the input and output words:
  - the translation of a foreign input word in position ***i*** to an English word in position *j* is modeled by an alignment probability distribution

$$a(i|j, l_{\mathbf{e}}, l_{\mathbf{f}})$$

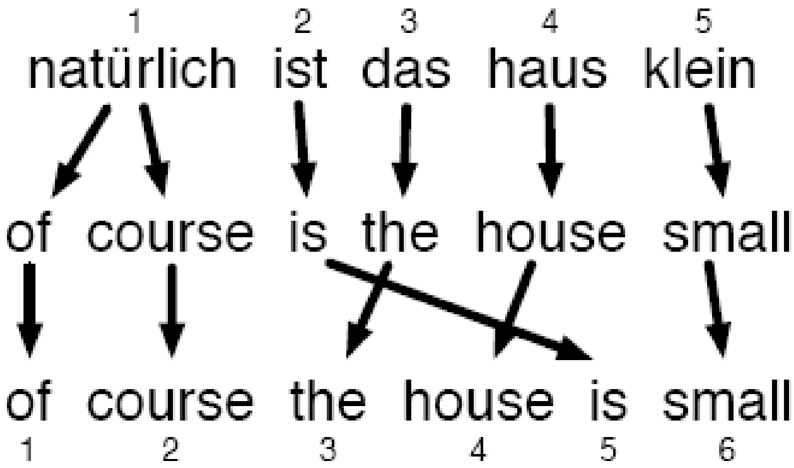
IBM Model 2

- Two-step process in IBM Model 2
  - Lexical translation step
  - Alignment step



IBM Model 2

- Two-step process in IBM Model 2
  - Lexical translation step
  - Alignment step

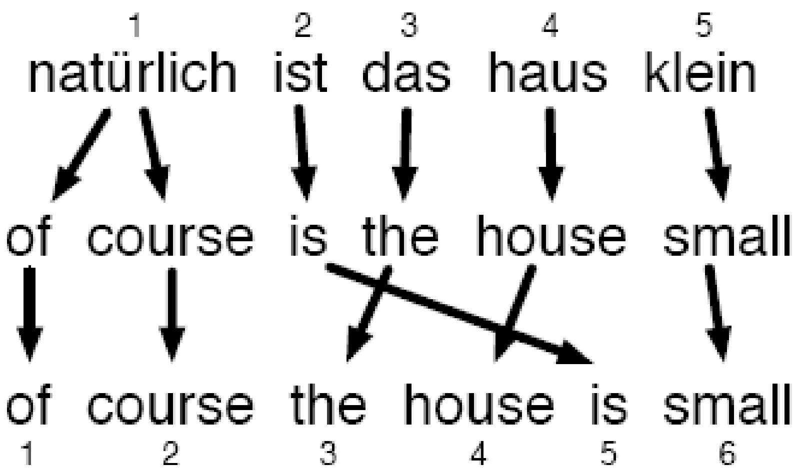


Model: $t(e|f)$

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

IBM Model 2

- Two-step process in IBM Model 2
  - Lexical translation step
  - Alignment step



Model: t(is|ist)

Model: a(2|5, 5, 6)

IBM Model 2

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

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

- At M-step, we collect fractional counts for lexical translation and alignment

IBM Model 2

- Combine the *Lexical translation step* and the *alignment step* to form IBM Model2:

$$p(\mathbf{e}, a|\mathbf{f}) = \in \prod_{j=1}^{l_{\mathrm{e}}} t\left(e_j \middle| f_{a(j)}\right) a(a(j)|j, l_{\mathbf{e}}, l_{\mathbf{f}})$$

- IBM Model 1 is a special case of Model 2 where $a(i|j, l_{\mathbf{e}}, l_{\mathbf{f}}) = \frac{1}{l_{\mathbf{f}}+1}$

IBM Model 2

- Similarly to IBM Model 1, apply the same trick (simplification) to derive $p(\mathbf{e}|\mathbf{f})$ to:

