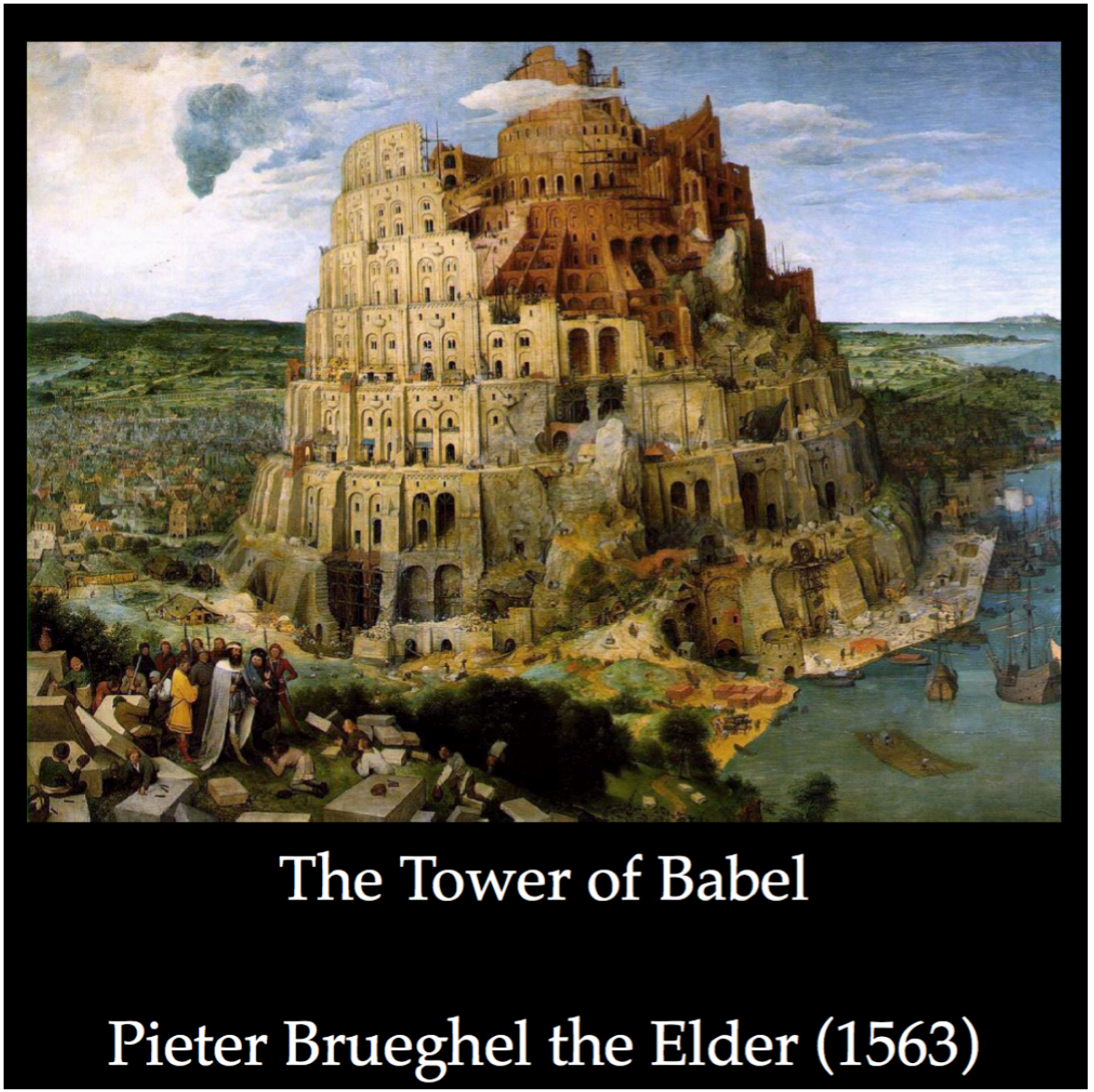
Statistical Machine Translation

What is machine translation?

نائب امریکی صدر ڈک چینی کا کہنا ہے کہ میں اسامہ بن لادن
کو زندہ یا مردہ دیکھنا چاہتا ہوں۔

American Vice President Dick Cheney has said that he wants to see Osama bin Laden dead or alive.

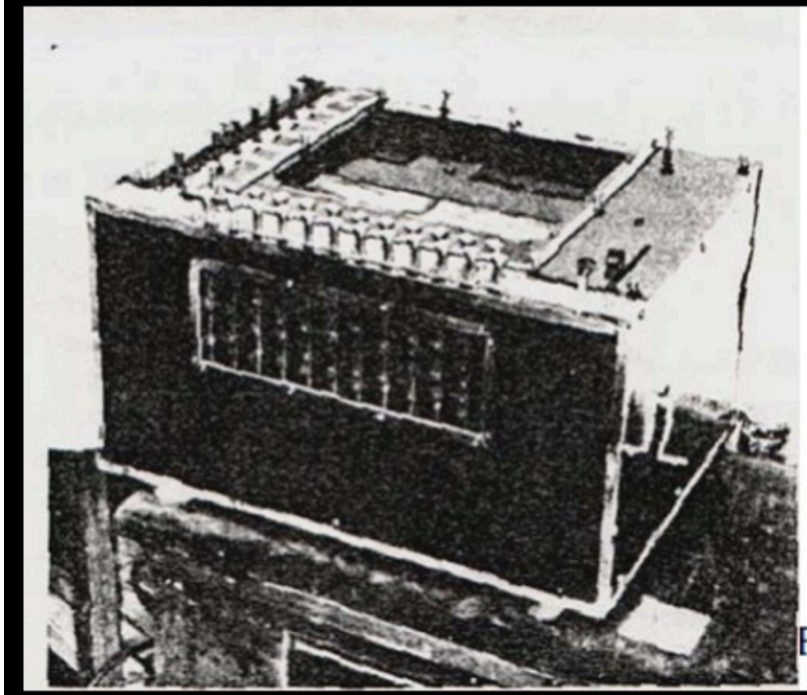
A brief history



The Tower of Babel

Pieter Brueghel the Elder (1563)

A brief history



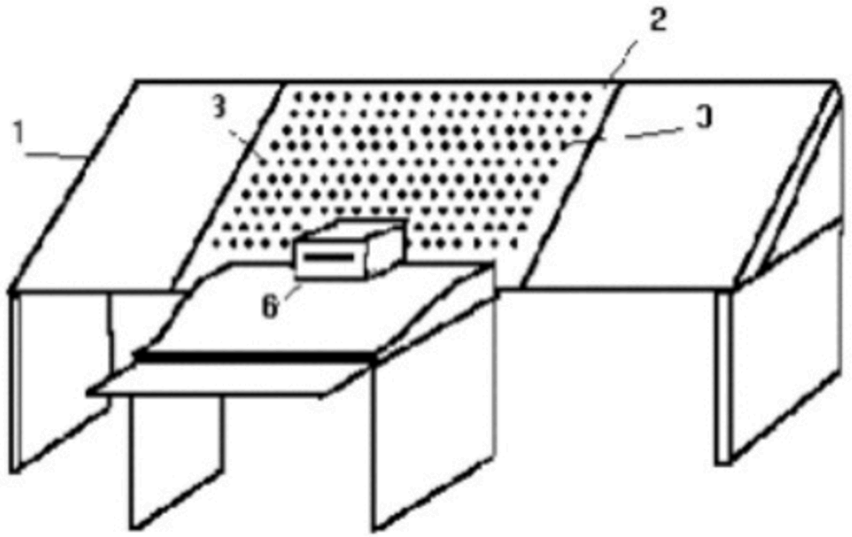


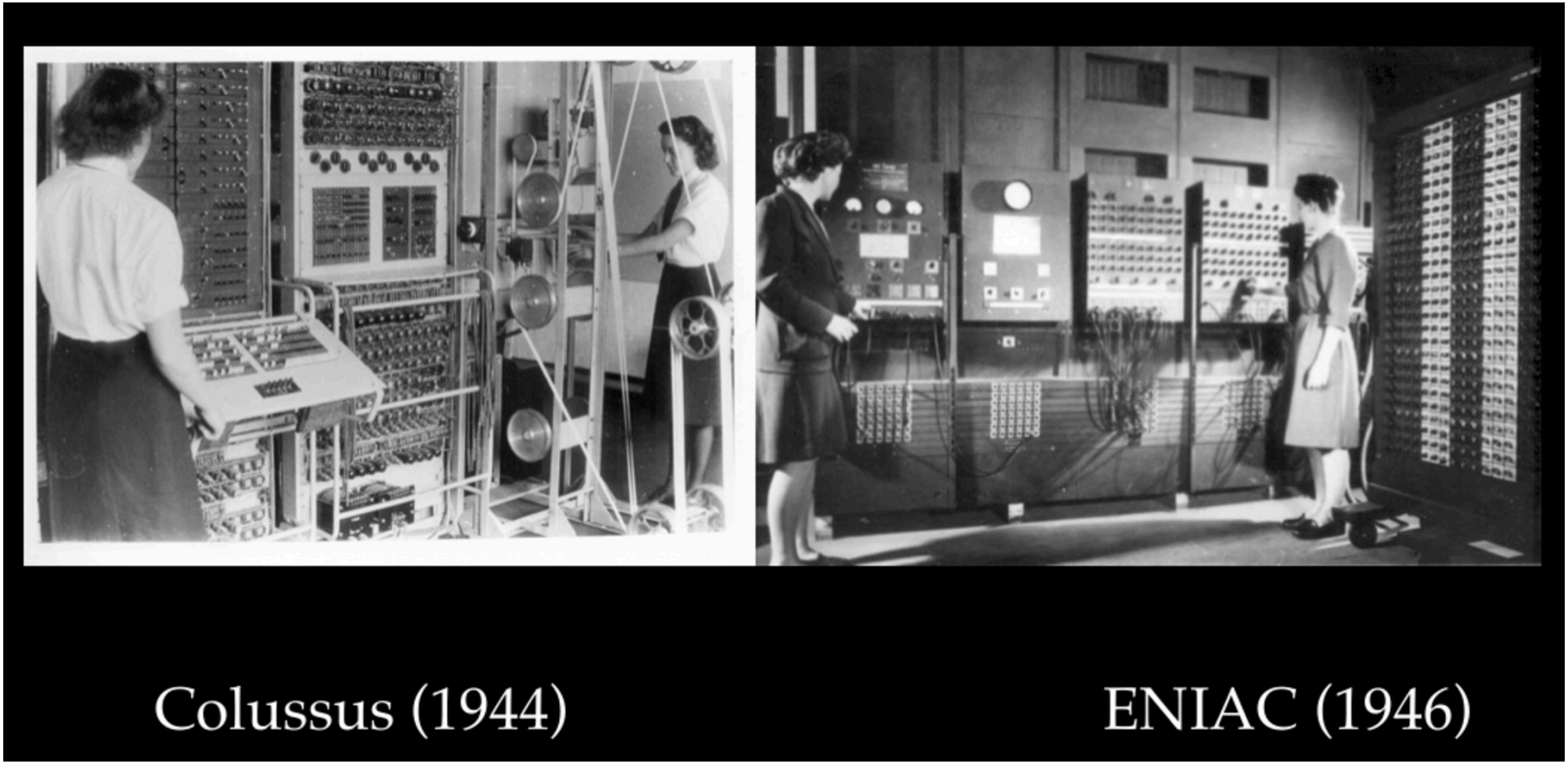
Fig. 3. Trojanskij's translating machine (*from* Bel'skaja et al. 1959)

mechanical brain
(Artsrouni, France 1933)

(Trojanskij, Russia 1933)

A brief history

A brief history



A brief history

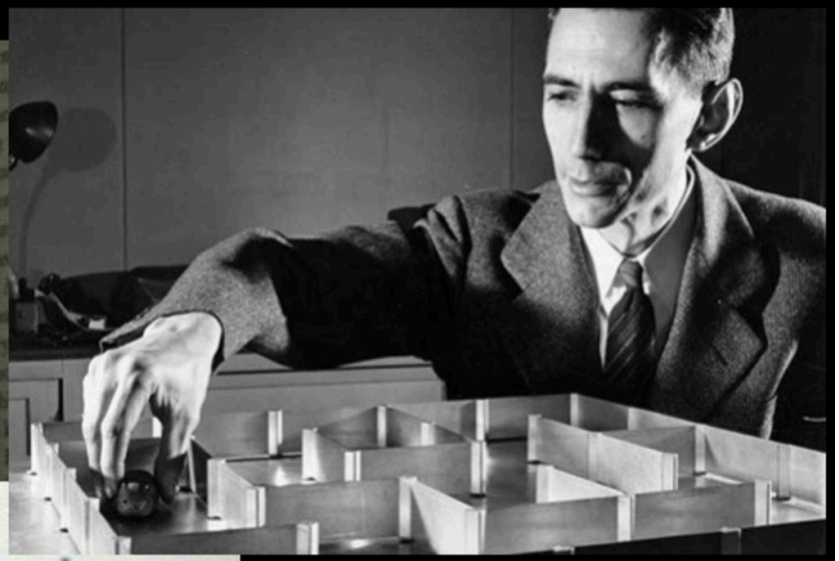
1948

Communication Theory of Secrecy Systems*

By C. E. SHANNON

1. Introduction and Summary

THE problems of cryptography and secrecy systems furnish an interesting application of communication theory.[1] In this paper a theory of secrecy systems is developed. The approach is on a theoretical level and is intended to complement the treatment found in standard works on cryptography.[2] There, a detailed study is made of the many standard types of codes and ciphers, and of the ways of breaking them. We will be more concerned with the general mathematical structure and properties of secrecy systems.



