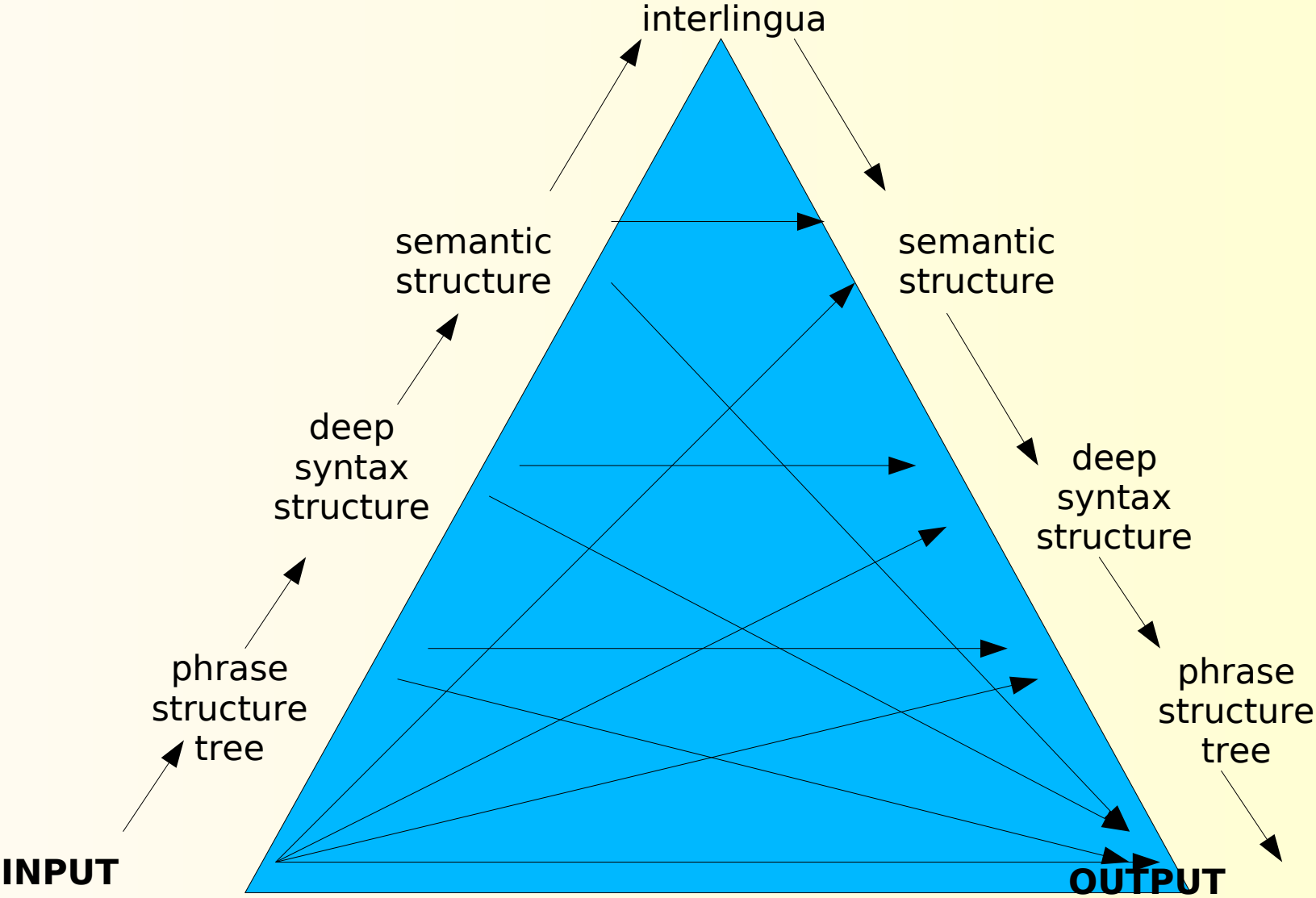


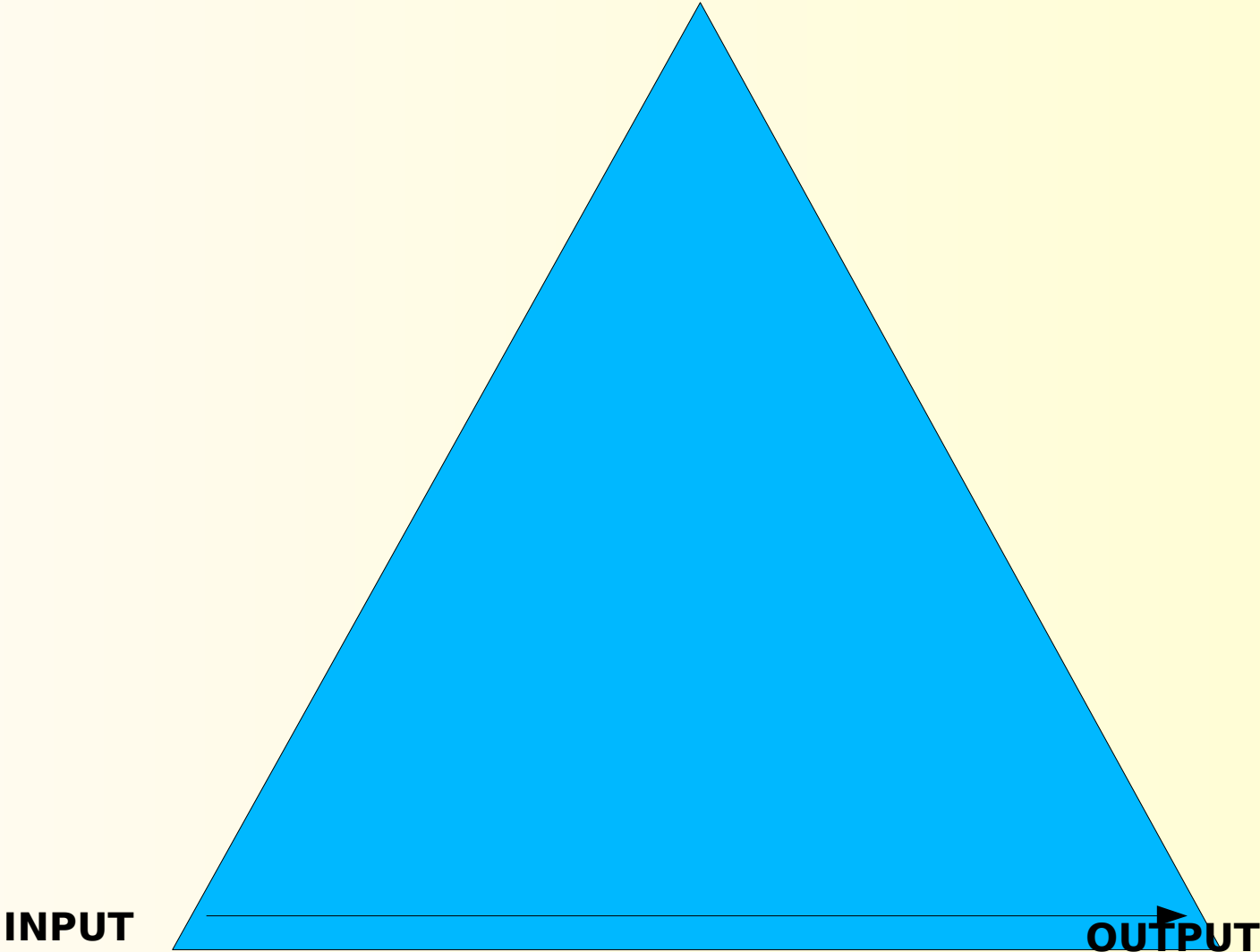
CNGL Tutorial:

Introduction to Statistical Machine Translation

Example Machine Translation Approaches



Phrase-based SMT



Statistical Machine Translation

SMT systems:

- automatically learn how to translate text from one language to another from large corpora of example translations (mostly produced by humans)
- learn how to translate during the **training phase**
- input is **unseen text** in the SL
- output is a single TL translation

The corpus of translations is known as a **bitext corpus**:

- usually sentence aligned
- also known as parallel corpus or bilingual corpus

Lexical Translation

- Simple model for machine translation based solely on **lexical translation**
- **lexical translation** is the translation of words in isolation from one another

For example, in a German-English dictionary we may find:

→ Haus – house, building, home, household, shell

- Most words have **multiple translations**
- Some **more likely** than others
- House will often be correct, whereas others are only used in particular circumstances, i.e. shell when talking about where snails live

Collecting Translation Statistics

- SMT implies the use of **statistics**
- To collect statistics for the translation of house we can
 - look at a corpus of German to English translations
 - count how often Haus is translated as each of the choices in the example, for example

| <u>Translation of Haus</u> | <u>Count</u> |
|----------------------------|--------------|
| house | 8000 |
| building | 1600 |
| home | 200 |
| household | 150 |
| shell | 50 |

- We can use these numbers to calculate a **lexical translation probability distribution**

Lexical Translation Probability Distribution

- The function p_f should return a high value if an English candidate word e is a common translation and a low value if e is a rare translation
- p_f is required to have the following two properties:

$$\sum_e p_f(e) = 1$$

$$\forall e : 0 \leq p_f(e) \leq 1$$

Maximum Likelihood Estimation

- We can use the ratio of counts to derive a probability distribution
- For example, if we have 10,000 occurrences of the word Haus in the text and 8,000 of those are translated as house, then dividing these two numbers gives us the ratio 0.8, so

$$p_{Haus}(\text{house}) = 0.8$$

- If we do this for all 5 choices we end up with the following function

$$p_f(e) = \begin{array}{ll} 0.8 & \text{if } e = \textit{house} \\ 0.16 & \text{if } e = \textit{building} \\ 0.02 & \text{if } e = \textit{home} \\ 0.015 & \text{if } e = \textit{household} \\ 0.0005 & \text{if } e = \textit{shell} \end{array}$$