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

$$= \in \sum_{a_{1=0}}^{l_{\mathbf{f}}} \sum_{a_{2=0}}^{l_{\mathbf{f}}} \dots \sum_{a_{l_{\mathbf{e}}=0}}^{l_{\mathbf{f}}} \prod_{j=1}^{l_{\mathbf{e}}} t\left(e_j \middle| f_{a_j}\right) a(a(j)|j, l_{\mathbf{e}}, l_{\mathbf{f}})$$

$$= \in \prod_{j=1}^{l_{\mathbf{e}}} \sum_{i=0}^{l_{\mathbf{f}}} t\left(e_j | f_{a(j)}\right) a(a(j)|j, l_{\mathbf{e}}, l_{\mathbf{f}})$$

IBM Model 2

- At M-step, we collect fractional counts for lexical translation and alignment

lexical translation

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_{j=1}^{l_e}\sum_{i=0}^{l_f} \frac{t(e|f)\ a(a(j)|j,l_e,l_f)\ \delta(e,e_j)\ \delta(f,f_i)}{\sum_{i'=0}^{l_f} t(e|f_{i'})\ a(i'|j,l_e,l_f))}$$

$$c(i|j,l_e,l_f;\mathbf{e},\mathbf{f}) = \frac{t(e_j|f_i)\ a(a(j)|j,l_e,l_f)}{\sum_{i'=0}^{l_f} t(e_j|f_{i'})\ a(i'|j,l_e,l_f))}$$

alignment

IBM Model 2

- Initialization for IBM Model 2:
  - Not uniformly
  - we initialize model parameters with the estimations we get from a few iterations of Model 1 training.

IBM Model 3

- Take more information into account:
- n: Fertility parameters
  - $n(1|klitzklein)$
  - $n(2|klitzklein)$ …
  - i.e. what is the probability that “klitzklein” will produce exactly 1 or 2 English words?

IBM Model 3

- Fertility model

$$n(\varphi|f)$$

- For each foreign word f, this probability distribution indicates how many output words it usually translates to: $\varphi = 0, 1, 2, \ldots$

IBM Model 3

- We also have word-translation parameters corresponding to insertions:
  - $t(\,just \mid NULL) \ = ?$
  - i.e. what is the probability that the English word is just inserted ?

IBM Model 3

- d: Distortion parameters
  - $d(2|2)$
  - $d(3|2)$ …
  - i.e. what is the probability that the German word in position 2 of the German sentence will generate an English word that ends up in position 2/3 of an English translation?
- Enhanced distortion scheme takes into account the lengths of the German and English sentences:
  - $d(3|2,4,6)$: Same as for d(3|2), except we also specify that the given German string has 4 words and the given English string has 6 words