The Bell System Technical Journal

Vol. XXVII July, 1948 No. 3

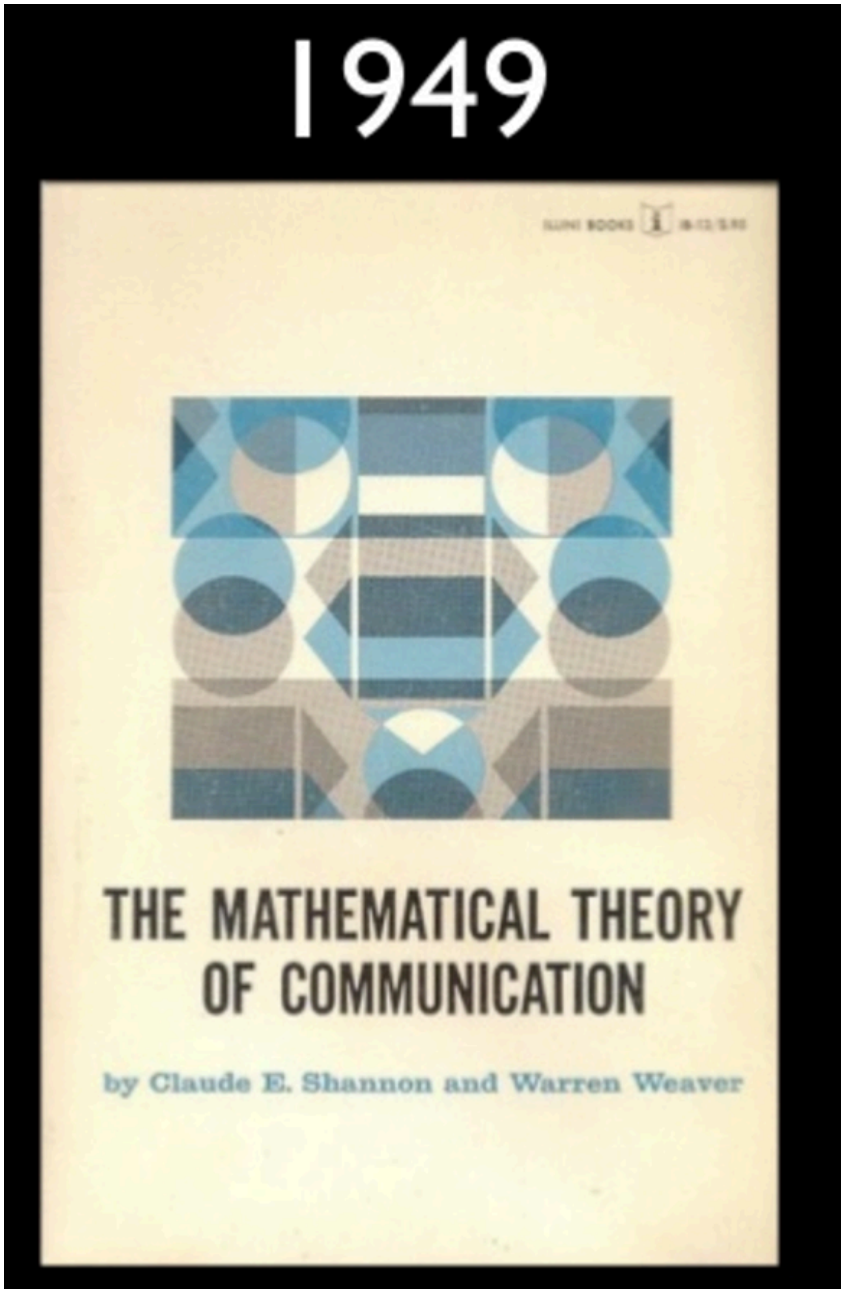
A Mathematical Theory of Communication

By C. E. SHANNON

Introduction

THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has in-

A brief history

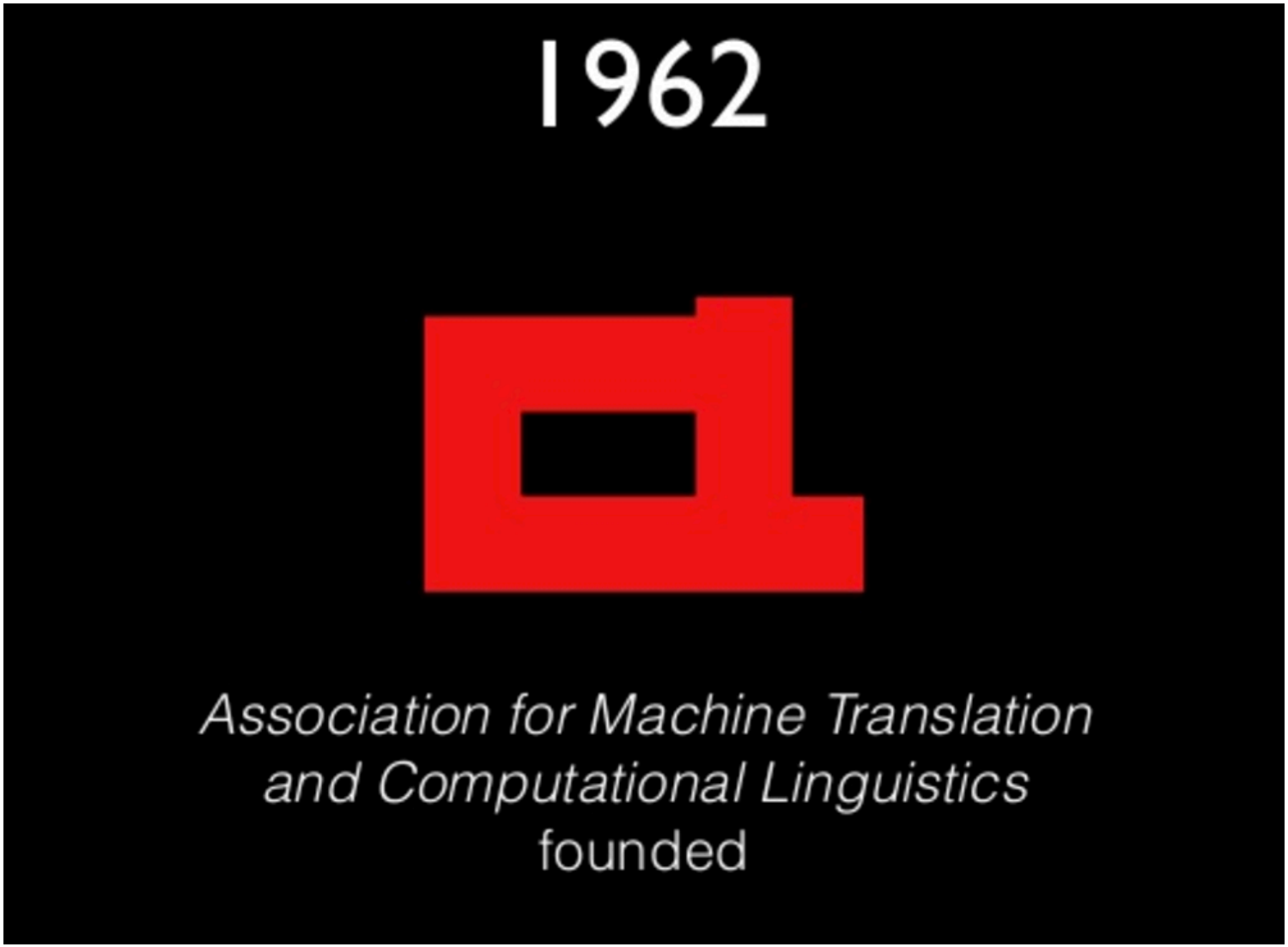


A brief history

When I look at an article in Russian, I say: "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Warren Weaver (1949)

A brief history



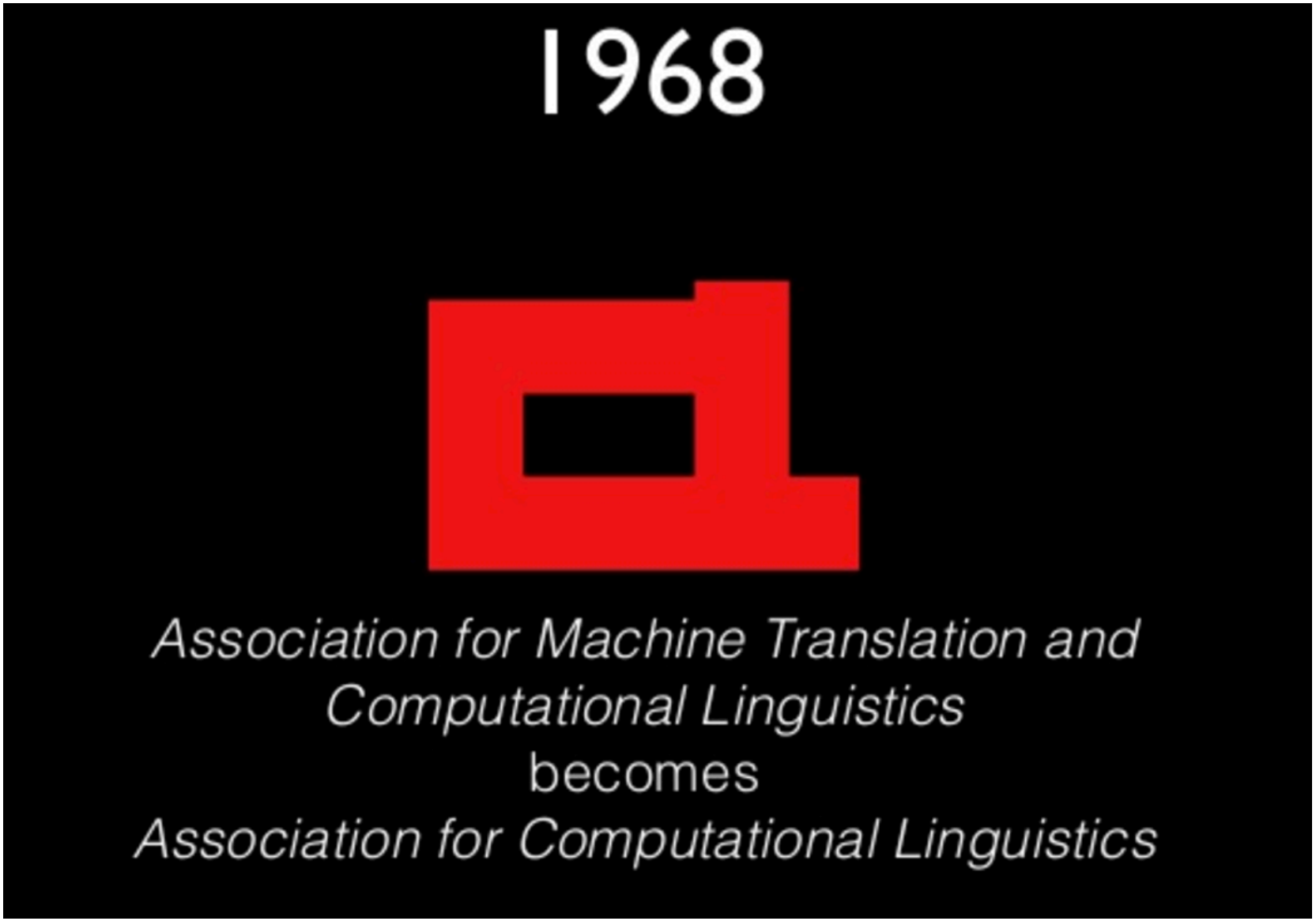
A brief history

1966

"'Machine Translation' presumably means going by algorithm from machine-readable source text to useful target text, without recourse to human translation or editing. In this context, there has been no machine translation of general scientific text, and none is in immediate prospect."

The ALPAC report

A brief history



A brief history

A brief history

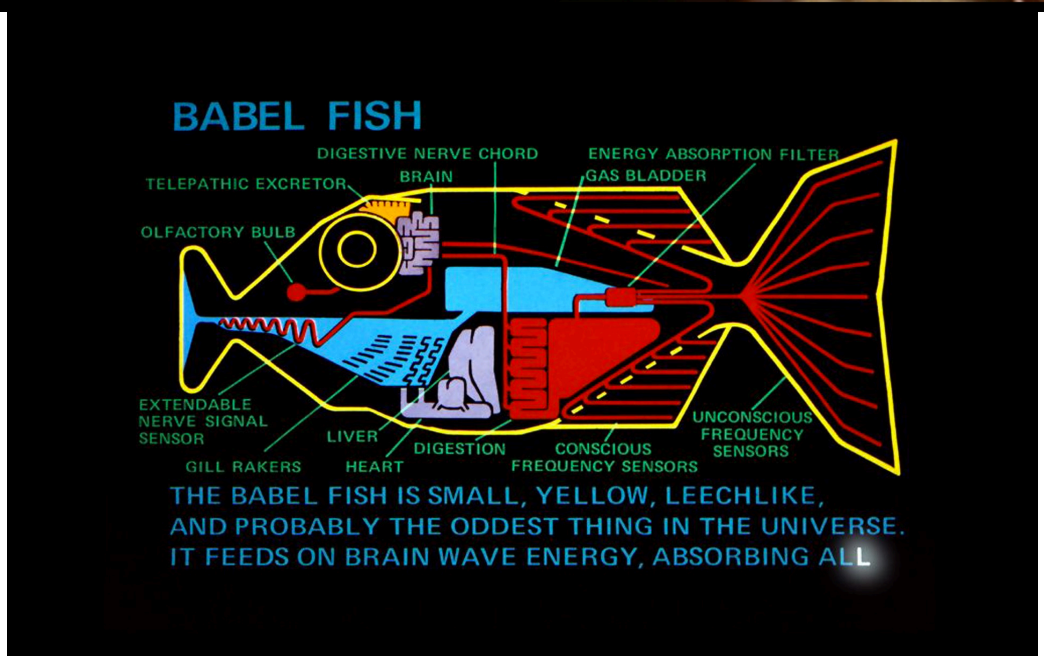
1975

$$\hat{W} = \arg\max_{W} P(W|A) = \arg\max_{W} P(A|W)P(W)$$

"Here's something we jocularly called the fundamental equation of speech recognition at ICASSP in 1981. As I'm sure all of you very well know, this is just an application of Bayes' Rule."

–Bob Mercer

A brief history



A brief history

1990

A Statistical Approach to Machine Translation

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin

IBM
Thomas J. Watson Research Center
Yorktown Heights, NY

In this paper, we present a statistical approach to machine translation. We describe the application of our approach to translation from French to English and give preliminary results.

1 Introduction

The field of machine translation is almost as old as the modern digital computer. In 1949 Warren Weaver suggested that the problem be attacked with statistical methods and ideas from information theory, an area which he, Claude Shannon, and others were developing at the time (Weaver 1949). Although researchers quickly abandoned this approach, advancing numerous theoretical objections, we believe that the true obstacles lay in the relative impotence of the available computers and the dearth of machine-readable text from which to gather the statistics vital to such an attack. Today, computers are five orders of magnitude faster than they were in 1950 and have hundreds of millions of bytes of storage. Large, machine-readable corpora are readily available. Statistical methods have proven their value in automatic speech recognition (Bahl et al. 1983) and have recently been applied to lexicography (Sinclair 1985) and to natural language processing (Baker

sentence in one language is a possible translation of any sentence in the other. We assign to every pair of sentences (S, T) a probability, $\Pr(T|S)$, to be interpreted as the probability that a translator will produce T in the target language when presented with S in the source language. We expect $\Pr(T|S)$ to be very small for pairs like (*Le matin je me brosse les dents*|*President Lincoln was a good lawyer*) and relatively large for pairs like (*Le president Lincoln était un bon avocat*|*President Lincoln was a good lawyer*). We view the problem of machine translation then as follows. Given a sentence T in the target language, we seek the sentence S from which the translator produced T. We know that our chance of error is minimized by choosing that sentence S that is most probable given T. Thus, we wish to choose S so as to maximize $\Pr(S|T)$. Using Bayes' theorem, we can write