- This type of estimation is called the Maximum Likelihood Estimation

Alignment

- First Model of of SMT only uses lexical translation probabilities
- The lexical translation probability, a conditional probability function $t(e|f)$, is used to denote the probability of translating a foreign word f into an English word e
- For example

| das | | Haus | | ist | | klein | |
|-------|--------|-----------|--------|--------|--------|--------|--------|
| e | t(e f) | e | t(e f) | e | t(e f) | e | t(e f) |
| the | 0.7 | house | 0.8 | is | 0.8 | small | 0.4 |
| that | 0.15 | building | 0.16 | 's | 0.16 | little | 0.4 |
| which | 0.075 | home | 0.02 | exists | 0.2 | short | 0.1 |
| who | 0.05 | household | 0.015 | has | 0.015 | minor | 0.06 |
| this | 0.025 | shell | 0.005 | are | 0.005 | pretty | 0.04 |

Alignment II

Translate word by word: *das Haus ist klein*

One possible translation: the house is small

- This translation contains an alignment, i.e. a mapping from English words to German words

| | | | |
|-----|-------|-----|-------|
| 1 | 2 | 3 | 4 |
| das | Haus | ist | klein |
| | | | |
| the | house | is | small |
| 1 | 2 | 3 | 4 |

Alignment Function

- Alignment function, a , maps each output English word at position j to a German word at position i :

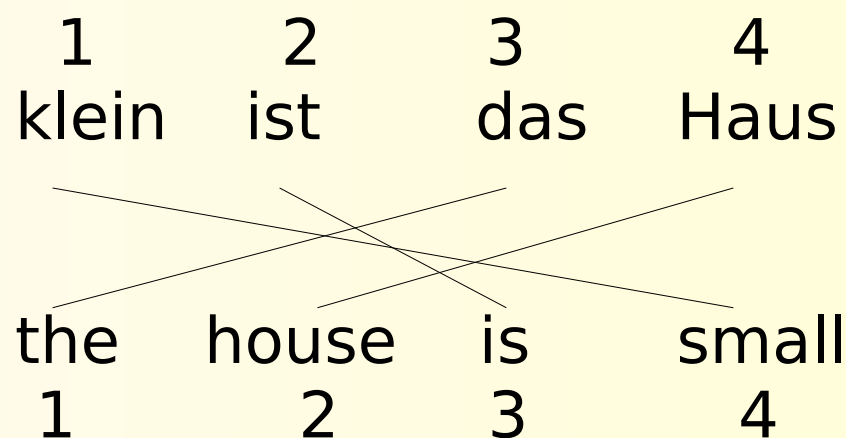
$$a: j \rightarrow i$$

In our previous example

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

Alignment Function II

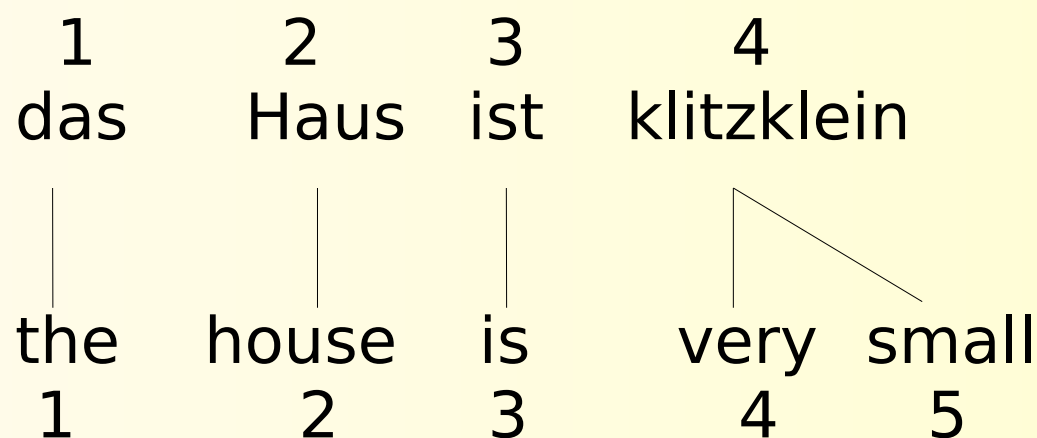
- Previous examples were very simple
- Word order was identical in German and English for the sentence pair
- Many languages have similar word order
- It is of course possible that a foreign language will have different word order to English



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

Alignment Function III

- Languages can also differ in how many words are necessary to express the same concept
- For example, one German word can be expressed by two English words



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

Alignment Model

In summary, the alignment model can allow for

- dropping of words
- adding of words

Two useful SMT Conventions:

- input is referred to as the foreign language
- output is referred to as English

IBM Model 1

- **IBM Model 1** is a model that generates a number of different translations for a sentence, each with a different probability, using lexical translation probabilities and the notion of alignment

IBM Model 1 is a **generative model** for translation

- the process of translating the sentence is broken up into smaller steps, by modeling the smaller steps with probability distributions and combining the steps into a coherent story

IBM Model 1

- For each **output** word \mathbf{e} that's produced by the model from an **input** word \mathbf{f} , we factor in the translation probability $t(\mathbf{e}|\mathbf{f})$ and nothing else
- The translation probability for a foreign sentence $\mathbf{f} = (\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_{l_f})$ of length l_f to an English sentence $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{l_e})$ of length l_e with an alignment of each English word \mathbf{e}_j to a foreign word \mathbf{f}_i according to the alignment function $\mathbf{a}: \mathbf{j} \rightarrow \mathbf{i}$ as follows:

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

IBM Model 1 ... a closer look...

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

- core part of formula is a product over the lexical translation probabilities for all generated English words e_j
- there are I_e English words
- the fraction before the product is necessary for normalisation
- since we include the special NULL token there are $I_f + 1$ input words and therefore there are $(I_f + 1)^{I_e}$ different alignments that map $I_f + 1$ into I_e output words
 - ϵ is a normalization constant

IBM Model 1 ... an example...

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

- Try to work out the probability of the translation
das Haus ist klein ||| the house is small
using IBM Model 1 and the translation table on slide 6

Learning Lexical Translation Models

- We described a model of translating sentences based on **lexical translation probability distributions**
- However, we can only do this if we **already** have a **word-aligned bitext corpus**
- We usually don't have that – a bitext corpus in general is **sentence aligned but not word aligned**
- We need a way to **automatically learn** the word alignment
- We have a method of learning the translation probability distributions from a sentence-aligned parallel test
- To do this we can use the **Expectation Maximization Algorithm**

Incomplete Data

- A common problem for machine learning is incomplete data
- The data we have is
 - sentence-aligned bitext corpus
- The data we are missing is
 - the alignment between the words of the sentences
- This is an example of the chicken-and-egg problem
 - if we knew the alignment, we could estimate the probability distribution
- We can consider the alignment as a hidden variable in our model

Expectation Maximization Algorithm

1. Initialize the model, typically with uniform distributions
2. Apply the model to the data (expectation step)
3. Learn the model from the data (maximization step)
4. Iterate steps 2 and 3 until convergence

Language Modeling: Ensuring Fluent Output

- IBM Model 1 ignores the context surrounding
- but context often matters for translation for example, when we translate the word 'klein' in German as little or small, both translations are valid
- depending on the context one is more appropriate
- for example, what sounds better in English 'small step' or 'little step'
- how can we automatically judge which is better?
- use Google?
 - 'small step': 2,070,000 occurrences in Google index
 - 'little step': 257,000 occurrences in Google index

Language Modeling

- An English language model tries to measure what is considered good English
- $p(e)$ is an estimate of the probability of an English string e
- Using Google is a neat trick for short sequences of words, but what about whole sentences?
- the longer the sentence the less likely it is to be found by Google
- we can use an old trick – generative modeling and break down the computation of the probability of long sentences into smaller steps
- we might be able to collect sufficient statistics for short sequences of words

N-gram Language Modeling

- N-gram language modeling is the most commonly used method for language modeling
- An example of an n-gram language model is a trigram language model

$$\begin{aligned} p(\mathbf{e}) &= p(e_1, e_2, e_3, \dots, e_n) \\ &= p(e_1) p(e_2 | e_1) p(e_3 | e_1, e_2) \dots p(e_n | e_1, e_2, \dots, e_{n-2}, e_{n-1}) \\ &\approx p(e_1) p(e_2 | e_1) p(e_3 | e_1, e_2) \dots p(e_n | e_{n-2}, e_{n-1}) \end{aligned}$$

- we make an assumption that only the two words that precede each word matter for predicting a word

N-gram Language Modeling

- Estimating the probabilities for a trigram language model involves the collection of statistics for three word sequences from large monolingual corpora
- When translating into English we can use the English side of the parallel corpus in addition to any other English text we want to use to train a language model

Noisy Channel Model

- In SMT we use the noisy channel model to combine the translation model and language model

- What we want to find is this:

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

- What English sentence **e** gives the highest value to **p(e|f)**, where f is the foreign sentence we want to translate