IBM Model 3

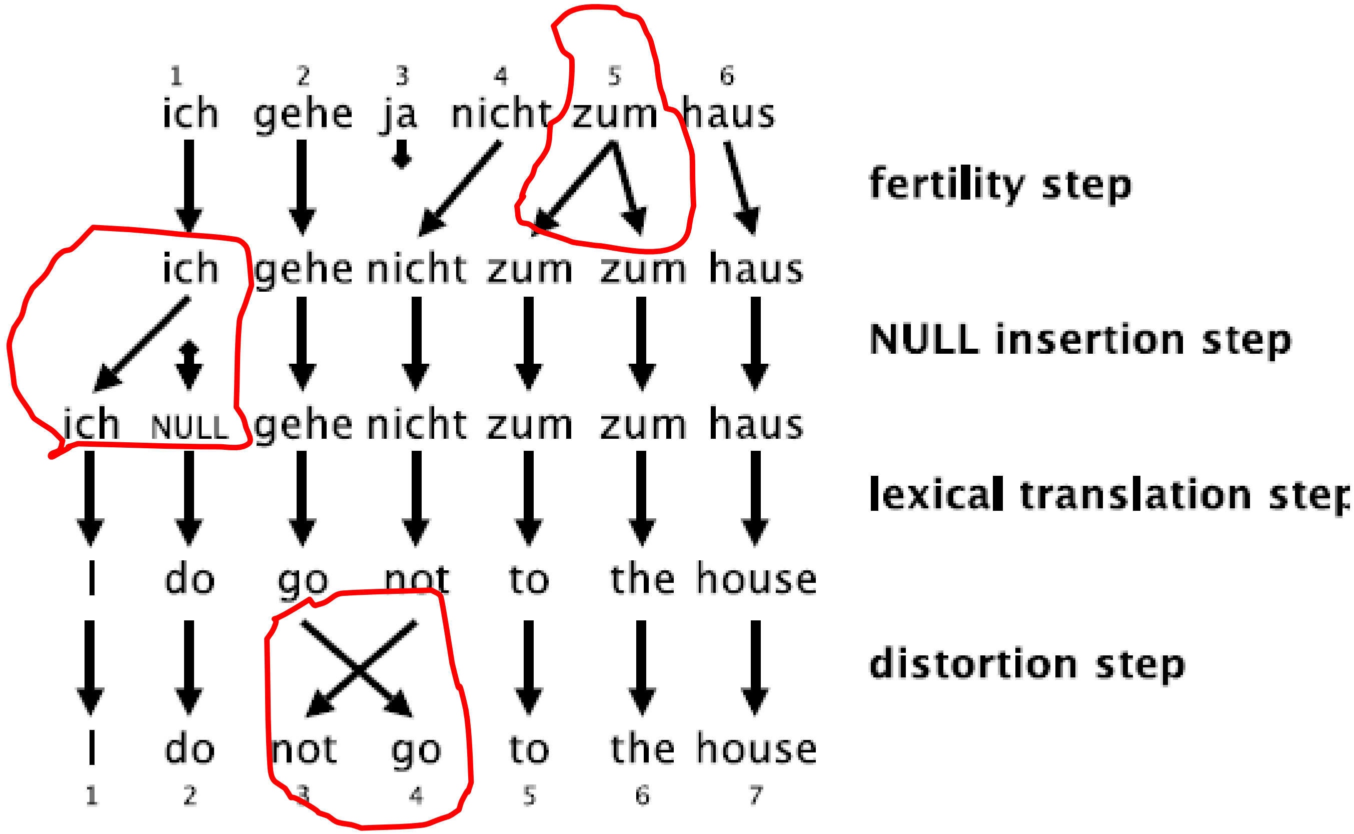
- Difference between alignment function, alignment probability distribution and distortion probability distribution:
  - Alignment function: defines for a specific alignment at which position i the foreign word is located that generated the English word j
  - Alignment probability distribution: indicates how likely an $a(i|j, l_{\mathbf{e}}, l_{\mathbf{f}})$ alignment is
  - the distortion probability distribution is set up in the translation direction predicting output word positions based on input word positions: $\mathrm{d}(j|i, l_{\mathbf{e}}, l_{\mathbf{f}})$

IBM Model 3

Type of information	**Probability Distribution Name**	**Description**
Lexical Translation	t	Table plotting source words against target words
Fertility	n	Table plotting source words against fertilities
Insertion	$p1$	Single number indicating the probability of insertion
Distortion	d	Table plotting sentence positions in source against sentence positions in target

IBM Model 3

- Four steps to form the IBM Model 3



Training of IBM Model 3

- Training IBM Model 3 with the EM algorithm
  - The trick that reduces exponential complexity does not work anymore
  - Not possible to exhaustively consider all alignments
- Finding the most probable alignment by hill climbing
  - start with initial alignment (from IBM Model 2)
  - change alignments for individual words
  - keep change if it has higher probability
  - continue until convergence
- Sampling: sampling the alignment space, collecting variations to collect statistics
  - all alignments found during hill climbing
  - neighboring alignments that differ by a move or a swap