$$\Pr(S|T) = \frac{\Pr(S)\Pr(T|S)}{\Pr(T)}$$

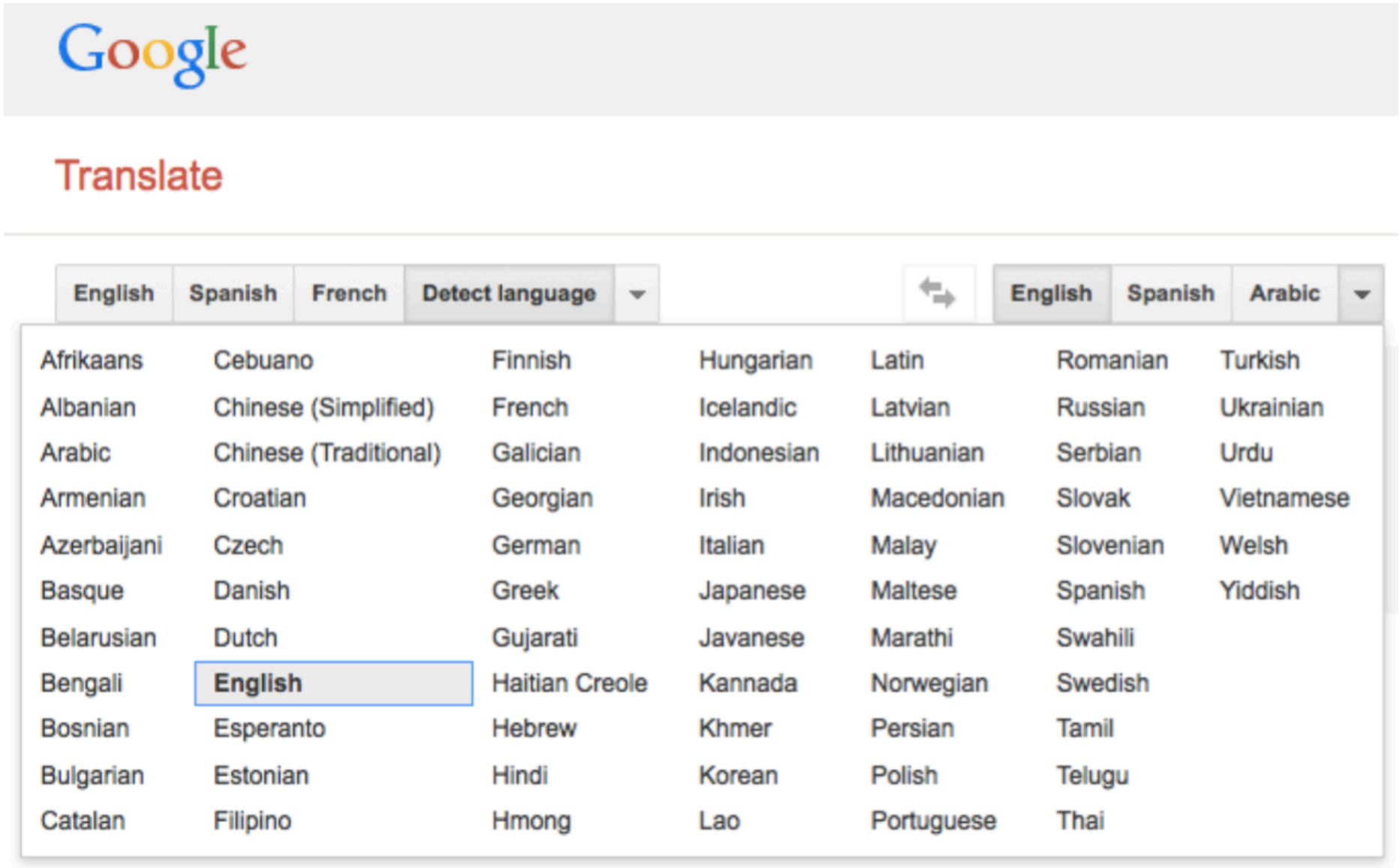
A brief history

Statistical Machine Translation Live

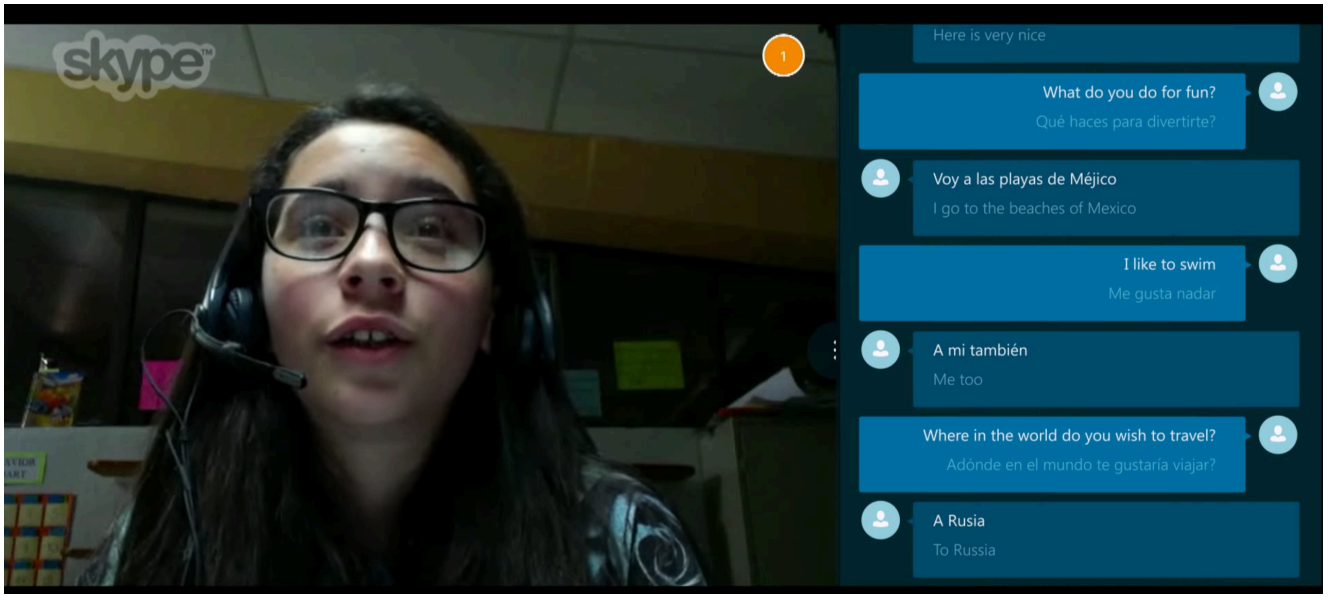
4/28/2006
Franz Och

Because we want to provide everyone with access to all the world's information, including information written in every language, one of the exciting projects at Google Research is machine translation... Now you can see the results for yourself. We recently launched an online version of our system for Arabic-English and English-Arabic. Try it out!

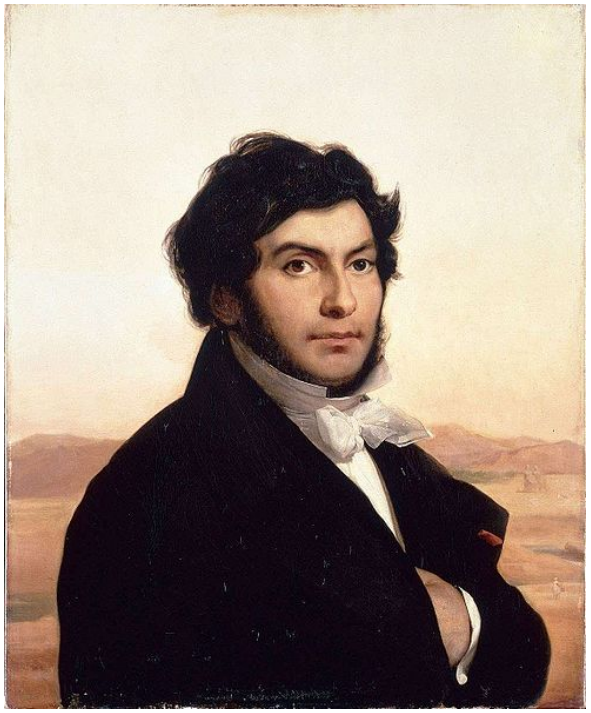
A brief history



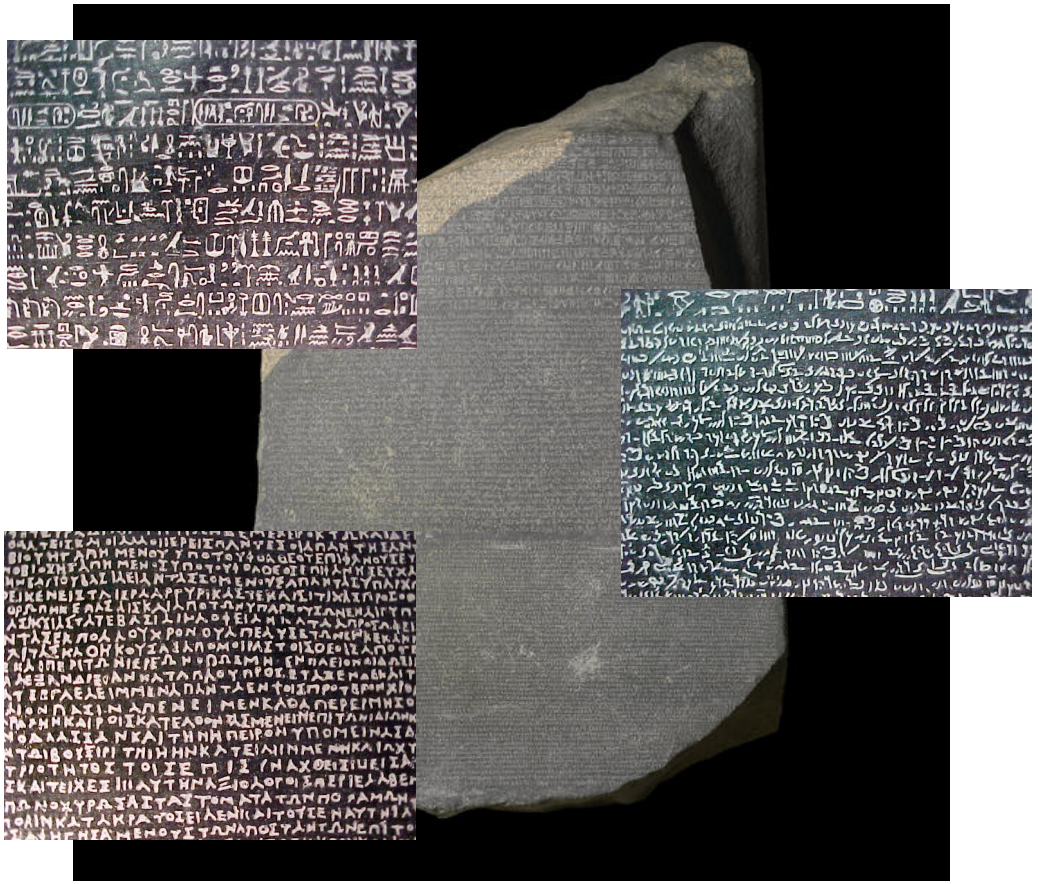
A brief history



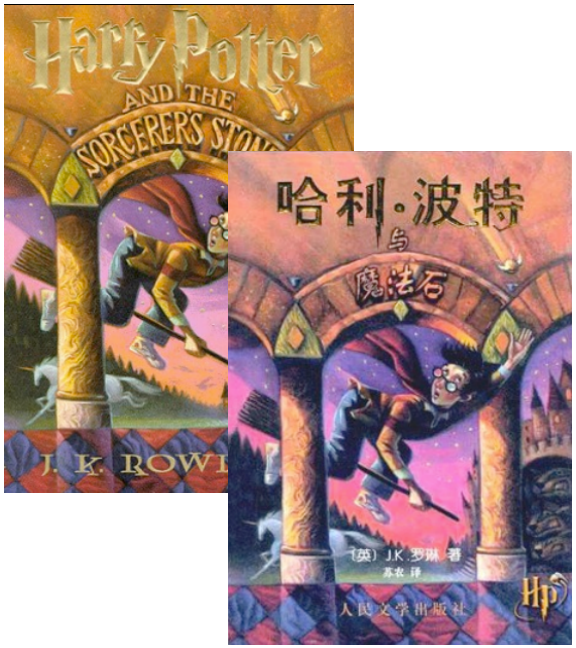
The main ingredient

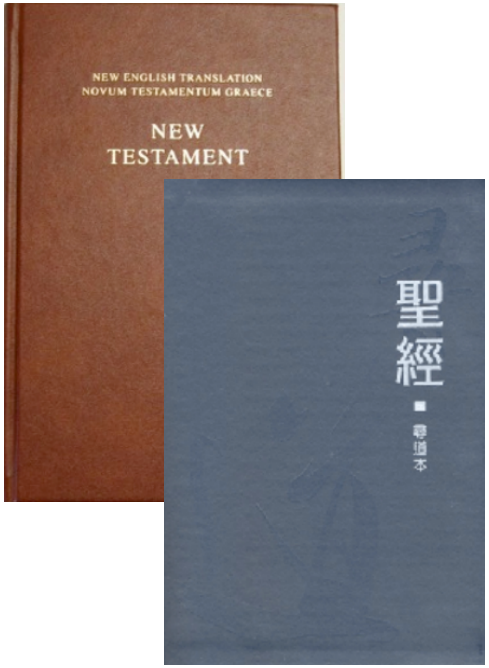


Jean-François Champollion
Rosetta Stone translation, 1818



The main ingredient





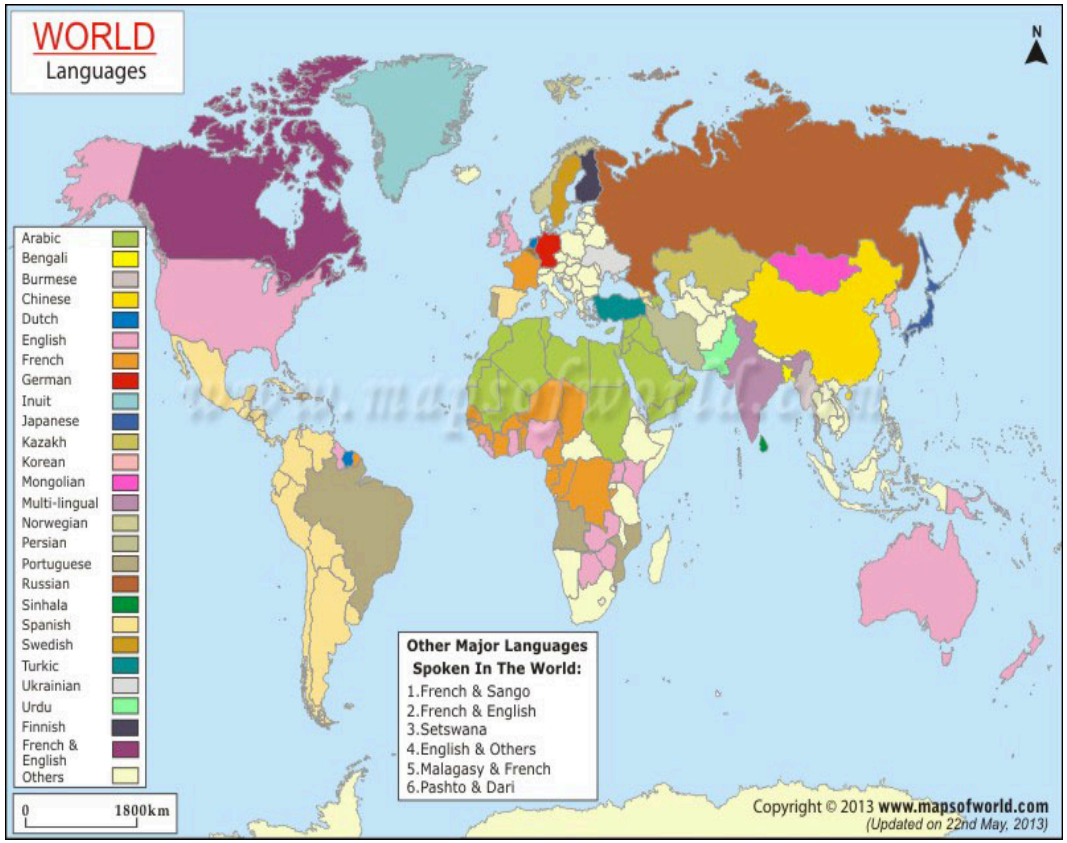
Interest in MT

Languages in the world

- 6,800 living languages
- 600 with written tradition
- 100 languages are spoken by 95% of world population

Translation market

- $26 Billion global market (2010)
- Doubling every five years
- EU spends $1 billion/year on Translation



Academic

- A challenging task
- Boost other NLP tasks

MT is hard

A human translator needs ...

- to understand the source language
- to know how to speak the target language (well!)
- knowledge about the relationship between source and target language
- knowledge about the topic of the text to be translated
- knowledge about culture, values, traditions and expectation av speakers of source and target language

Computers have big problems with all of these issues

MT is hard

Word Translation Problems

- Words are ambiguous

He deposited money in a bank account
with a high interest rate.

Sitting on the bank of the Mississippi,
a passing ship piqued his interest.

- How do we find the right meaning, and thus translation?
- Context should be helpful

MT is hard

Syntactic Translation Problems

- Languages have different sentence structure

das	behaupten	sie	wenigstens
this	claim	they	at least
the		she	

- Convert from object-verb-subject (OVS) to subject-verb-object (SVO)

- Ambiguities can be resolved through syntactic analysis
  - the meaning the of das not possible (not a noun phrase)
  - the meaning she of sie not possible (subject-verb agreement)

MT is hard

Semantic Translation Problems

- Pronominal anaphora

  I saw the movie and it is good.

- How to translate it into German (or French)?
  - it refers to movie
  - movie translates to Film
  - Film has masculine gender
  - ergo: it must be translated into masculine pronoun er

MT is hard

Semantic Translation Problems

- Coreference

Whenever I visit my uncle and his daughters,
I can't decide who is my favorite cousin.

- How to translate cousin into German? Male or female?

- Complex inference required

MT is hard

Semantic Translation Problems

- Discourse

Since you brought it up, I do not agree with you.

Since you brought it up, we have been working on it.

- How to translated since? Temporal or conditional?