- Applying **Bayes Rule** we can use **p(e)**

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

- Since **p(f)** is a constant, we can drop it from the equation

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

Log Probabilities

- In NLP we often encode the probabilities we need to deal with in the form of **positive log probabilities**
- when we deal with probabilities, we often want to calculate the **product of lots of** individual **probabilities**
- the **result** of such arithmetic can be **a very very small number**

$0.0013 * 0.000016 * 0.0043 * \dots =$ a very small number

Log Probabilities

- if we **encode** the **probabilities** we want to deal with **as** their **log instead**, we no longer need to multiple lots of small numbers with each other

Instead of doing this:

$$0.0625 * 0.25 * 0.5 * 0.125 * \dots = 0.0009765625$$

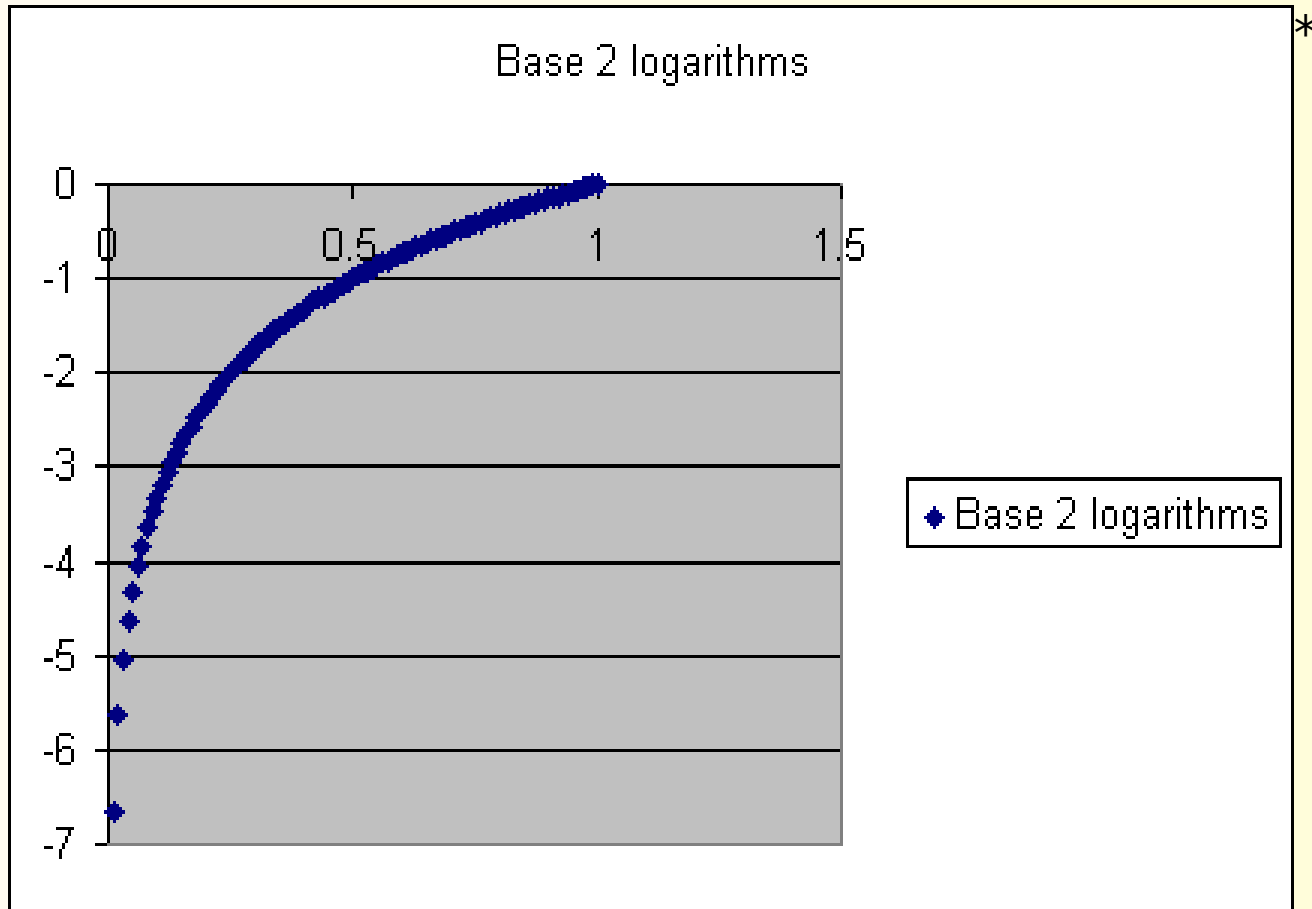
We can do this:

$$\log_2(0.0625) + \log_2(0.25) + \log_2(0.5) + \log_2(0.125) + \dots = ?$$

$$(-4) + (-2) + (-1) + (-3) + \dots = -10$$

And work on **log scale** instead

Log Probabilities



If we have

$$p(x) < p(y)$$

y has a higher probability than x

and if

$$\log_2(p(x)) < \log_2(p(y))$$

y has a higher probability than x

*diagram taken from John Goldsmith Probability for Linguists

Positive Log Probabilities

- since **all probabilities lie between 0 and 1**, **all the values** we deal with will be **negative**, so if we like we can use the positive value instead

Instead of doing this:

$$0.0625 * 0.25 * 0.5 * 0.125 * \dots = 0.0009765625$$

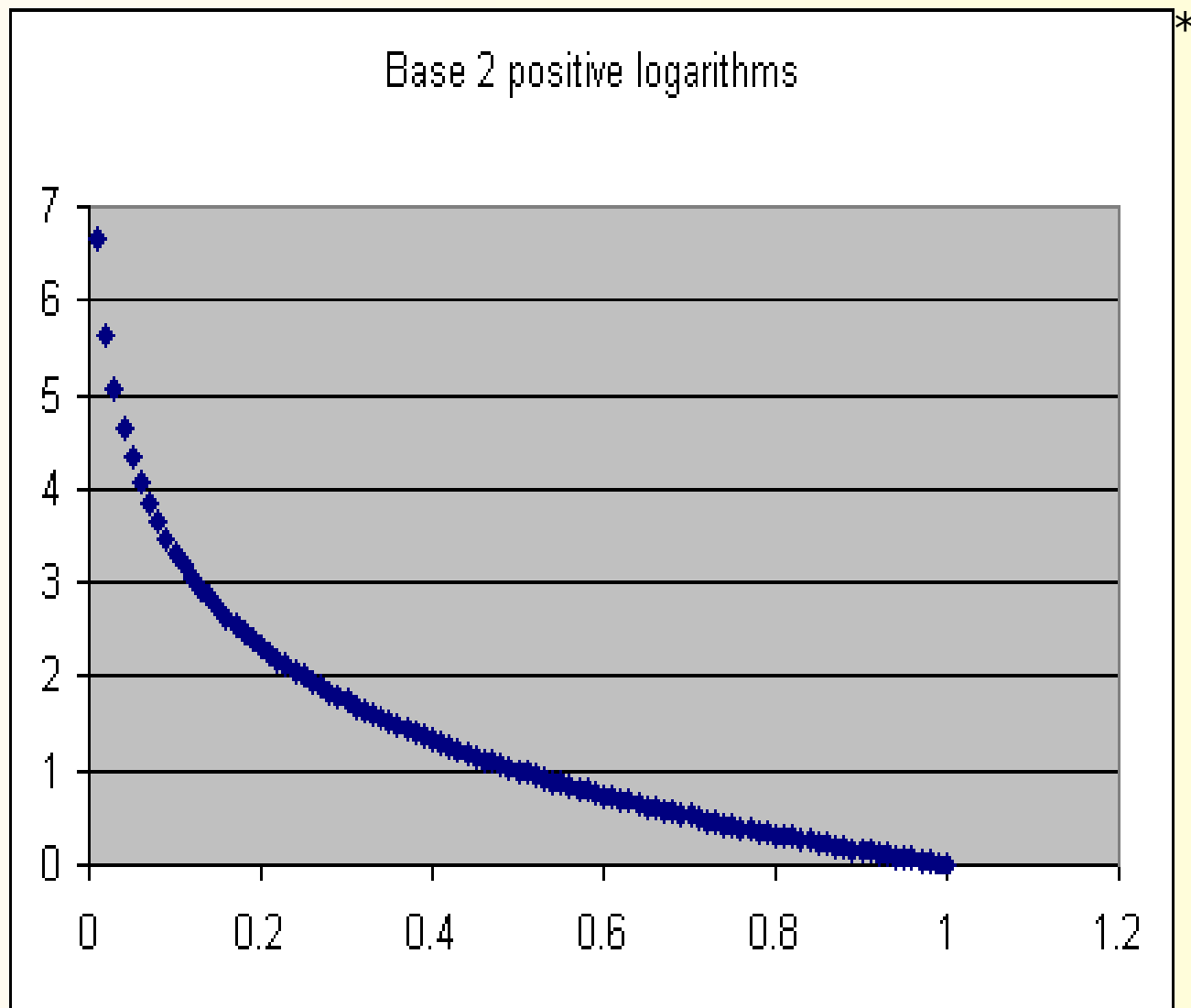
We can do this:

$$(-1 * \log_2(0.0625)) + (-1 * \log_2(0.25)) + (-1 * \log_2(0.5)) + (-1 * \log_2(0.125)) + \dots = ?$$

$$(4) + (2) + (1) + (3) + \dots = 10$$

And work on **positive** log scale instead

Positive Log Probabilities



If we have

$$p(x) < p(y)$$

y has a higher probability than x

and if

$$\text{plog}_2(p(x)) > \text{plog}_2(p(y))$$

y has a higher probability than x

Perplexity

- We have some data with **missing information** - the alignments between words of sentence pairs
- We have a model with **unknown parameters** - the lexical translation probabilities
- The **goal** of the **EM** algorithm is to **find a model that best fits the data**
- **Perplexity** can be **used to compare different probability models** to each other
- We can **use the perplexity** of the model after each iteration of the EM algorithm to **measure our progress**