IBM Model 4

- Reordering in IBM Model 2 and 3
  - recall: $d(j|i, l_{\mathbf{e}}, l_{\mathbf{f}})$
  - for large sentences (large $l_{\mathbf{f}}$ and $l_{\mathbf{e}}$), sparse and unreliable statistics
  - phrases tend to move together
- IBM Model 4
  - A better reordering model
  - Relative reordering model: relative to previously translated words (cepts)

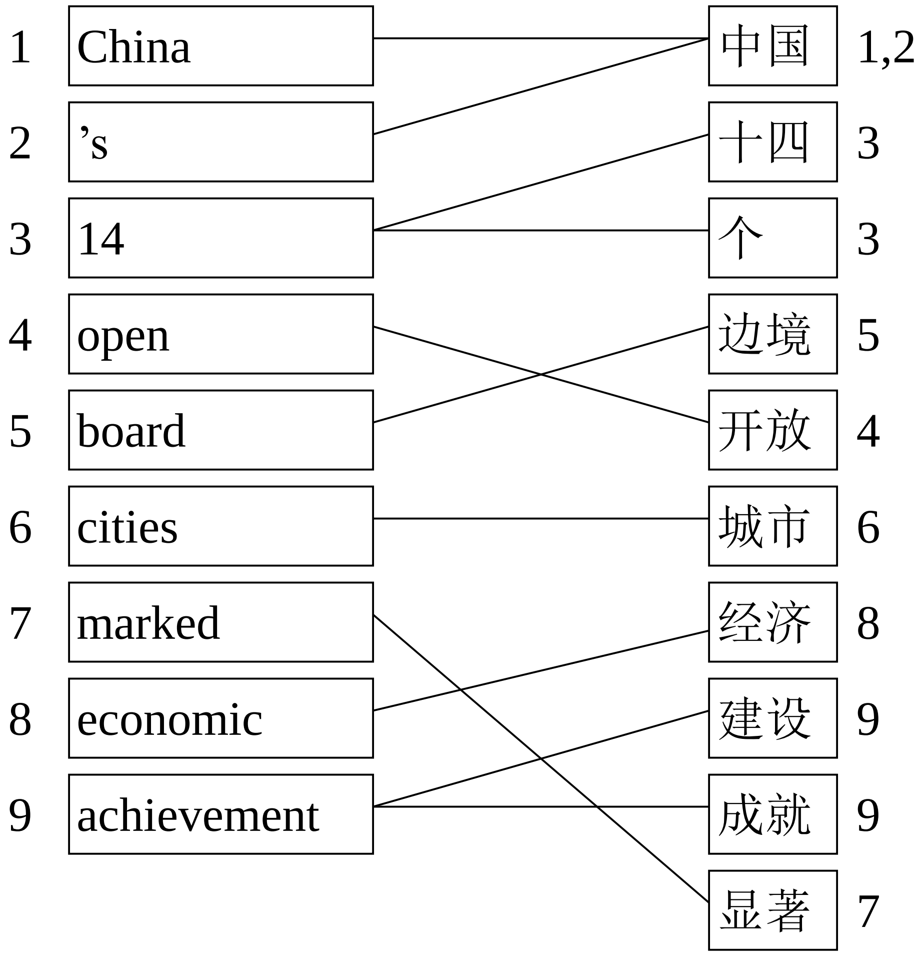
Summary of IBM Models

- IBM Model 1 - lexical translation
- IBM Model 2 - adds absolute reordering model
- IBM Model 3 - adds fertility model
- IBM Model 4 - relative reordering model