- Analysis of discourse structure — a hard problem

Two views of MT

- **Direct modeling** (aka pattern matching)
  - I have **really good learning algorithms** and a bunch of **example inputs** (source language sentences) and **outputs** (target language translations)
- **Code breaking** (aka the noisy channel, Bayes rule)
  - I know the **target language**
  - I have example **translations texts** (example enciphered data)

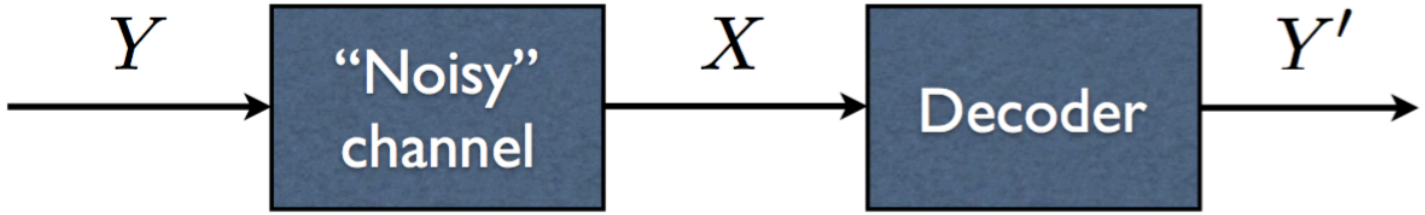
MT as direct modeling

$$\hat{e} = \arg\max_{e} p_{\lambda}(e \mid f)$$

target source

- one model does everything
- trained to reproduce a corpus of translations

MT as code breaking (noisy channel)



Sent transmission

English

Received transmission

"French"

Recovered message

English'

$$\hat{e} = \arg\max_{e} p_{\varphi}(e) \times p_{\theta}(f \mid e)$$

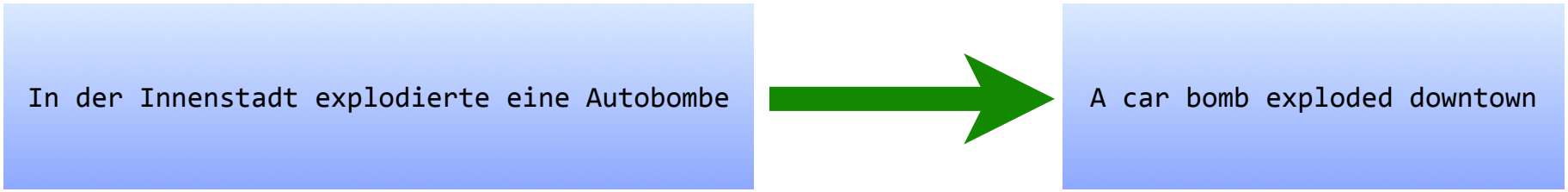
language model

translation model

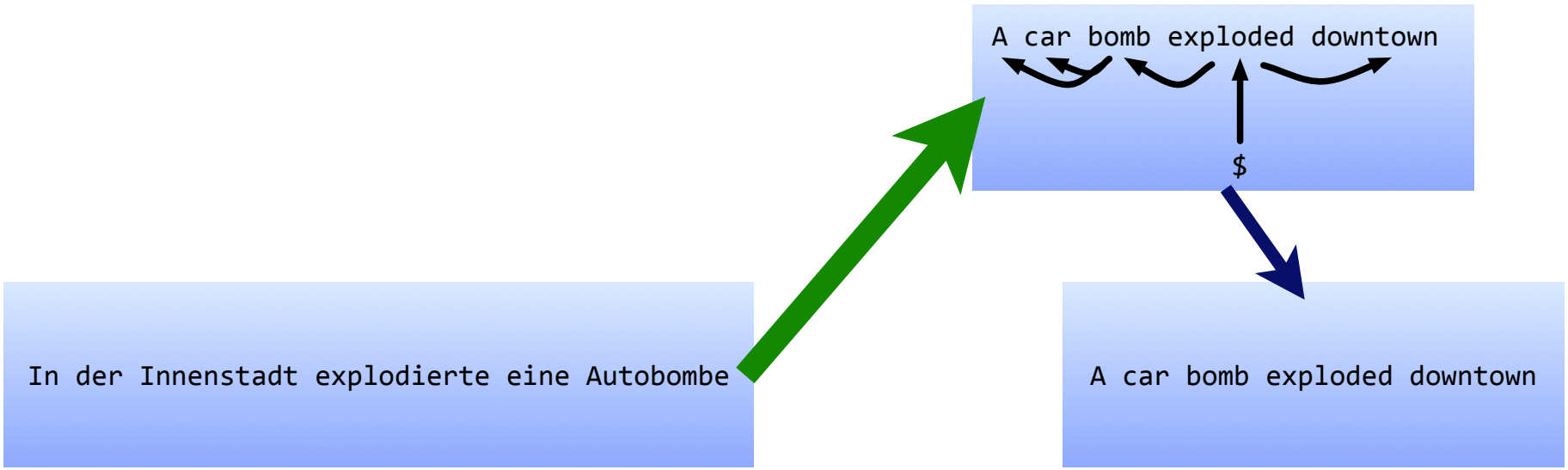
Which view in 2016?

- Direct modeling is where most of the action is
  - Neural networks are very good at generalizing - and conceptually very simple
  - Inference in “product of two models” is hard
- Noisy channel ideas are incredibly important and still play a big role in how we think about translation
- We will cover both views