Perplexity of the model

- For each model, we can calculate **the probability of the output side of the training data given the input side**
- In statistical machine translation, perplexity PP is defined as $\log_2 PP = -\sum_s \log_2 p(e_s | f_s)$
- so, for our example training corpus, the perplexity can be calculated

| | Initial | 1st iteration | 2nd iteration | 3rd iteration | ... | Final |
|--|----------------|---------------------------------|---------------------------------|---------------------------------|------------|--------------|
| p(the house 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 | ... | 0.2500 |
| p(a book ein buch) | 0.0625 | 0.1875 | 0.1907 | 0.1913 | ... | 0.1875 |
| perplexity (when $\epsilon=1$) | 4095 | 202 | 154 | 132 | ... | 114 |

- theoretically, the perplexity is guaranteed to decrease or stay the same at each iteration of EM for IBM Model 1

Higher 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

IBM Model 2

- IBM Model 2 adds an explicit model for the position of aligned words in the sentence pair
- Previously, with IBM Model 1, both the following translations would be given the same probability:

naturlich ist das haus klein
of course the house is small

naturlich ist das haus klein
the course small is of house

IBM Model 2

- In IBM Model 2, the translation of a foreign input word in position i to an English word in position j is modeled by an alignment probability distribution

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

natürlich ist das haus klein

of course is the house small

of course the house is small

1. lexical translation step

2. alignment step

IBM Model 2

IBM Model 2:

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

- EM training is done on a bilingual corpus for IBM Model 2
- counts and estimations for the distribution a is done in the same way as for the lexical translation probability t

IBM Model 3

- So far, we have allowed two words in English to translate as a single foreign word
- but we have not included this explicitly in our model
- IBM Model 3 explicitly models how many output words are produced from each input word – we call this **fertility**
- Fertility of input words is modeled with a probability distribution

$$n(\Phi|f)$$

- Φ = the number of output words
- This model indicates how many output words Φ each input word usually translates to

IBM Model 3

- For example, what do you think might be the value for Φ for these foreign words and probabilities in this example?

$$n(\Phi|\text{haus}) = 0.97$$

$$n(\Phi|\text{zum}) = 0.96$$

$$n(\Phi|\text{ja}) = 0.95$$

IBM Model 3: NULL insertion

- We use **NULL insertion** to add an English output word that does not correspond to any foreign input words
- For example, in the following sentence pair *do* has no real translation in the input German sentence

ich gehe ja nicht zum haus

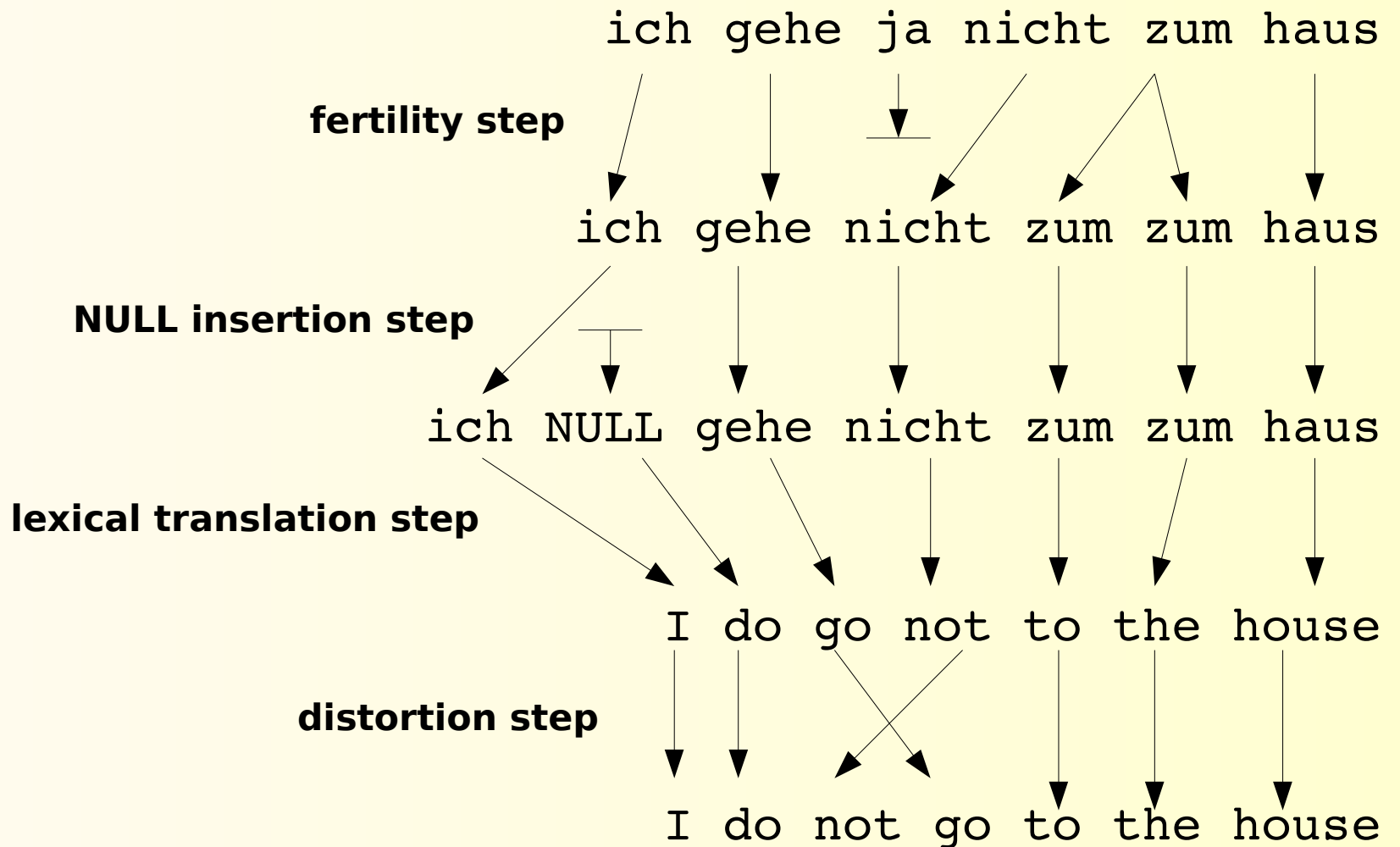
i **do** not go to the house

- if we want to add the English word *do*, it can be generated by the special NULL token

NULL ich gehe ja nicht zum haus

i **do** not go to the house

IBM Model 3



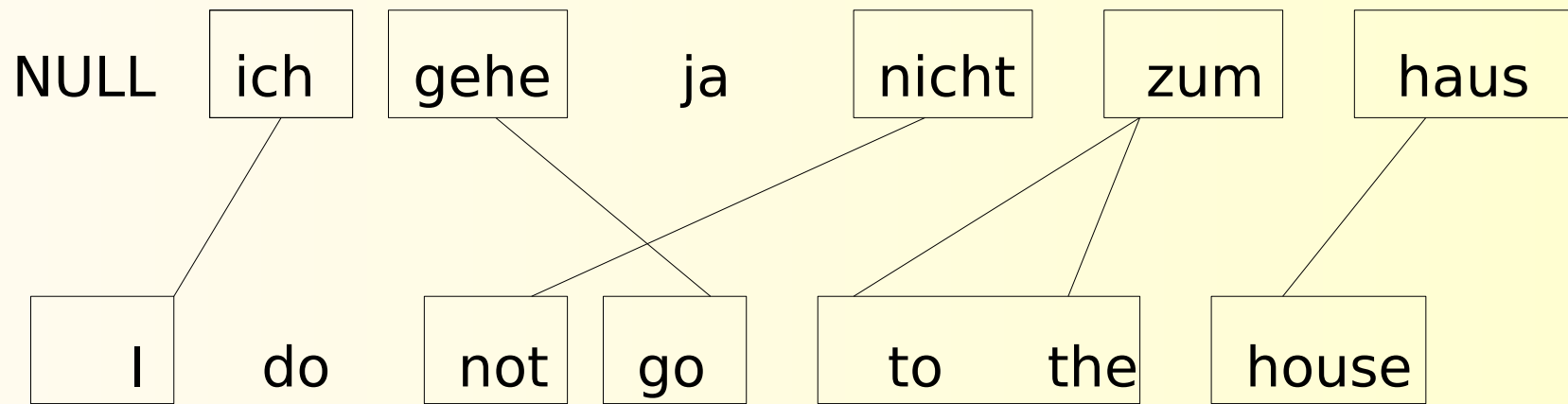
IBM Model 3 Recap

IBM Model 3 lets us do the following

- translation of words (lexical translation)
- reordering (distortion)
- insertion of words (null insertion)
- dropping words (input words with 0 fertility)
- one-to-many translation