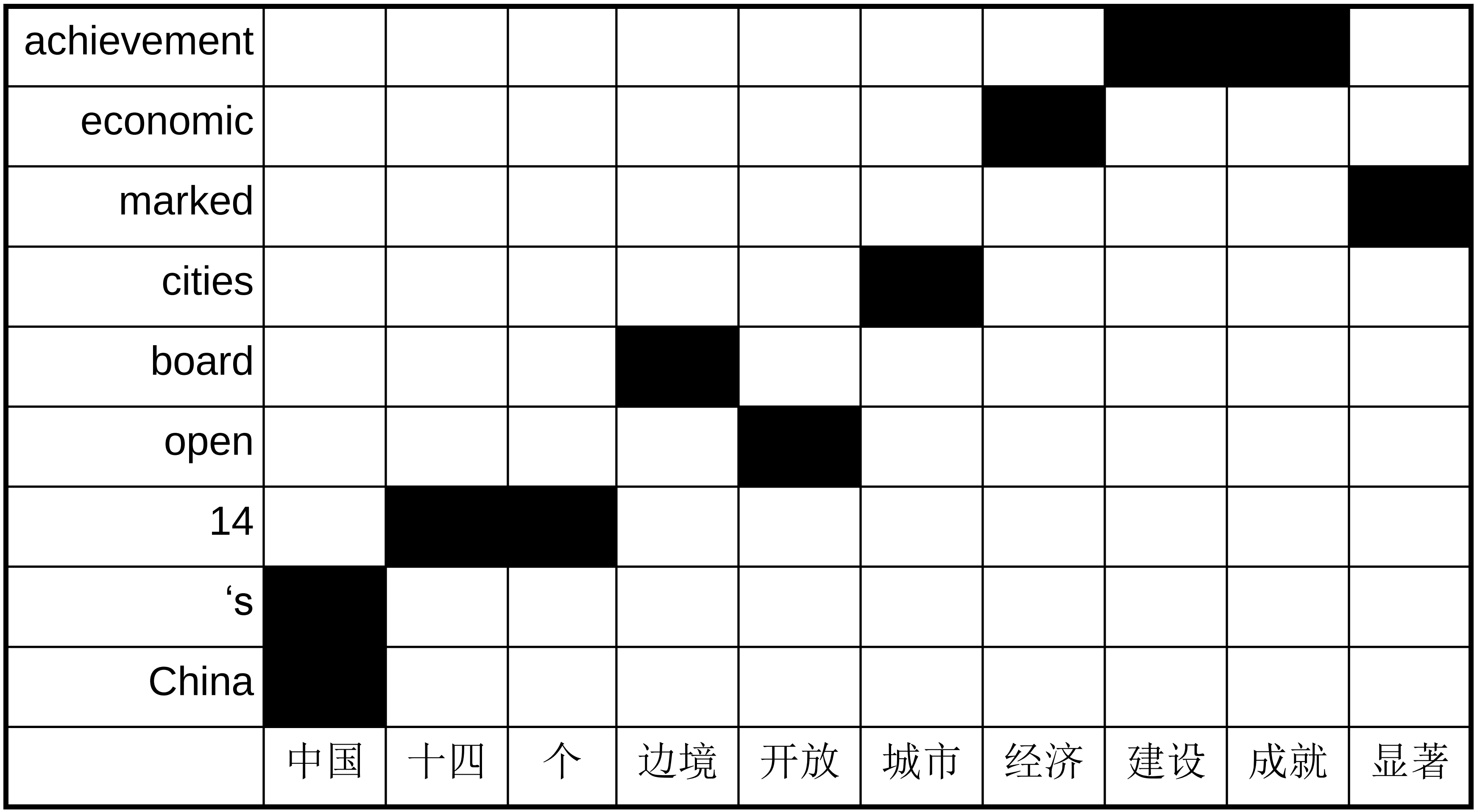
Content

- IBM Model 1
- Higher IBM Models
- **Word Alignment**
- Exercises

Word Alignment



Visualization: Word Alignment



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)

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

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

Content

- IBM Model 1
- Higher IBM Models
- **Word Alignment**
- **Exercises**

Exercise 1

Given a sentence aligned corpus

{das Haus – the house, das Buch – the book, ein Buch – a book},

calculate the lexical probabilities of

$t(the|das)$

$t(house|das)$

$t(book|das)$

$t(a|das)$

after the first iteration of EM using IBM Model 1.