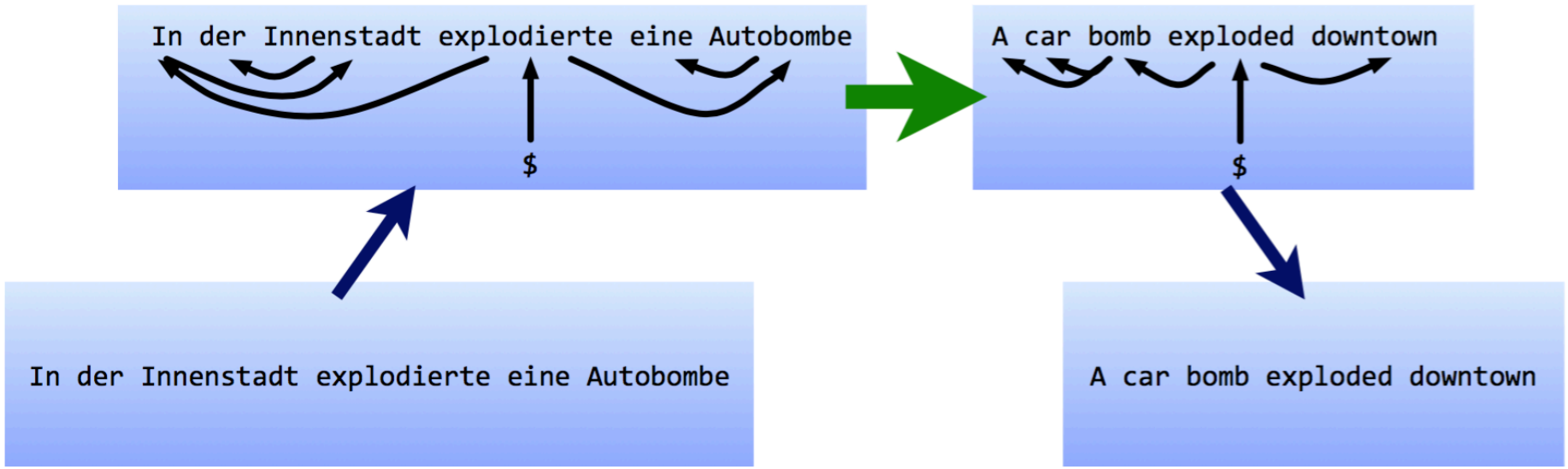
Translation Modeling



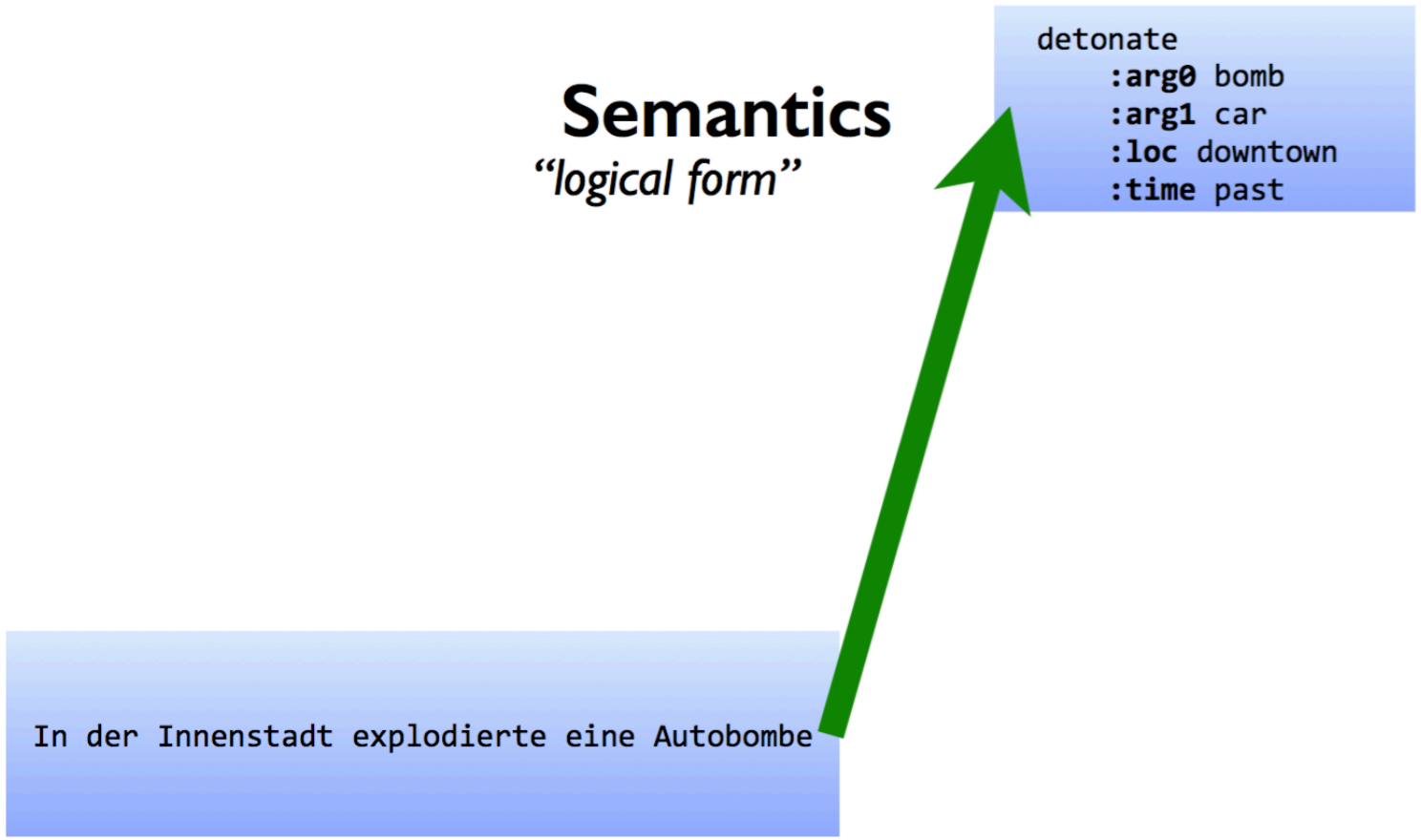
Translation Modeling



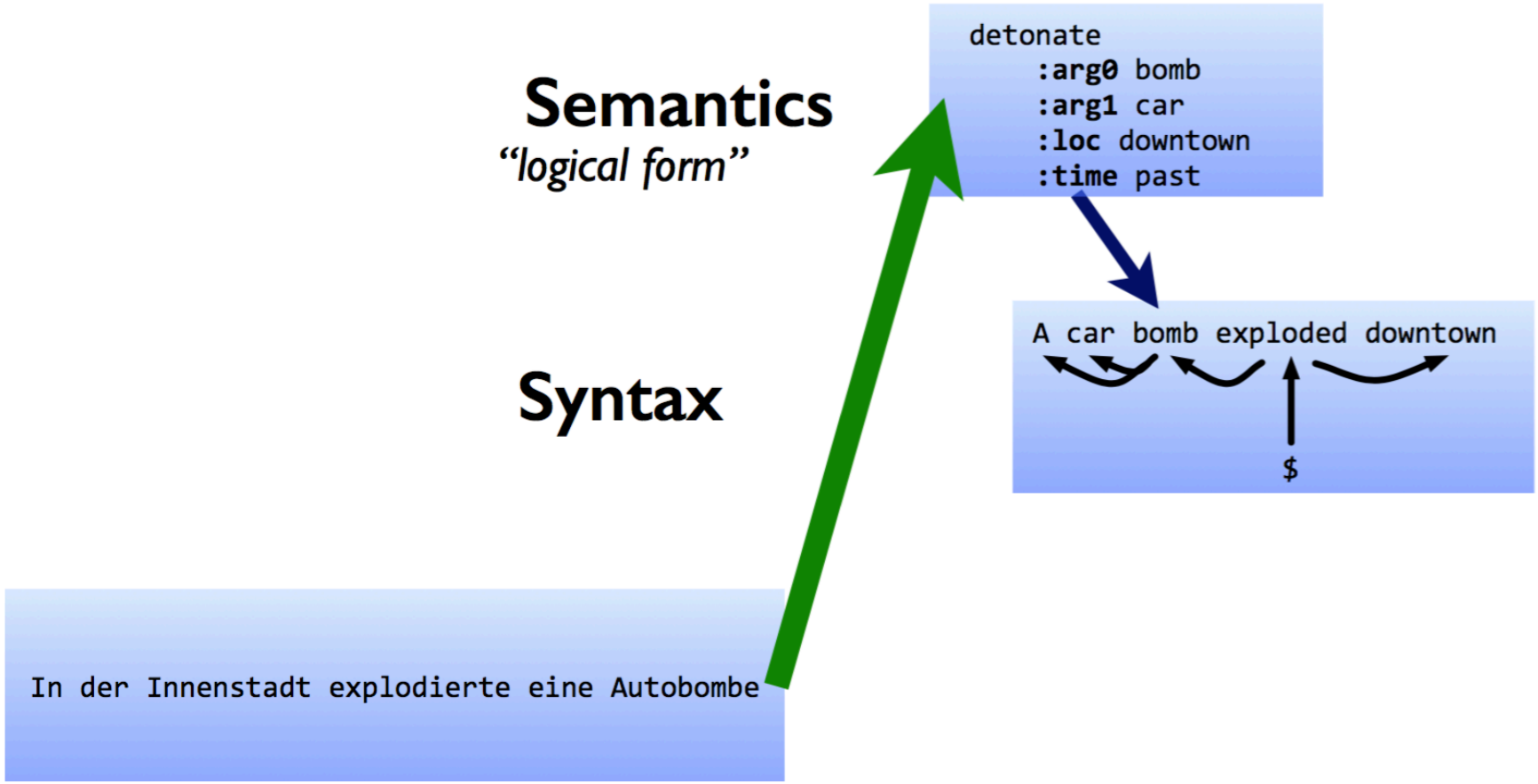
Translation Modeling



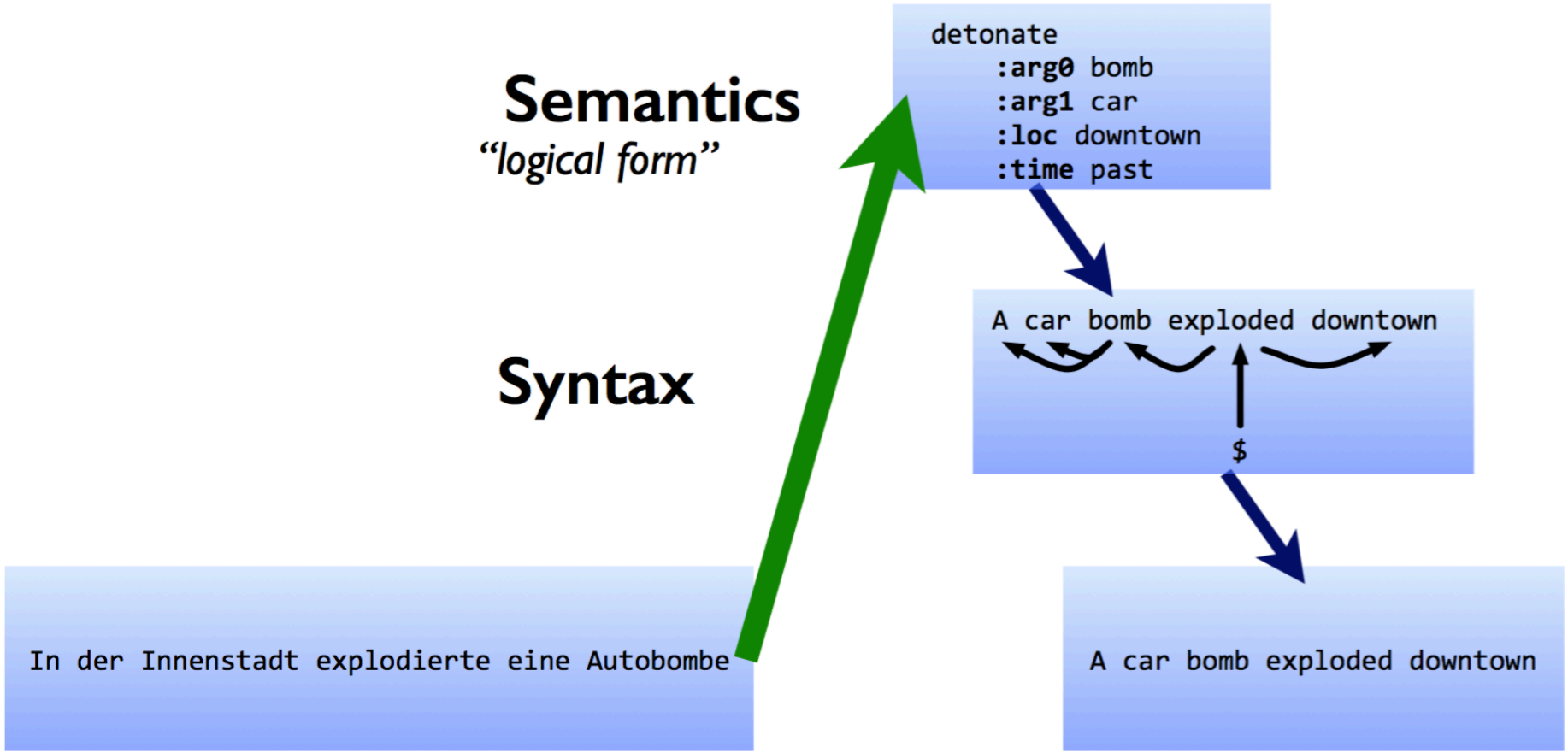
Translation Modeling



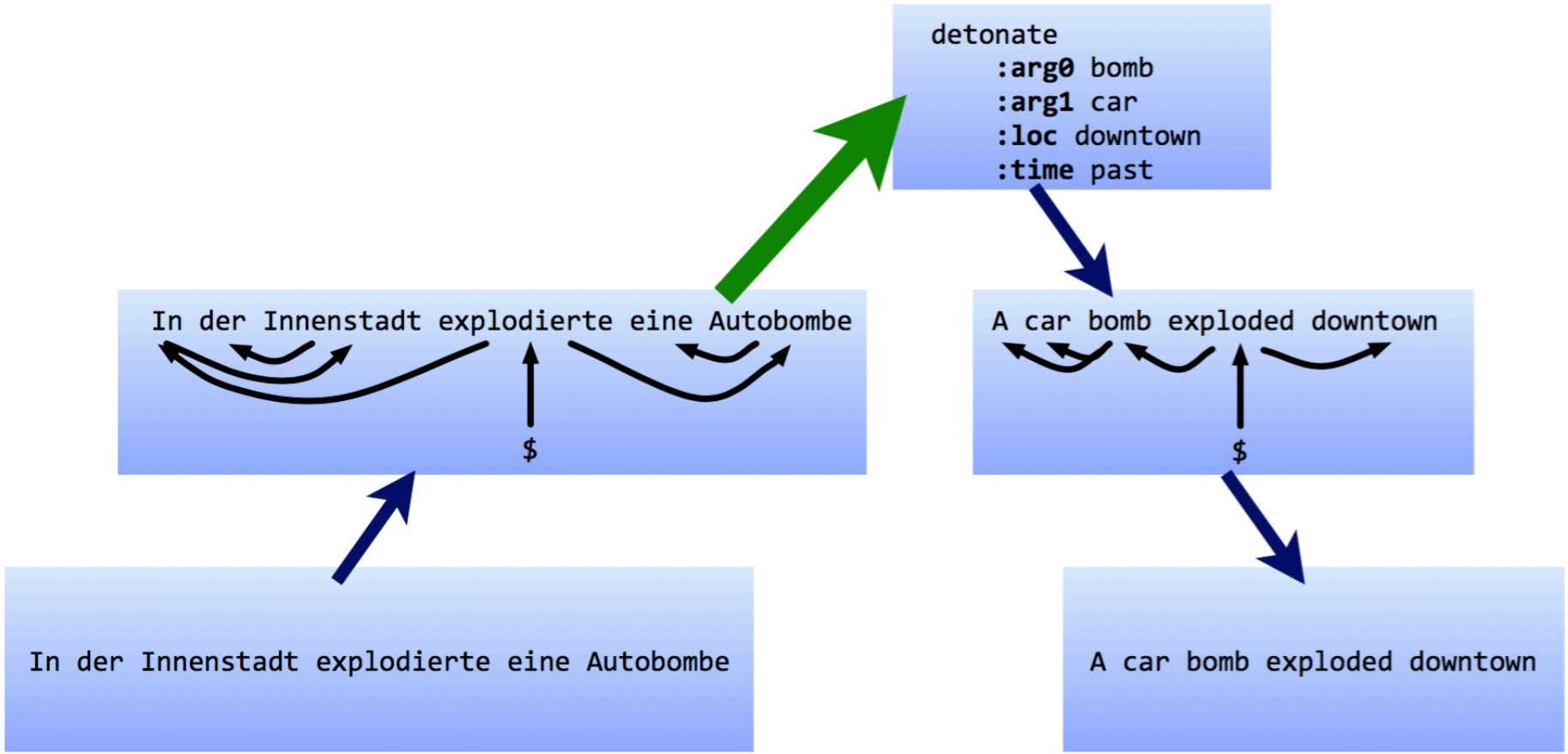
Translation Modeling



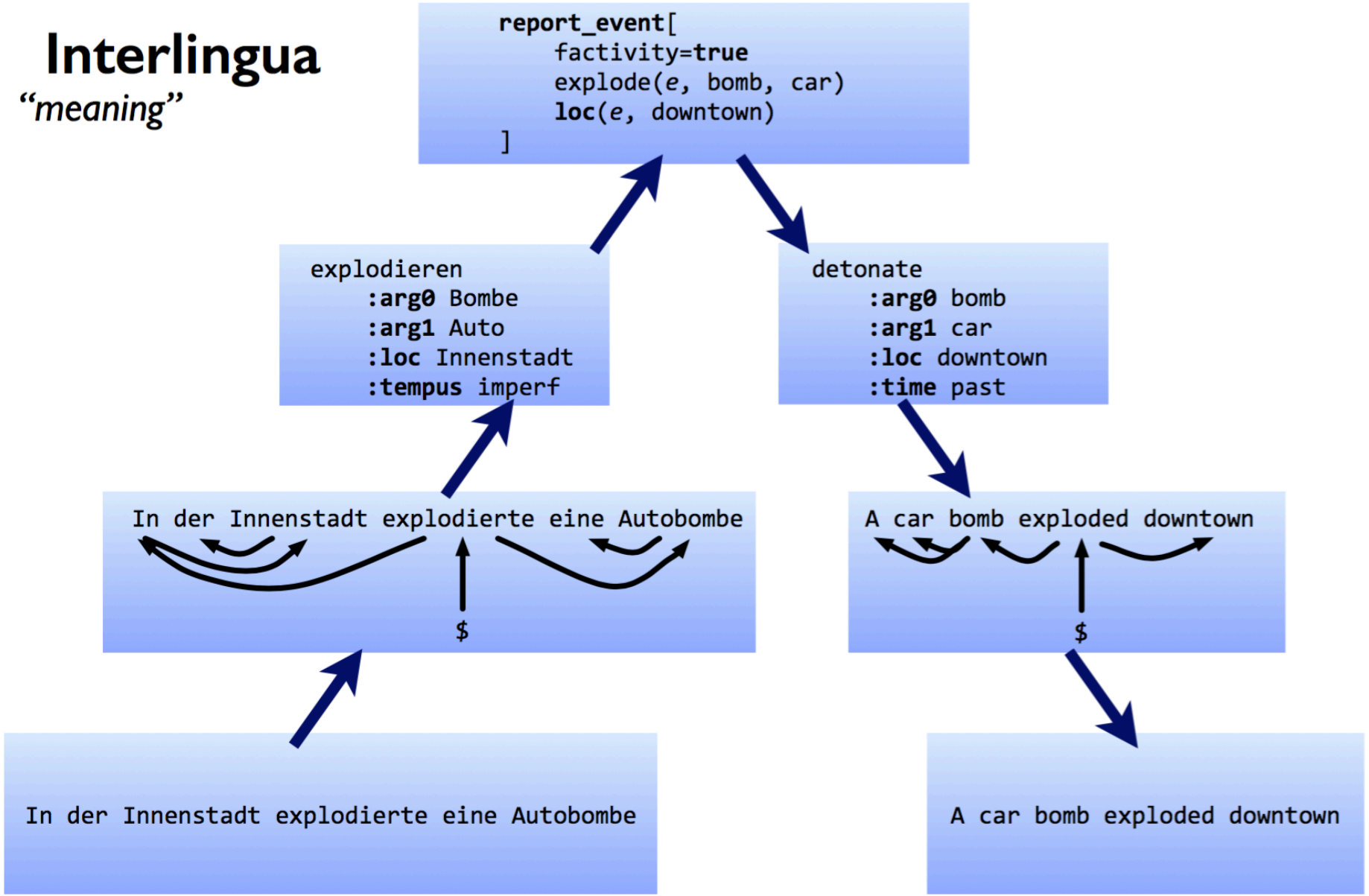
Translation Modeling



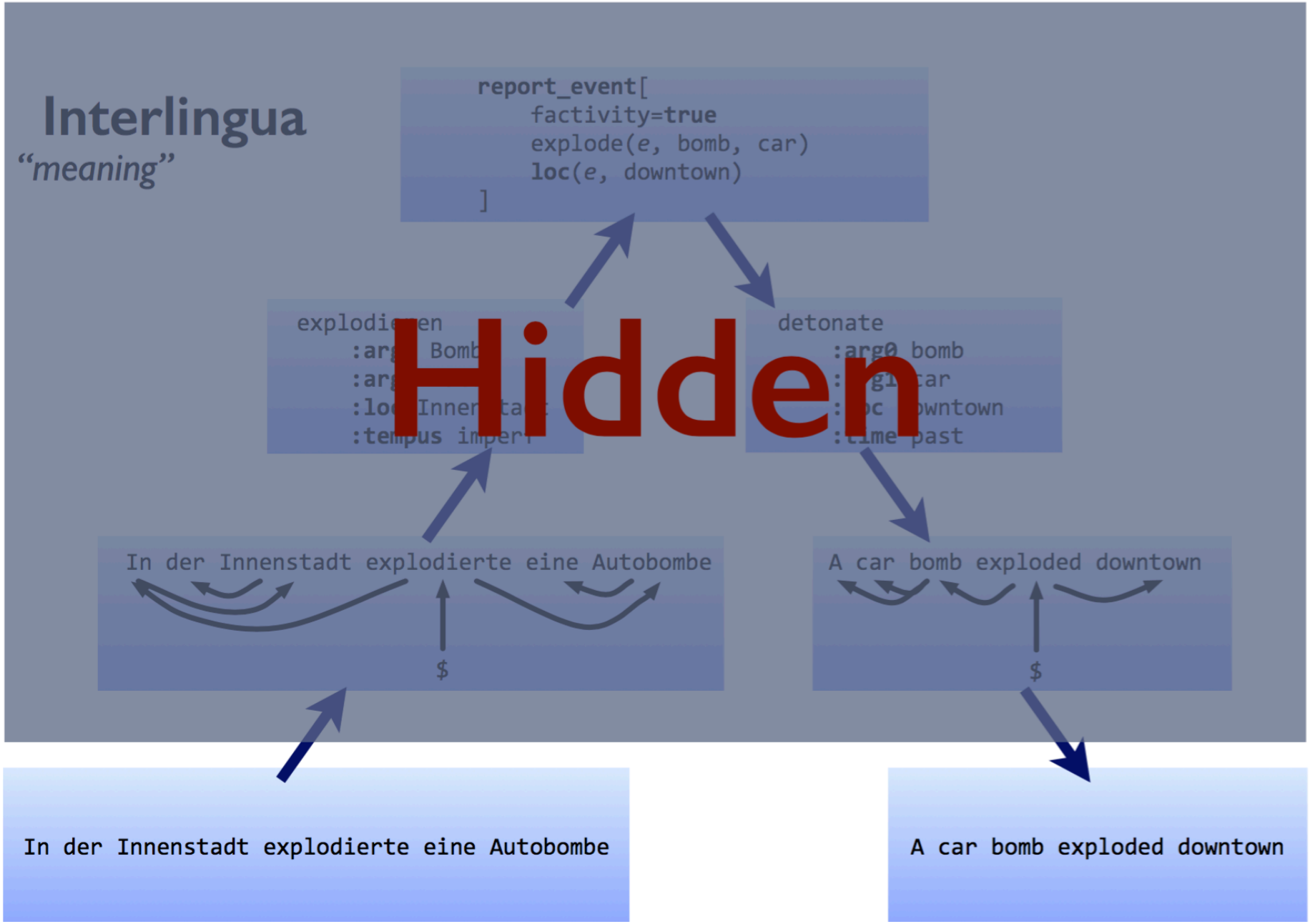
Translation Modeling



Translation Modeling



Translation Modeling



Evaluation: many right answers

这个 机场 的 安全 工作 由 以色列 方面 负责 .

Israeli officials are responsible for airport security.
Israel is in charge of the security at this airport.
The security work for this airport is the responsibility of the Israel government.
Israeli side was in charge of the security of this airport.
Israel is responsible for the airport's security.
Israel is responsible for safety work at this airport.
Israel presides over the security of the airport.
Israel took charge of the airport security.
The safety of this airport is taken charge of by Israel.
This airport's security is the responsibility of the Israeli security officials.

Evaluation: BLEU

N-Gram precision

$$p_n = \frac{\sum_{n\text{-gram} \in hyp} count_{clip}(n\text{-gram})}{\sum_{n\text{-gram} \in hyp} count(n\text{-gram})}$$

Bounded above by highest count of n-gram in any reference sentence

brevity penalty

$$\mathbf{B} = \begin{cases} \mathbf{e^{(1-\,|ref|\,/\,|hyp|)}} & \mathbf{if\ |ref| > |hyp|} \\ \mathbf{1} & \mathbf{otherwise} \end{cases}$$

Bleu score: brevity penalty, geometric mean of N-Gram precisions

$$\mathbf{Bleu=}\quad B \cdot \exp\left[\frac{1}{N}\sum_{n=1}^{N} p_n\right]$$

Plan

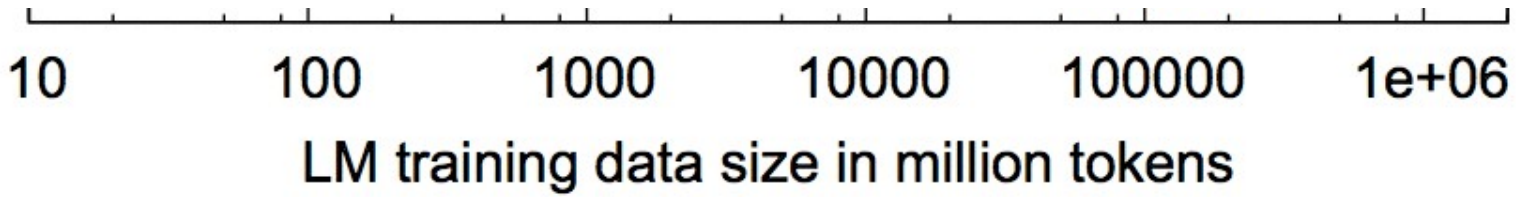
- Language models
- Word-based translation models
- Phrase-based translation models
- Decoding in PBSMT
- Discriminative modeling
- Neural machine translation