IBM Model 4

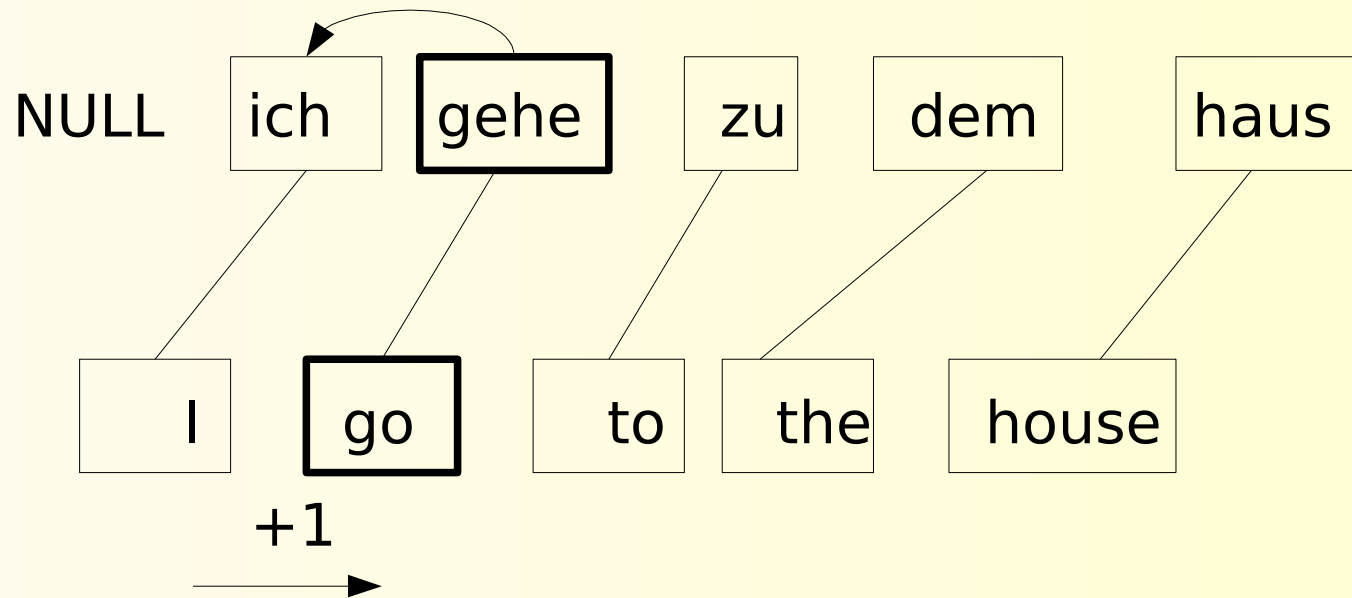
- IBM Model 4 uses the idea of a **cept** to model **relative distortion**



- likelihood of placement of English words relative to the position of the preceding cept

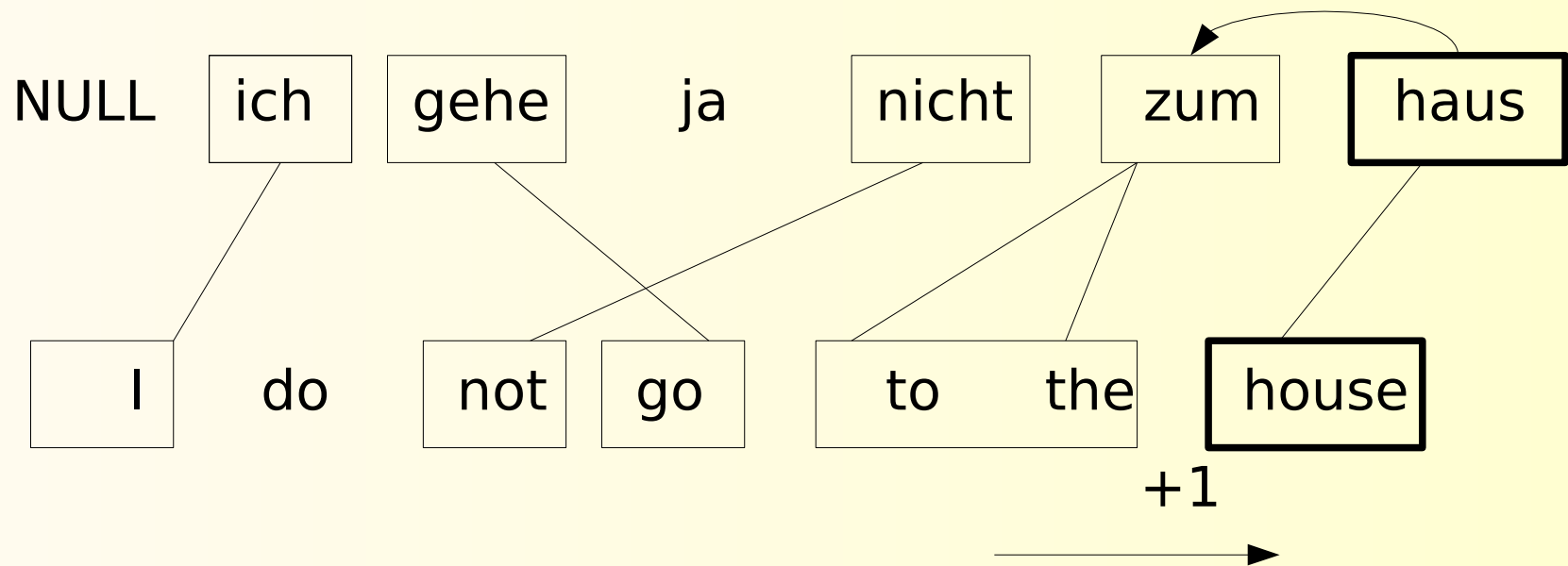
IBM Model 4

- For example, all of the English words have relative distortion of +1 in the following translation



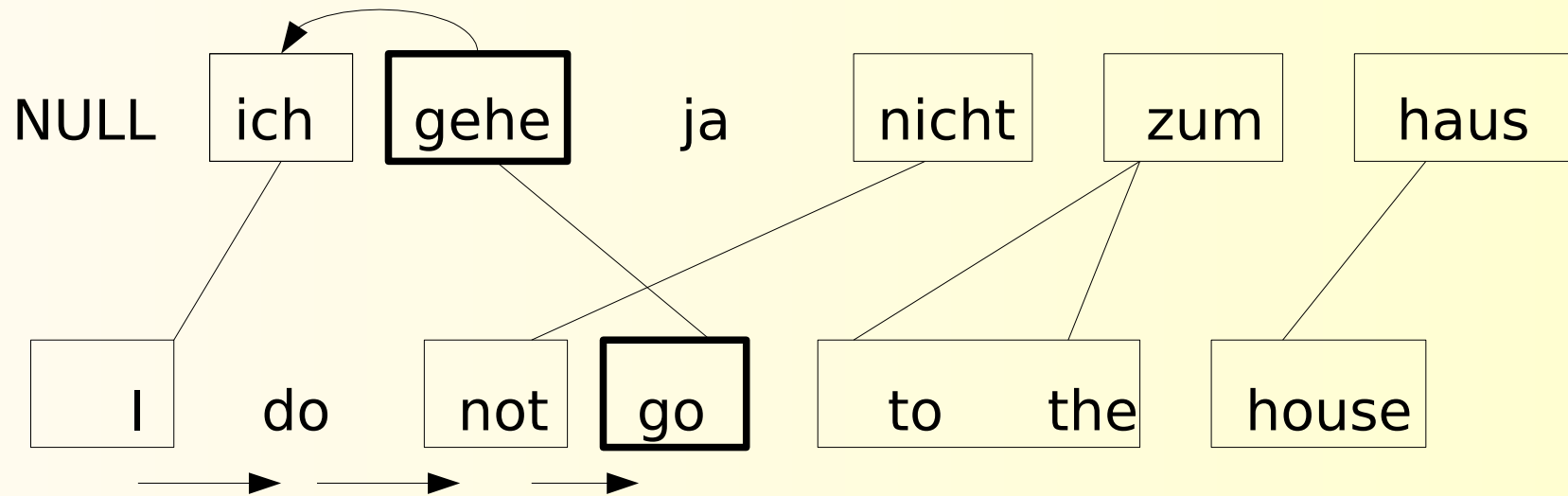
IBM Model 4 Example

- No distortion relative to the preceding cept is said to have relative distortion of **+1**