Solution 1

{das Haus – the house, das Buch – the book, ein Buch – a book},

1. Initialization uniformly

Input words = { das, Haus, Buch, ein}

Output words = { the, house, book, a}

t(the|das) = 0.25

t(house|Haus) = 0.25

t(house|das) = 0.25

t(the|Haus) = 0.25

t(the|Buch) = 0.25

t(book|Buch) = 0.25

t(book|das) = 0.25

t(a|ein) = 0.25

t(book|ein) = 0.25

t(a|Buch) = 0.25

Solution 1

{das Haus – the house, das Buch – the book, ein Buch – a book},

2. Using Simplified IBM Model 1 to collect counts sentence by sentence

(***s***, ***f***): das Haus – the house

c(the|das) = 0.25/(0.25+0.25)*1 = 0.5

c(house|Haus) = 0.25/(0.25+0.25)*1=0.5

c(house|das) = 0.25/(0.25+0.25)*1=0.5

c(the|Haus) = 0.25/(0.25+0.25)*1=0.5

(***s***, ***f***): das Buch – the book

c(the|das) = 0.25/(0.25+0.25)*1 = 0.5

c(the|Buch) = 0.25/(0.25+0.25)*1=0.5

c(book|Buch) = 0.25/(0.25+0.25)*1=0.5

c(book|das) = 0.25/(0.25+0.25)*1=0.5

(***s***, ***f***): ein Buch – a book

c(book|Buch) = 0.25/(0.25+0.25)*1=0.5

t(a|ein) = 0.25/(0.25+0.25)*1=0.5

t(book|ein) = 0.25/(0.25+0.25)*1=0.5

t(a|Buch) = 0.25/(0.25+0.25)*1=0.5

Solution 1

{das Haus – the house, das Buch – the book, ein Buch – a book},

2. Using Simplified IBM Model 1 to estimate new word translation probabilities

c(the|das) = (0.5+0.5)/(0.5+0.5+0.5+0.5)=0.5

c(house|Haus) = (0.5)/(0.5+0.5)=0.5

c(house|das) = (0.5)/(0.5+0.5+0.5+0.5)=0.25

c(the|Haus) = 0.5/(0.5+0.5)=0.5

c(the|Buch) = (0.5)/(0.5+0.5+0.5+0.5)=0.25

c(book|Buch) = (0.5+0.5)/(0.5+0.5+0.5+0.5)=0.5

c(book|das) = 0.5/(0.5+0.5+0.5+0.5)=0.25

t(a|ein) = 0.5/(0.5+0.5)=0.5

t(book|ein) = 0.5/(0.5+0.5)=0.5

t(a|Buch) = 0.5/(0.5+0.5+0.5+0.5)=0.25

Exercise 2

Given the sentence pair *the house – la maison*, calculate $p(\mathbf{e}|\mathbf{f})$ for all alignments and then perform the factoring out trick.

Solution 2

the house – la maison, (ignoring the *NULL* word)

$p(\boldsymbol{e}|\boldsymbol{f})=(1/2^2)*(t(e1|f1)t(e2|f1)+t(e1|f1)t(e2|f2)+t(e1|f2)t(e2|f1)+t(e1|f2)t(e2|f2))$

$=(1/4)*(t(e1|f1)(t(e2|f1)+t(e2|f2)) + t(e1|f2)(t(e2|f1)+t(e2|f2)))$

$=(1/4)*(t(e1|f1)+t(e1|f2))*(t(e2|f1)+t(e2|f2))$

Discussion