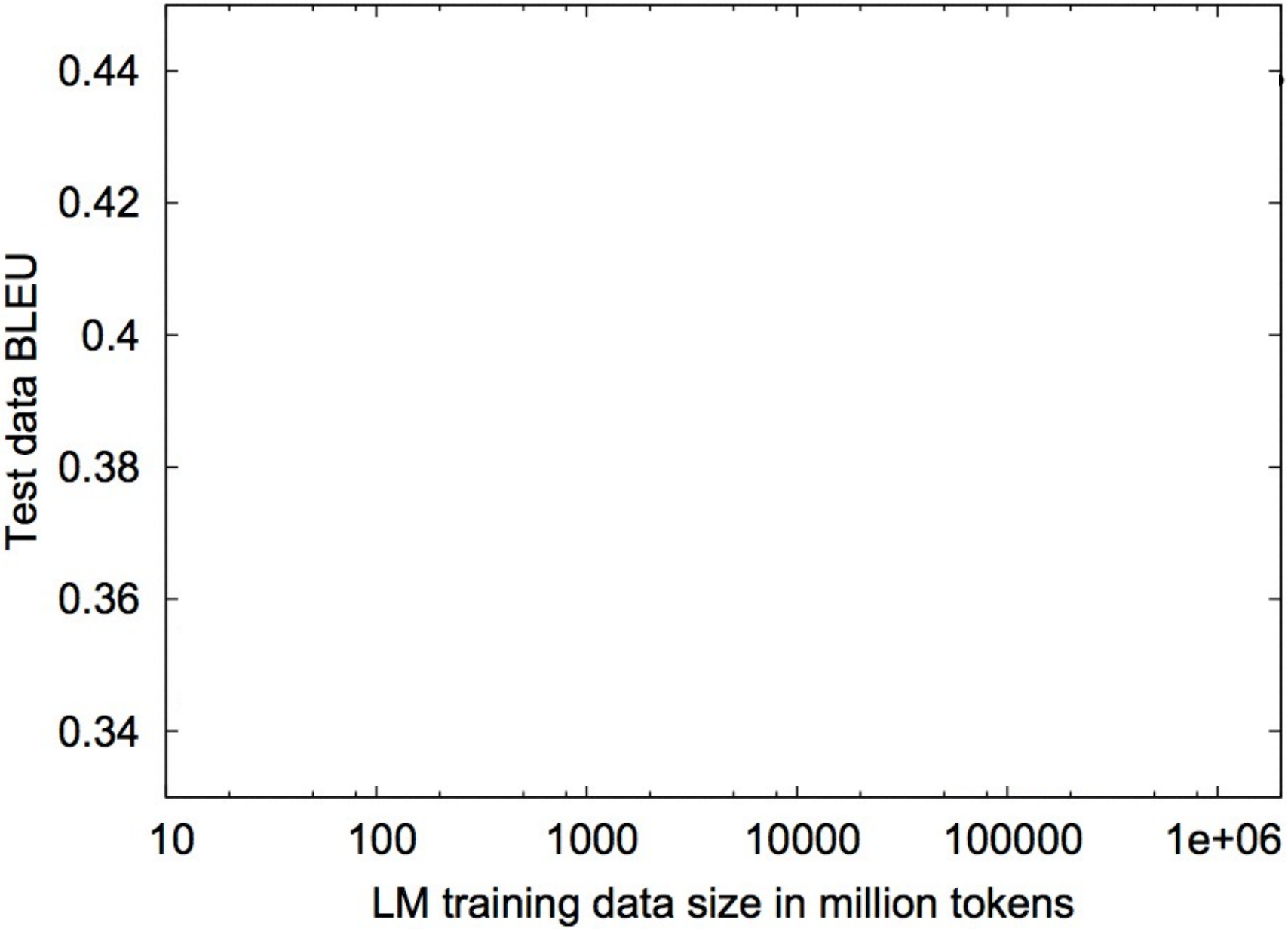
Language Models
(Lecture 5)

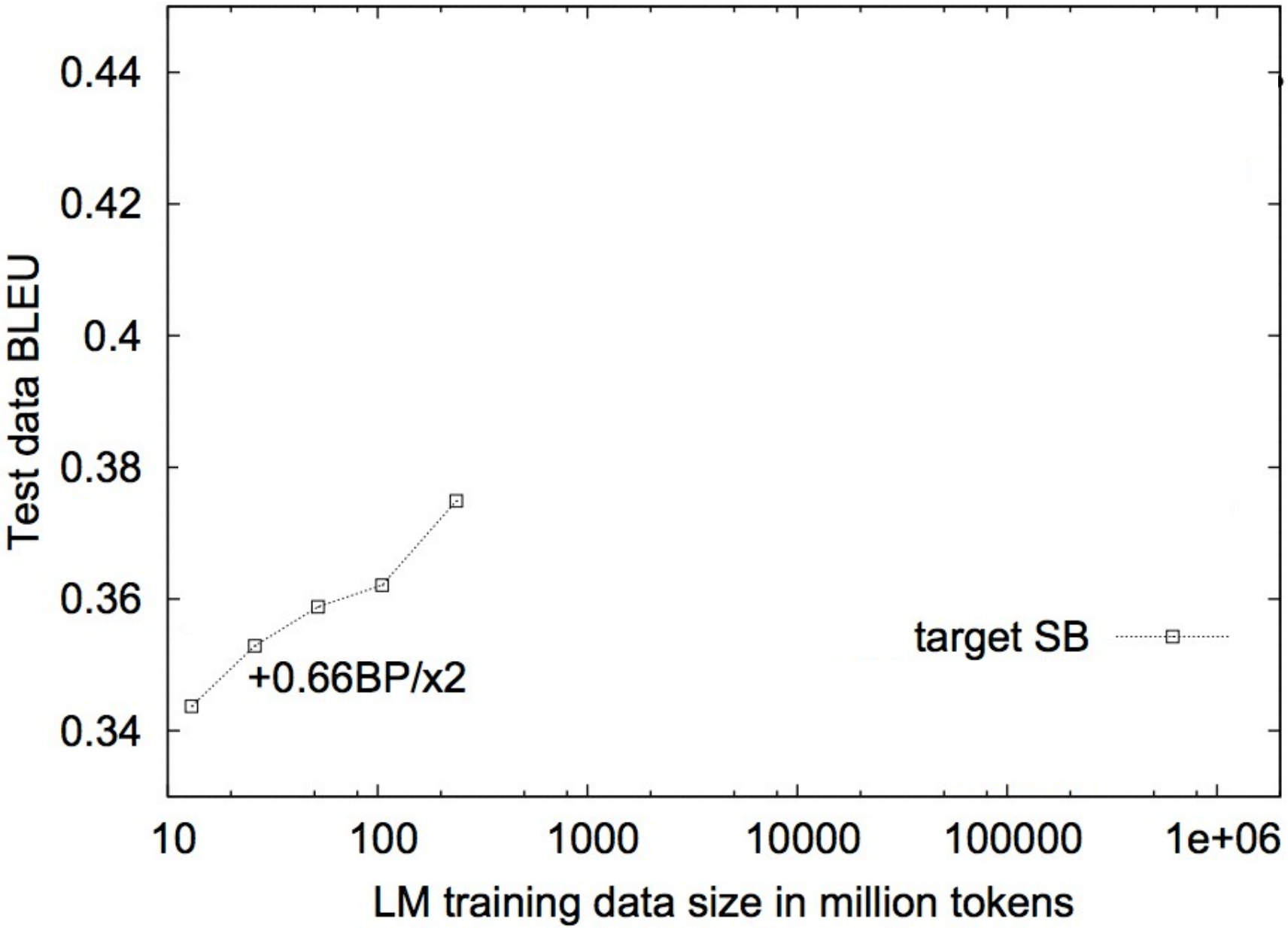
Language Models

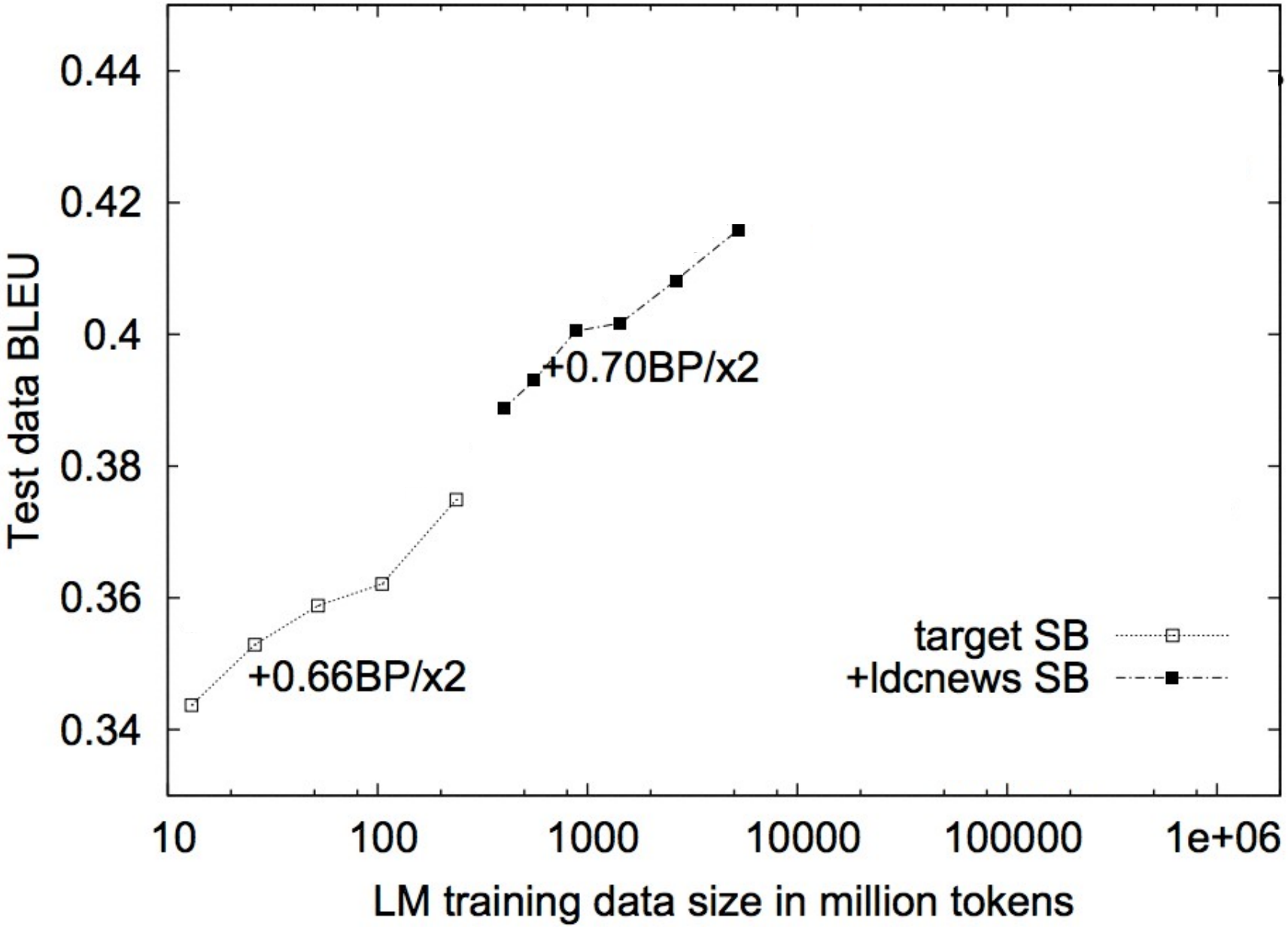
- Models of p(sentence)
- Later

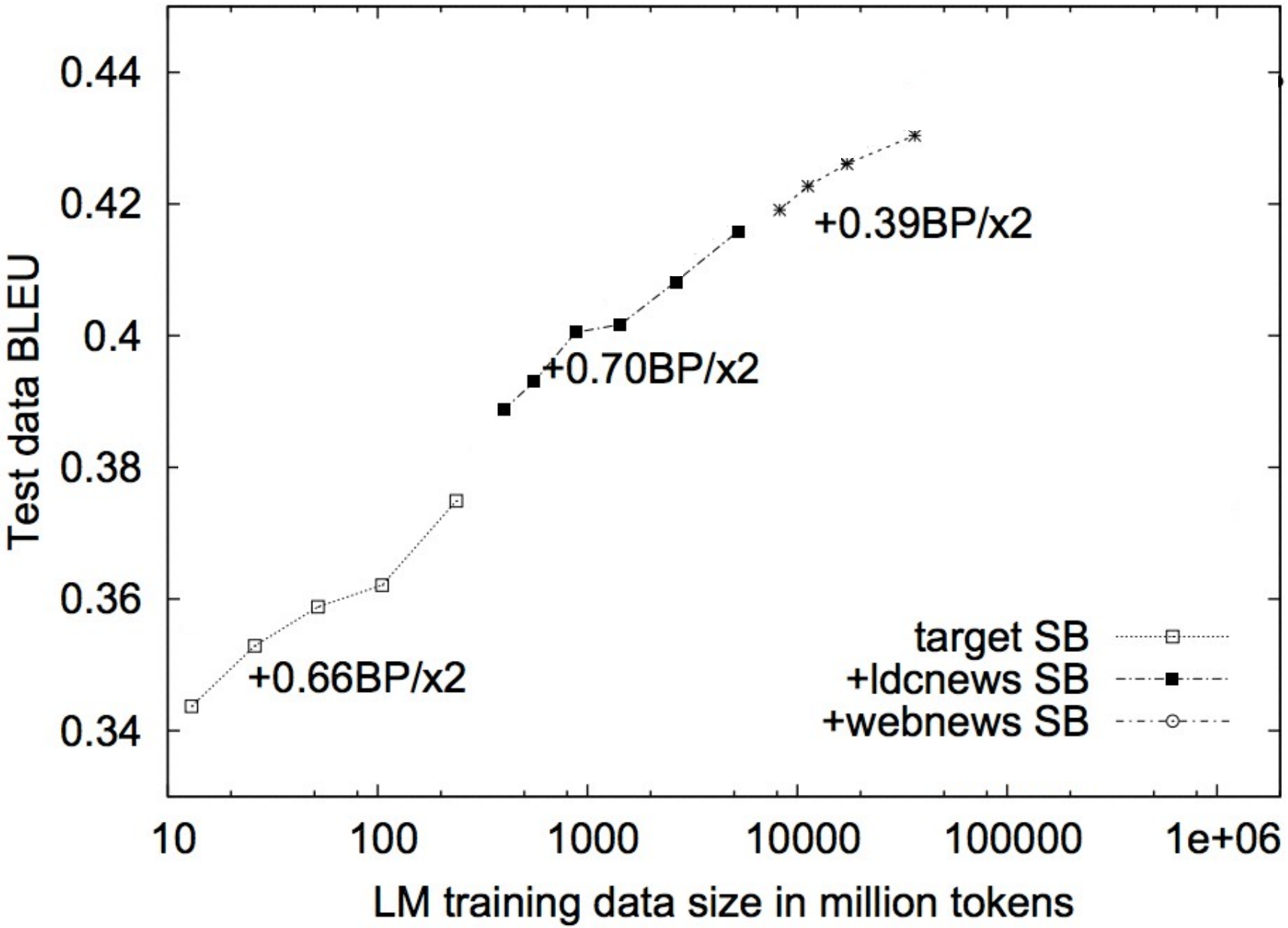
  p(translated sentence | input sentence)
- Why do it this way?
  - Conditioning on more stuff makes modeling more complicated. That is: p(sentence) is easier than p(translated sentence | input sentence).
  - Language models are arguably the most important models in statistical MT

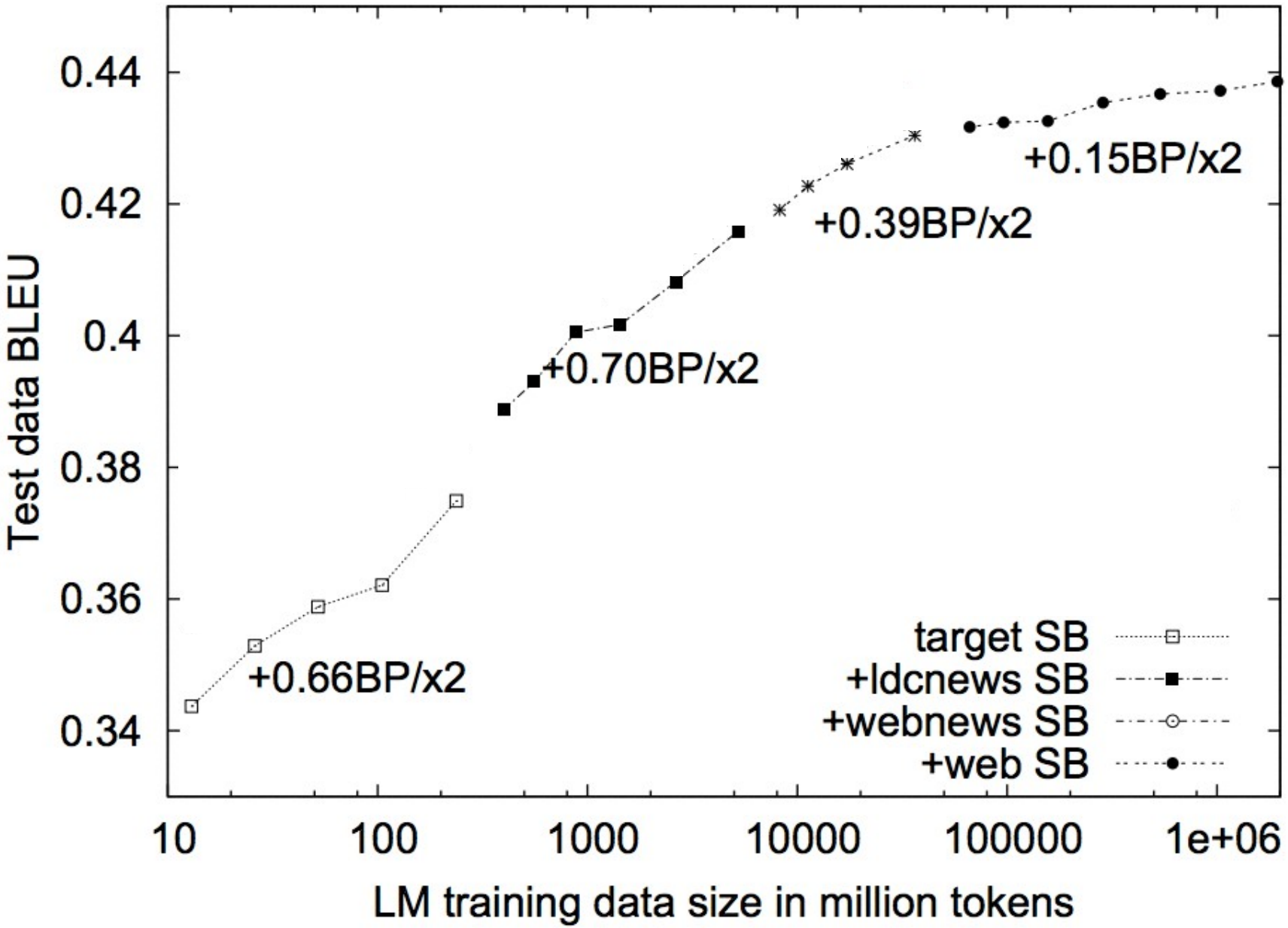

10
100
1000
10000
100000
1e+06
LM training data size in million tokens


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
10
100
1000
10000
100000
1e+06
LM training data size in million tokens