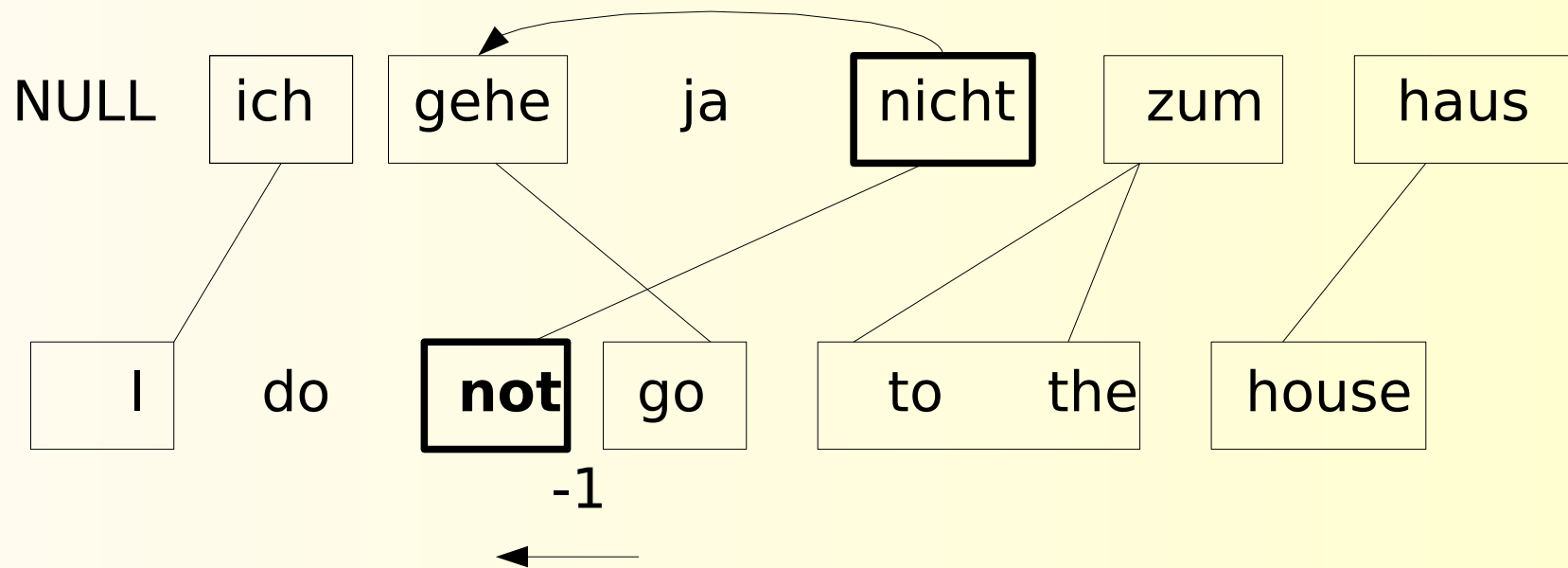
IBM Model 4

- For example, **go** has relative distortion of **+3**



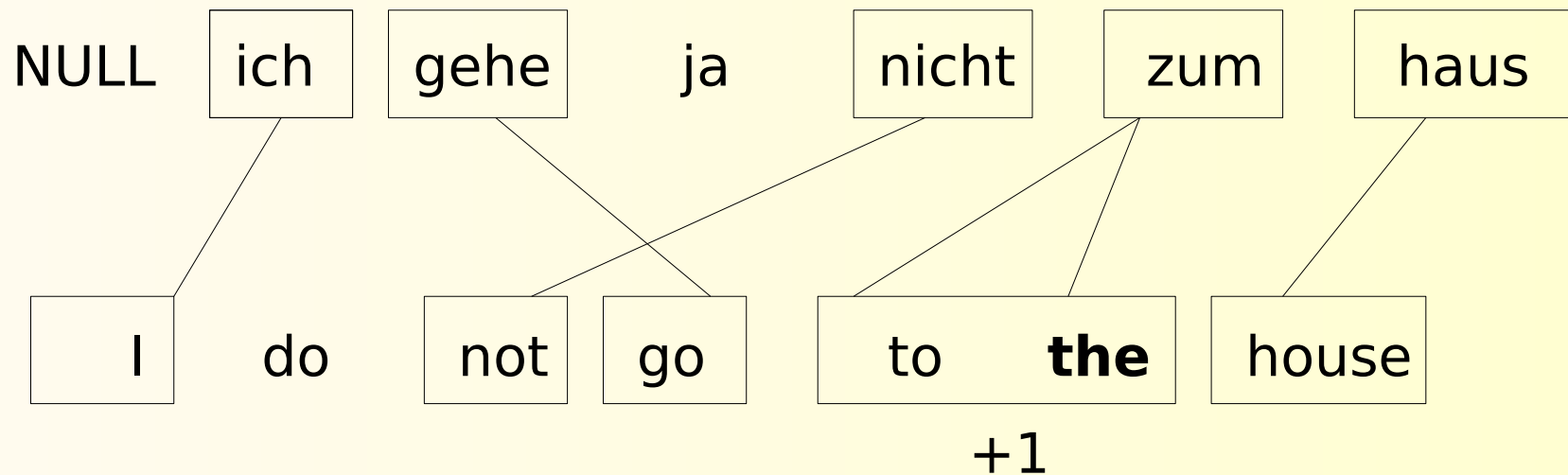
IBM Model 4 Example

- For example, the word **not** has relative distortion **-1**



IBM Model 4

- the 2nd, 3rd, 4th, ... words within a cept are treated differently



- we consider the position of the previous word in the cept

IBM Model 5


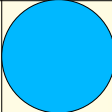
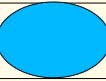

- IBM Models 3 and 4 allow multiple output words to be placed in the same position
- This is of course not possible in practice
- So, some impossible alignments have positive probabilities according to Models 3 and 4
- This wastes probability mass
- IBM Model 5 uses relative alignment similar to IBM Model 4 but eliminates the possibility of multiple output words having the same position in the sentence
- IBM Model 5 keeps track of vacant word positions and only allows placement into these positions


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

| | john | wohnt | hier | nicht |
|------|------|-------|------|-------|
| john | ● | | | |
| does | | ● | | |
| not | | | | ● |
| live | | ● | | |
| here | | | ● | |

| | john | biss | ins | gras |
|--------|------|------|-----|------|
| john | ● | | | |
| kicked | | ● | | |
| the | | | | |
| bucket | | | | ● |

Problematic Word Alignment

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

| | john | biss | ins | gras |
|--------|---|------|-----|------|
| john |  | | | |
| kicked | | ? | ? | ? |
| the | | ? | ? | ? |
| bucket | | ? | ? | ? |

Measuring Word Alignment Quality

- There are many different methods of automatically word aligning a sentence aligned bitext corpus
- The quality of a particular method can be measured for a test set of sentence pairs by comparing the alignment the method gets to the alignment a human annotator produced for the test set
- Two different types of alignments are included
 - **sure** alignment points (e.g. *wohnt* - *live*)
 - **possible** alignment points – sure alignment points and ambiguous alignment points (e.g. *wohnt* – *live* but also *biss* - *kicked*)

Alignment Error Rate

- A common metric for evaluation of word alignment methods is the alignment error rate:

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

A : alignment produced by the alignment method

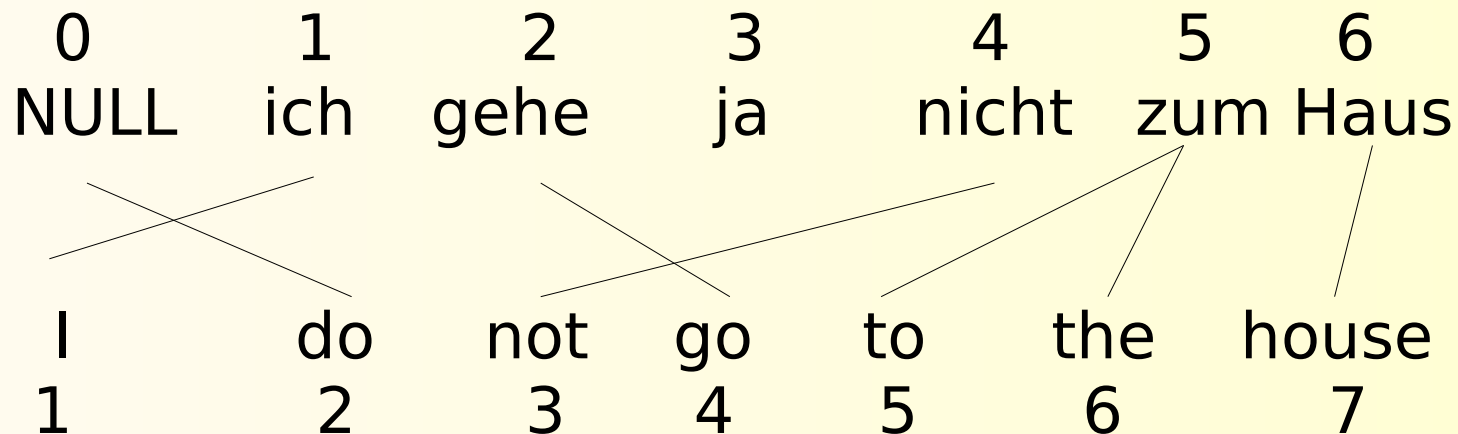
S : sure alignment points in gold standard

P : sure and possible alignment points in gold standard

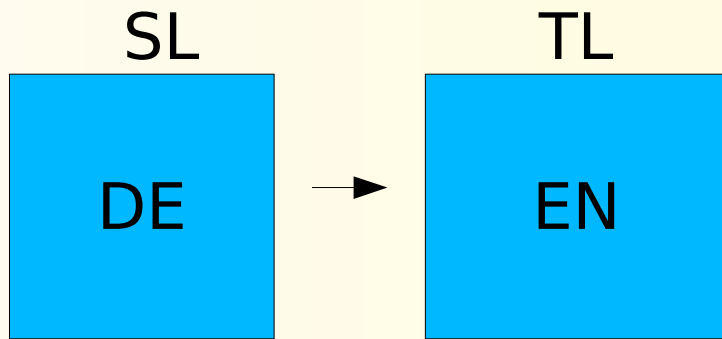
A perfect error rate of 0% can be achieved by an alignment that has all of the sure points and some of the possible alignment points

Word Alignment with IBM Models

- To use the IBM Models for word alignment we can let the model training run its course and output the Viterbi alignment of the last iteration
- The Viterbi alignment is simply the most probable alignment



Symmetrization of Word Alignment



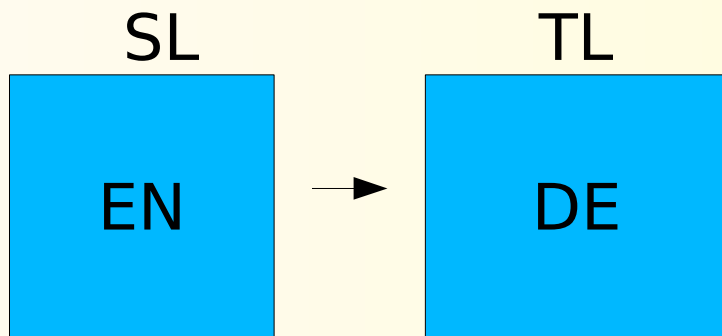
IBM Models give us

- **one DE to many EN**

We also want

- many DE to one EN

We can get this by **running word alignment in the opposite direction**, i.e. eventhough we still want to translate from DE to EN we run word alignment also as if we were translating from EN to DE to effectively get many-to-one alignments



IBM Models now give us

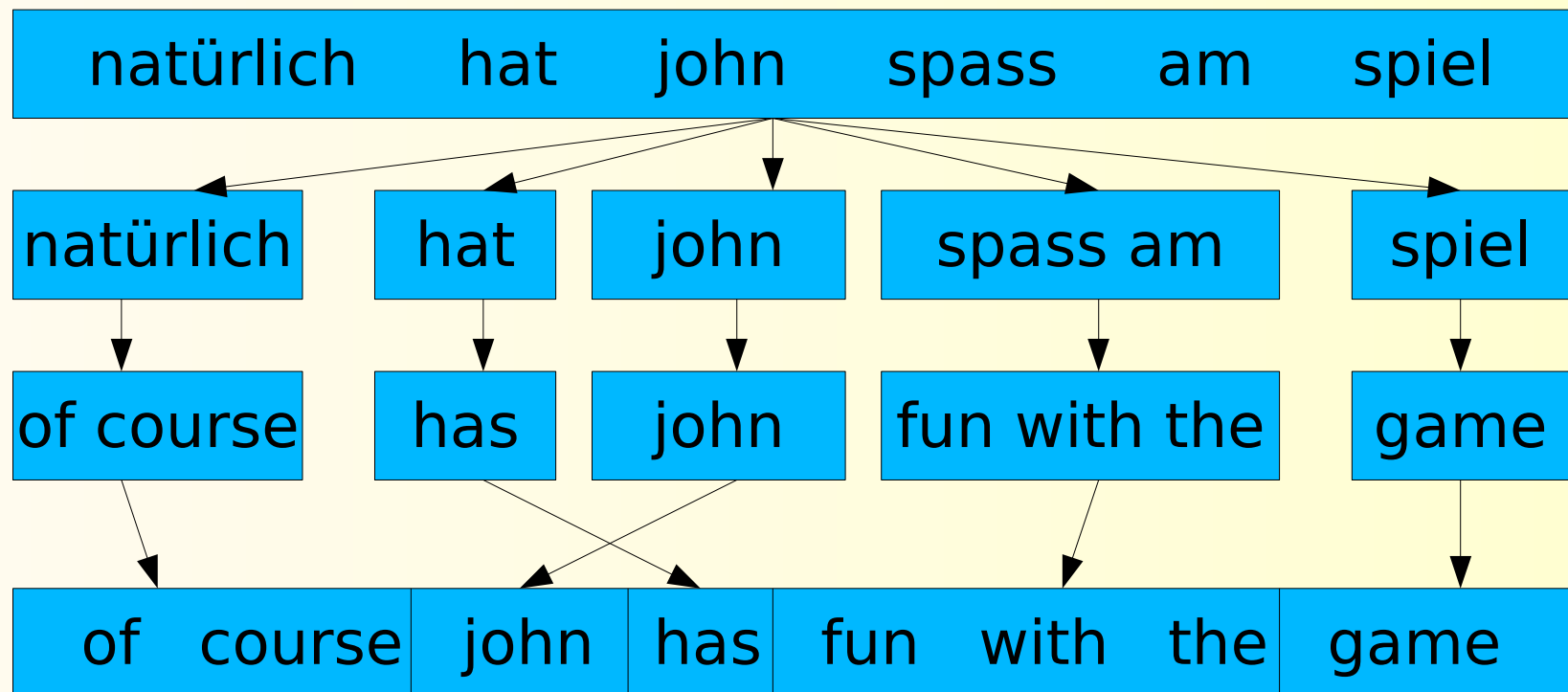
- **one EN to many DE**
- which is **many DE to one EN**

Phrase-based Models

- Using the *word* as the smallest unit of translation may not be the best option available
- Often one word in a foreign language translates as multiple words in English, and vice-versa
 - › *Hausfrau* – *house wife*
 - › *stieg um* - *changed*
- Sometimes we even have many-to-many translations between foreign and English
 - › *Grossbritannien und Nordirland* – *United Kingdom*

The Phrase

- In phrase-based models of translation, a *phrase* is used as the smallest unit for translation
- A *phrase* is simply any contiguous sequence of words – no notion of linguistic phrase used



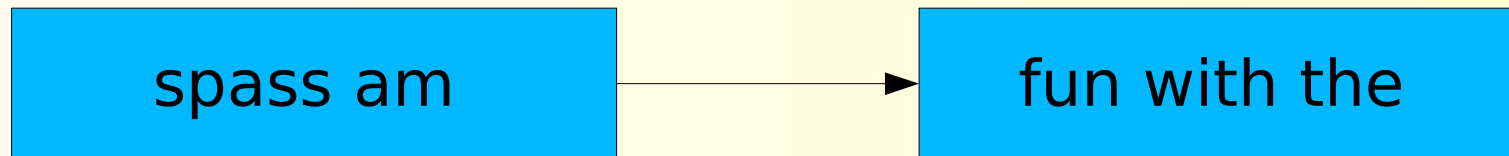
Phrase Translation Table

- The Phrase Translation Table contains the probability of translating foreign *phrases* into English
- For example, the phrase translation table entries for *natürlich*:

| Translation | Probability |
|---------------|-------------|
| of course | 0.5 |
| naturally | 0.3 |
| of course , | 0.15 |
| , of course , | 0.05 |

Resolving Ambiguity

- When translating the use of phrases, helps to resolve ambiguity

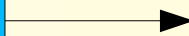


- lots of possible translations for *spass* and *am* as single words
- not so many possible translations for *spass am* as a multi-word phrase

Fluent Output

- When we have a large corpus we can learn longer and longer translation phrases

Rezessionsangst in der USA



US fear of recession

- this helps to produce fluent output since we know the English side of the phrase is a fluent sequence of words – it appeared in our training corpus
- its even possible to learn phrases that translate entire sentences

Fertility, Insertion and Deletion?

- Fertility, insertion and deletion can be handled within phrases
- Instead of allowing arbitrary words to be deleted or inserted we capture the surrounding context in a phrase

john ist ja nett

john is nice

ist ja nett

is nice

Symmetrization

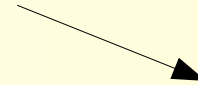
- If we run alignment in just the foreign to English direction (f2e) we won't get any many-to-one alignment points
- We can run automatic word alignment also for English to foreign (e2f) and this will allow us to get many-to-one for the f2e direction
- We use a symmetrization algorithm to decide which alignment points out of the union of the two sets of alignment points we should use e.g. the GROW-DIAG-FINAL algorithm

Symmetrization

- A symmetrization algorithm takes the bidirectional word alignment and decides which alignment points to keep and which to throw away before we extract phrases

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

Bidirectional Alignment



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

Symmetrized Alignment

Consistent Phrases

Phrase pair (f,e) is consistent with an alignment A , if all words f_1, \dots, f_n in f that have alignment points in A have these with words e_1, \dots, e_f in e and vice versa:

(f,e) consistent with alignment $A \Leftrightarrow$

$$\forall e_i \in e : (e_i, f_j) \in A \Rightarrow f_j \in f \text{ AND}$$

$$\forall f_j \in f : (e_i, f_j) \in A \Rightarrow e_i \in e \text{ AND}$$

$$\exists e_i \in e, f_j \in f : (e_i, f_j) \in A$$



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

michael – michael

michael assumes –
michael geht davon aus

michael assumes -
michael geht davon aus ,

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 – geht davon
aus

assumes – geht davon
aus ,

assumes that – geht
davon aus , dass

assumes that he – geht
davon aus , dass er

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



michael geht davon aus , dass er im haus bleibt

that - , dass

michael
assumes
that
he
will
stay
in
the
house

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ■ | | | | | | | | | |
| | ■ | ■ | ■ | ■ | | | | | |
| | | | | | ■ | | | | |
| | | | | | | ■ | | | |
| | | | | | | | | | ■ |
| | | | | | | | | | ■ |
| | | | | | | | ■ | | |
| | | | | | | | ■ | | |
| | | | | | | | | ■ | |

that he - , dass er

that he will stay in the house - ,
dass er mi haus bleibt

that - dass

that he - dass er

that he will stay in the house -
dass er im haus bleibt

he - er

he will stay in the house -
er im haus bleibt

will stay - bleibt

will stay in the house - im
haus bleibt

in the - im

in the house - im haus

house - haus

All Consistent Phrases

- We extract *all* consistent phrases since
- Short phrases are needed to achieve high coverage of unseen SL sentences
- Longer phrases will provide more fluent TL output since we know that the words of the TL side of a phrase appeared in the training data in that order