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
+0.66BP/x2
target SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
+0.70BP/x2
+0.66BP/x2
target SB
+ldcnews SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
+0.39BP/x2
+0.70BP/x2
+0.66BP/x2
target SB
+ldcnews SB
+webnews SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
+0.66BP/x2
+0.70BP/x2
+0.39BP/x2
+0.15BP/x2
target SB
+ldcnews SB
+webnews SB
+web SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens

Language Models Matter

- Language models play the role of ...
  - a judge of *grammaticality*
  - a judge of *semantic plausibility*
  - an enforcer of *stylistic consistency*

Lexical (word-based) Translation Models

Lexical Translation

- How do we translate a word? Look it up in the dictionary

  Haus : *house, home, shell, household*

- Multiple translations
  - Different word senses, different registers, different inflections (?)
  - *house, home* are common
  - *shell* is specialized (the Haus of a snail is a shell)

How common is each?

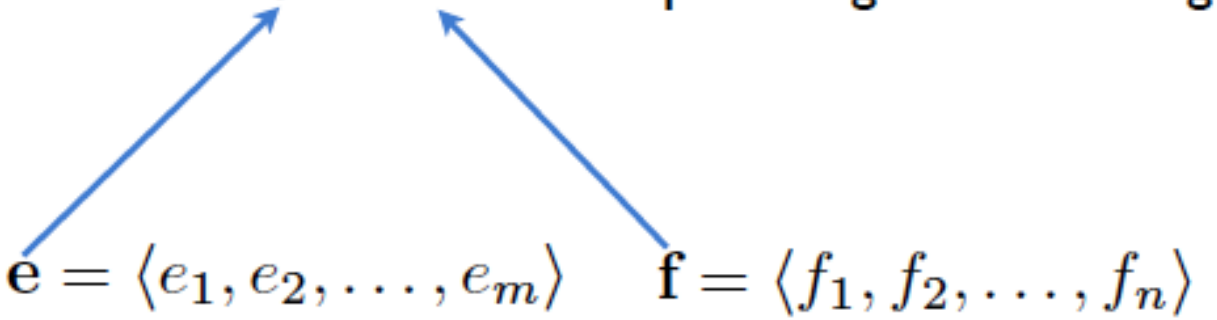
Translation	Count
house	5000
home	2000
shell	100
household	80

MLE

$$\hat{p}_{\text{MLE}}(e \mid \texttt{Haus}) = \begin{cases} 0.696 & \text{if } e = \texttt{house} \\ 0.279 & \text{if } e = \texttt{home} \\ 0.014 & \text{if } e = \texttt{shell} \\ 0.011 & \text{if } e = \texttt{household} \\ 0 & \text{otherwise} \end{cases}$$

Lexical Translation

- Goal: a model $p(\mathbf{e} \mid \mathbf{f}, m)$
- where $\mathbf{e}$ and $\mathbf{f}$ are complete English and Foreign sentences



Lexical Translation

- Goal: a model $p(\mathbf{e} \mid \mathbf{f}, m)$
- where $\mathbf{e}$ and $\mathbf{f}$ are complete English and Foreign sentences
- Lexical translation makes the following ***assumptions***:
  - Each word in e_i in $\mathbf{e}$ is generated from exactly one word in $\mathbf{f}$
  - Thus, we have an *alignment* a_i that indicates which word e_i "came from", specifically it came from f_{a_i}.
  - Given the alignments $\mathbf{a}$, translation decisions are conditionally independent of each other and depend *only* on the aligned source word f_{a_i}.

Lexical Translation

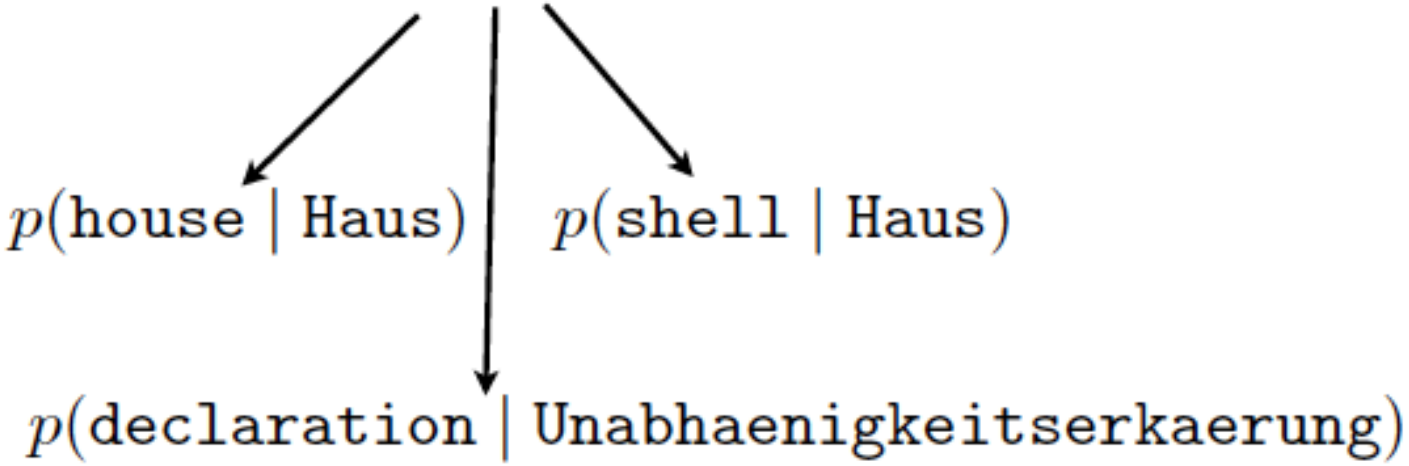
- Putting our assumptions together, we have:

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

Alignment × Translation | Alignment

Lexical Translation

$$p(e_i \mid f_{a_i})$$



Remember bigram models...

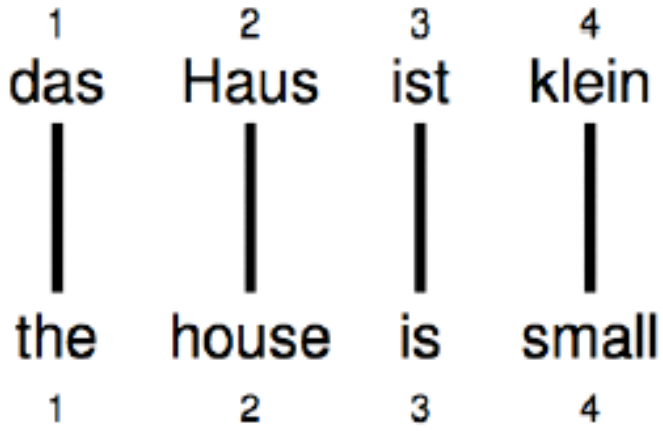
Alignment

$$p(\mathbf{a} \mid \mathbf{f}, m)$$

Most of the action for the first 10 years of MT was here. Words weren't the problem, word *order* was hard.

Alignment

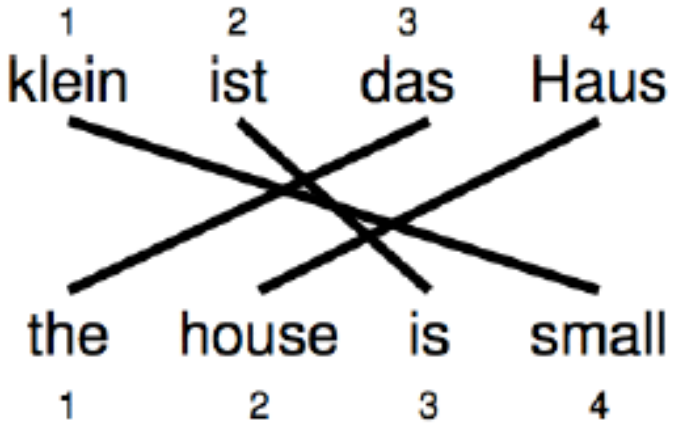
- Alignments can be visualized in by drawing links between two sentences, and they are represented as vectors of positions:



$$\mathbf{a} = (1, 2, 3, 4)^\top$$

Reordering

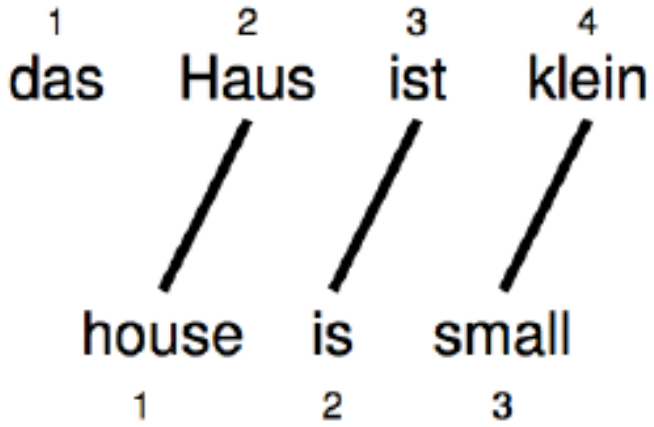
- Words may be reordered during translation.



$$\mathbf{a} = (3, 4, 2, 1)^\top$$

Word Dropping

- A source word may not be translated at all

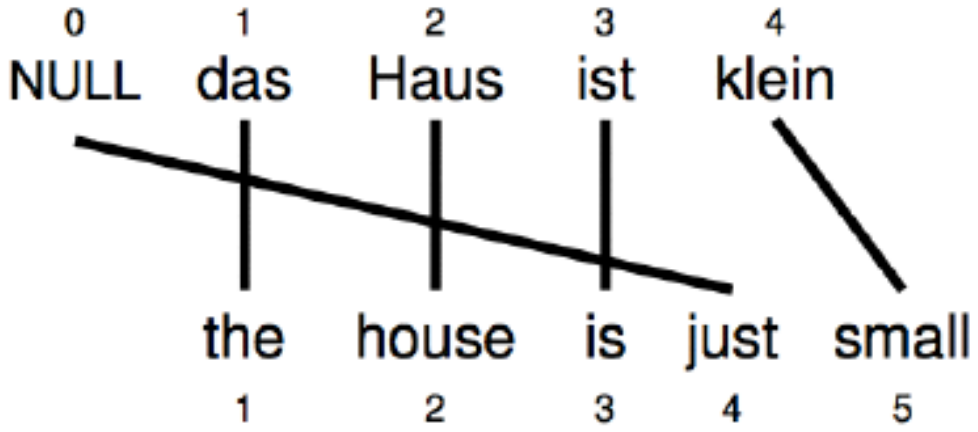


$$\mathbf{a} = (2, 3, 4)^\top$$

Word Insertion

- Words may be inserted during translation
  English just does not have an equivalent

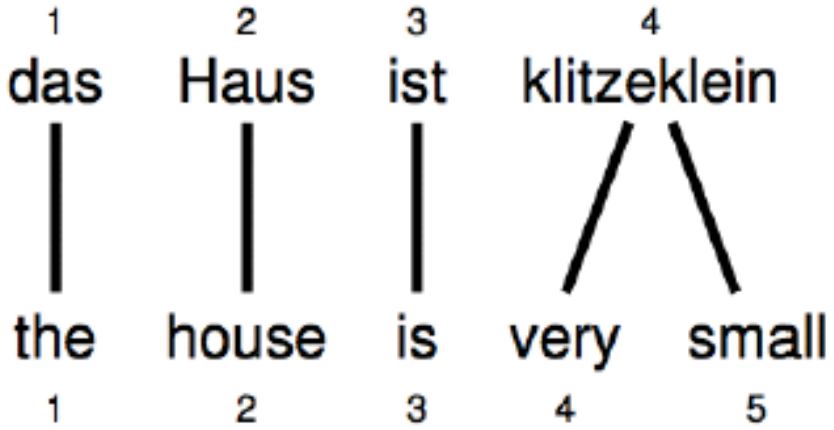
  But it must be explained - we typically assume every source sentence contains a NULL token



$$\mathbf{a} = (1, 2, 3, 0, 4)^\top$$

One-to-many Translation

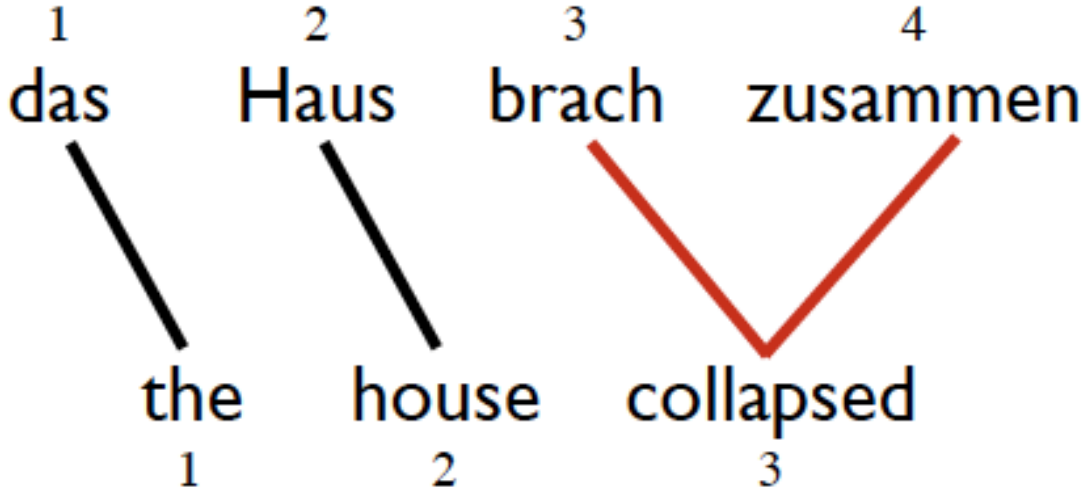
- A source word may translate into **more than one** target word



$$\mathbf{a} = (1, 2, 3, 4, 4)^\top$$

Many-to-one Translation

- **More than one source word** may **not** translate as a unit in lexical translation



$\mathbf{a} = ???$

IBM Model 1

- Simplest possible lexical translation model
- Additional assumptions
  - The m alignment decisions are independent
  - The alignment distribution for each a_i is uniform over all source words and NULL

$$\text{for each } i \in [1, 2, \ldots, m]$$
$$a_i \sim \text{Uniform}(0, 1, 2, \ldots, n)$$
$$e_i \sim \text{Categorical}(\boldsymbol{\theta}_{f_{a_i}})$$