Decoding

- **Decoding** refers to the process of taking in a foreign sentence as input and constructing and searching through possible translations for that input
- The search space is too large to carry out an exhaustive search of all possible translations
- **Heuristic search** used to manage the large search space
- **Pruning** is the abandonment of some lower probability hypotheses during the search
- **Dynamic Programming** is used to combine duplicate paths to the same hypothesis
- A **Reordering Limit** is used to limit the computational complexity that arises from allowing any reordering between words/phrases

Decoding

- Translations are constructed from left to right beginning with the first word/phrase in the translation
- This is done to allow the language model scores to be computed for translations as they are constructed
- We need to give a score each partial hypothesis as each time a word/phrase is added to it
- However, the input words are permitted to be translated in any order

Input = “klein ist das haus”

Hypothesis 1: small is

Hypothesis 2: the house

Hypothesis 3: is the

Translation Options

| | | | | | | | | |
|-------|----|------|-----|----------|---|----|-------|-------|
| Maria | no | daba | una | bofetada | a | la | bruja | verde |
|-------|----|------|-----|----------|---|----|-------|-------|

| | | | | | | | | |
|---------------------|------------|-------------|---------------|-------------|-----------|---------------|--------------------|--------------|
| <u>Mary</u> | <u>not</u> | <u>give</u> | <u>a</u> | <u>slap</u> | <u>to</u> | <u>the</u> | <u>witch</u> | <u>green</u> |
| <u>did not</u> | | | <u>a slap</u> | | <u>by</u> | | <u>green witch</u> | |
| <u>no</u> | | | <u>slap</u> | | | <u>to the</u> | | |
| <u>did not give</u> | | | | | | <u>to</u> | | |
| | | | | | | <u>the</u> | | |
| | | | <u>slap</u> | | | | <u>the witch</u> | |

Model

- **State-of-the-art** systems use a log-linear model to combine different models that each help to rank good quality translations higher than lower quality ones

$$p(e|f) = \exp \sum_{i=1}^n \lambda_i h_i(e, f)$$

- **Translation model** – computed from relative frequencies of phrases extracted from the training corpus
- **Language Model** Probability
- **Word Penalty** – counter-balance the bias of the LM toward shorter output

More Features

- **Lexical Weighting** – richer model used a back-off from the translation model that uses the lexical translation probabilities of pairs of aligned words
- **Phrase Penalty** – counter-balance the bias of the translation model toward shorter phrases
- **Future cost estimation** – takes into account the difficulty of translating the rest of the sentence, to avoid eliminating good solutions
- **Reordering Model**

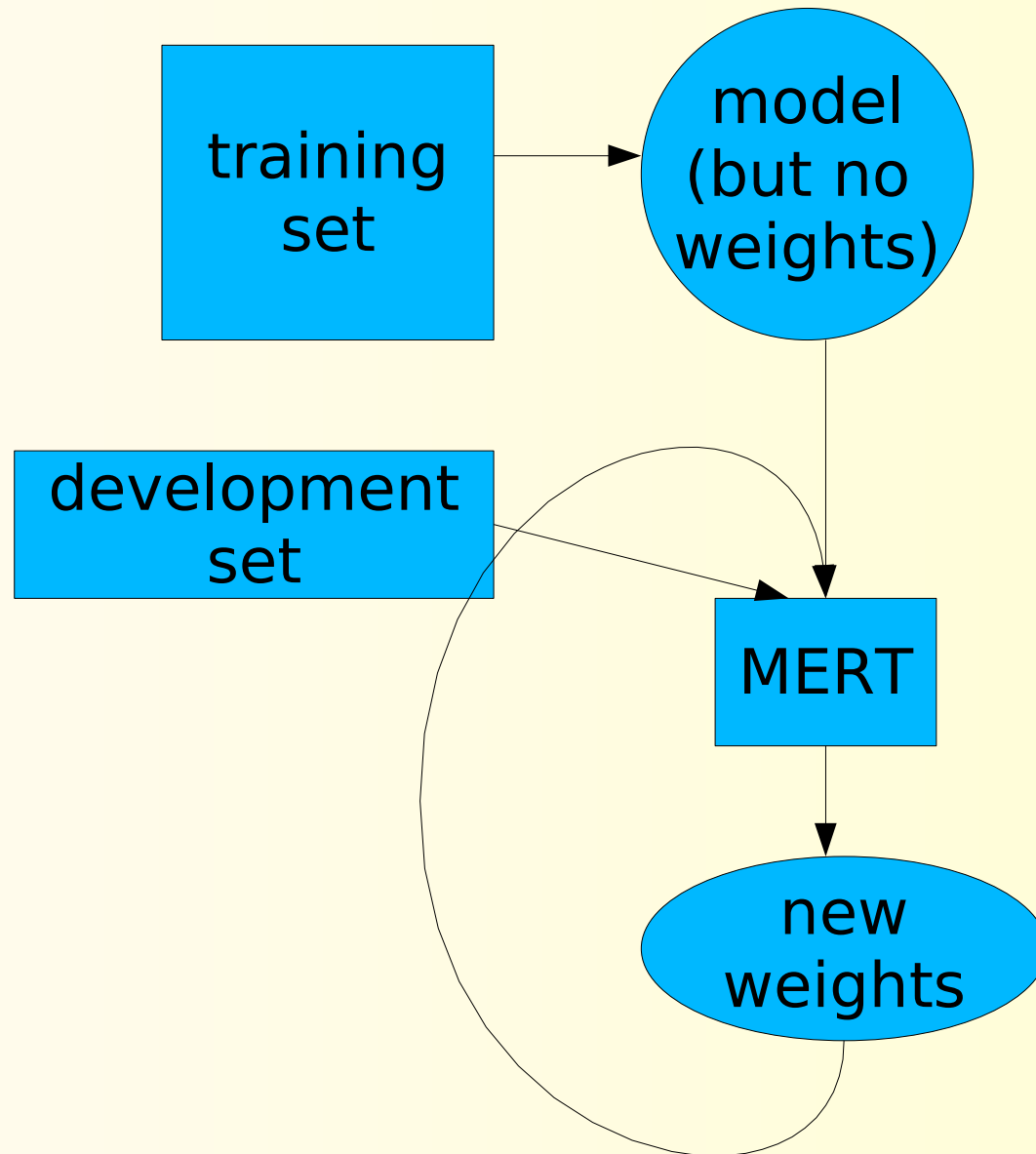
Reordering

- Reordering of words is explicitly modeled using a lexicalized reordering model
- Three types of reordering are modeled
 - monotone
 - swap
 - discontinuous

For phrase pair (f,e):

$$p_o(\textit{orientation}|f,e) = \frac{\textit{count}(\textit{orientation},f,e)}{\sum_o \textit{count}(\textit{orientation},f,e)}$$

Training an SMT system



Minimum Error Rate Training

- MERT is used to obtain a suitable set of weights to use to for log-linear combination of feature scores during translation
- We begin with some default set of weights
- Run the machine translation system on a development set of sentences (e.g. 1000 or 2000 sentences) for which a reference translation is available
- Get the n-best output for each translation (e.g. the top 100 translations for each input sentence) along with the scores assigned by each feature in the model
- Use an automatic evaluation metric (e.g. Bleu) as an objective function to score the translations

MERT

- Use an algorithm to get the best set of weights we can find using the n-best translations and their scores and the comparison with the reference translation
- Use the new set of weights and run the system on the development set again
- Add the new n-best translations to the initial set and find a new better set of weights
- Do this until convergence (e.g. no new translations are found)
- These weights are then used on the unseen test sentences

MERT

Development Set

klein ist das haus
john ist ja nett
...

the house is small
john is nice
...

N-best output TM LM LT ...

klein ist das haus

john ist ja nett
...

| | | | | |
|--------------------|-----|------|------|-----|
| small is the house | 3.5 | 23.2 | 29.1 | ... |
| the house small is | 2.3 | 25.6 | 33.2 | ... |
| the house is small | 1.4 | 28.9 | 30.0 | ... |
| ... | ... | | | |
| john is kind | | | | |
| ... | | | | |

Find better weights using evaluation metric (e.g. BLEU)

klein ist das haus

john ist ja nett
...

the house is small

john is nice
...

| | | |
|--------------------|---|-----|
| small is the house | $\lambda_1 3.5 + \lambda_2 23.2 + \lambda_3 29.1$ | ... |
| the house small is | $\lambda_1 2.3 + \lambda_2 25.6 + \lambda_3 33.2$ | ... |
| the house is small | $\lambda_1 1.4 + \lambda_2 28.9 + \lambda_3 30.0$ | ... |
| ... | | |
| john is kind | | |
| ... | | |

Summary

- IBM Models with Expectation Maximization can be used for automatically word alignment of the bitext corpus
- Both language directions are necessary to include many-to-one types of alignment
- Word alignment is symmetrized and phrases are extracted and counted to compute the translation model
- A log-linear model is used to combine this translation model and other features
- A language model is included to produce fluent output as well as several other features
- An exhaustive search of all possible translations is (far) too expensive

Summary

- A heuristic search is carried out that allows lower scoring hypothesis to be pruned/dropped during construction of translations
- Minimum error rate training is carried out on a development set with reference translations to obtain weights for translating unseen test data