IBM Model 1

for each $i \in [1, 2, \ldots, m]$

$$a_i \sim \text{Uniform}(0, 1, 2, \ldots, n)$$

$$e_i \sim \text{Categorical}(\boldsymbol{\theta}_{f_{a_i}})$$

$$p(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m) = \prod_{i=1}^{m}$$

IBM Model 1

for each $i \in [1, 2, \ldots, m]$

$$a_i \sim \text{Uniform}(0, 1, 2, \ldots, n)$$

$$e_i \sim \text{Categorical}(\boldsymbol{\theta}_{f_{a_i}})$$

$$p(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m) = \prod_{i=1}^{m} \frac{1}{1+n}$$

IBM Model 1

for each $i \in [1, 2, \ldots, m]$

$$a_i \sim \text{Uniform}(0, 1, 2, \ldots, n)$$

$$e_i \sim \text{Categorical}(\boldsymbol{\theta}_{f_{a_i}})$$

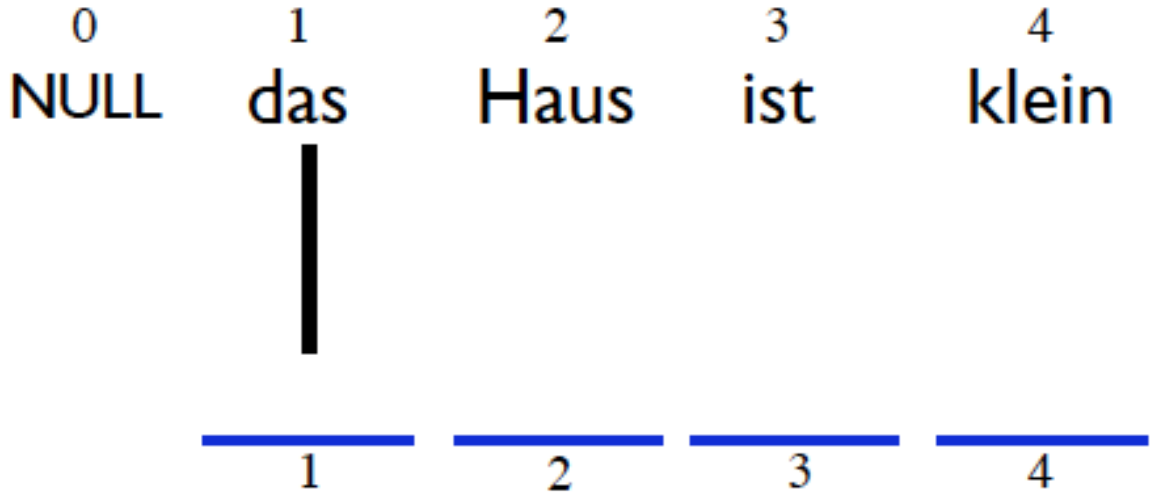
$$p(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m) = \prod_{i=1}^{m} \frac{1}{1+n} \, p(e_i \mid f_{a_i})$$

Example

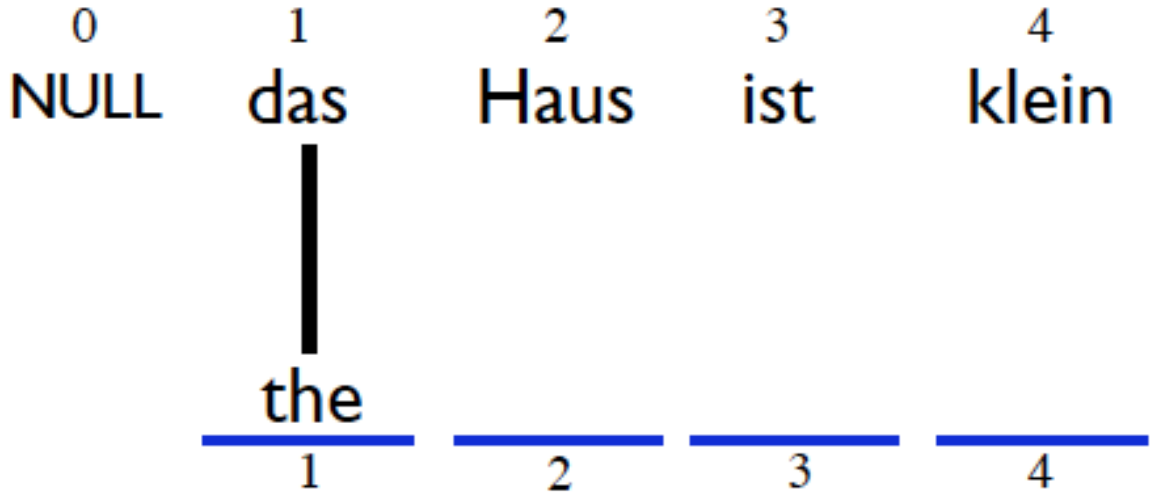
0	1	2	3	4
NULL	das	Haus	ist	klein

Start with a foreign sentence and a target length.

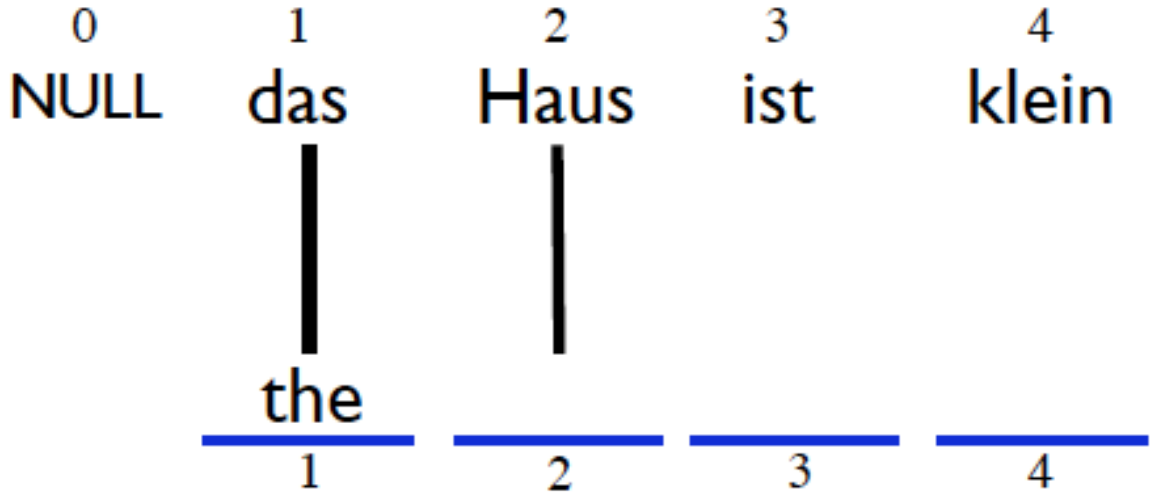
Example



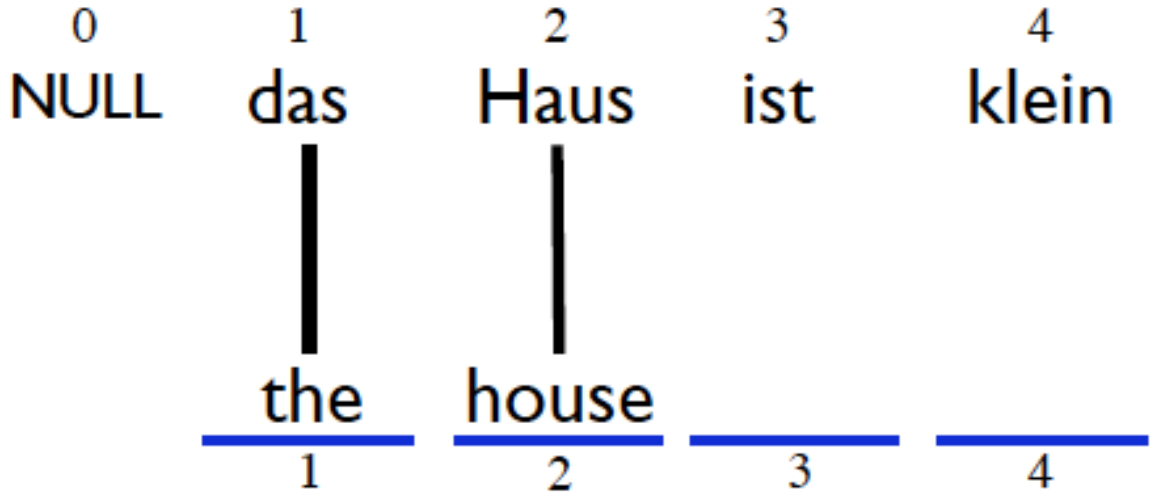
Example



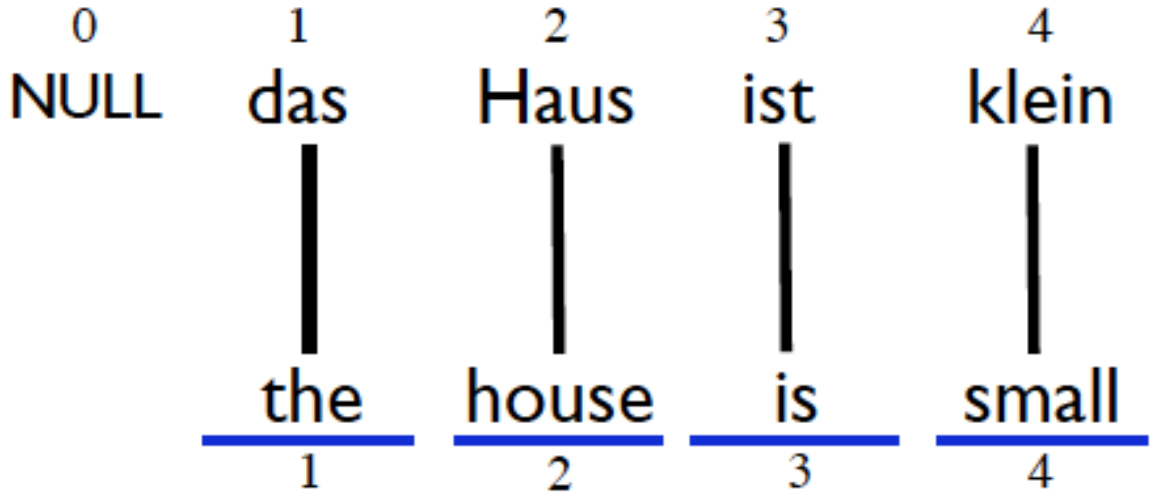
Example



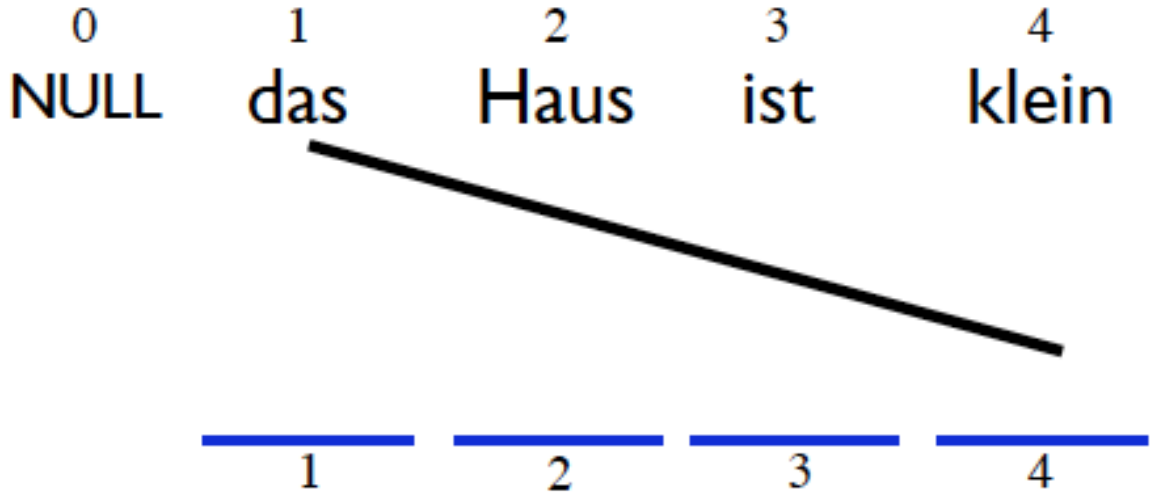
Example



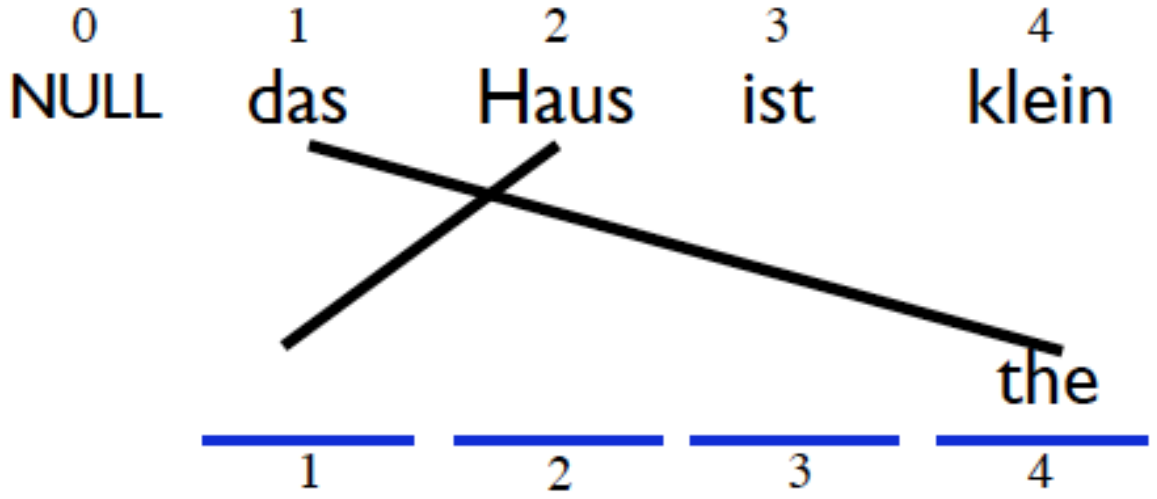
Example



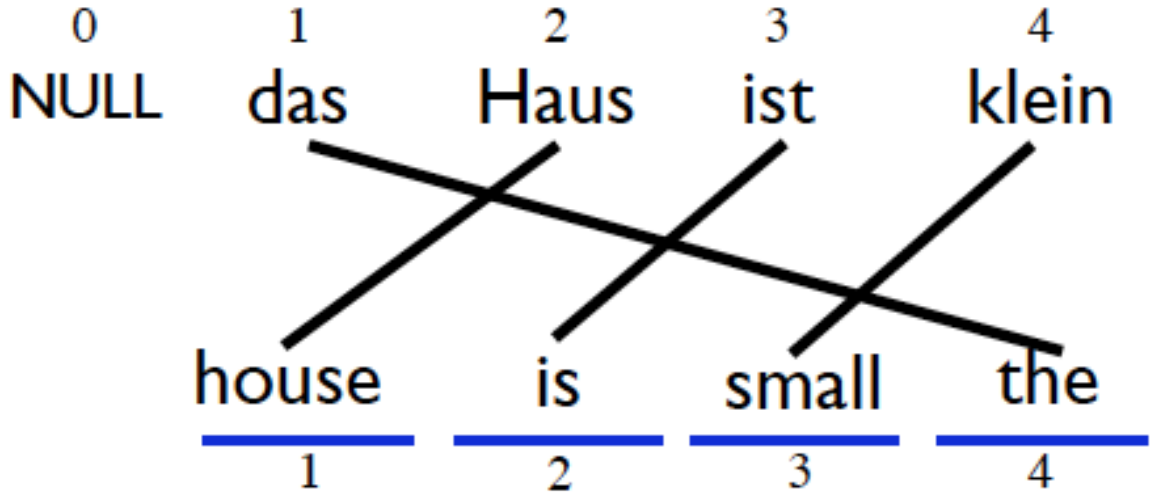
Example



Example



Example



Finding the Viterbi Alignment

$$\begin{aligned}
\mathbf{a}^* &= \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} p(\mathbf{a} \mid \mathbf{e}, \mathbf{f}) \\
&= \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} \frac{p(\mathbf{e}, \mathbf{a} \mid \mathbf{f})}{\sum_{\mathbf{a}'} p(\mathbf{e}, \mathbf{a}' \mid \mathbf{f})} \\
&= \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} p(\mathbf{e}, \mathbf{a} \mid \mathbf{f})
\end{aligned}$$

$$\begin{aligned}
a_i^* &= \arg \max_{a_i=0}^{n} \frac{1}{1+n} p(e_i \mid f_{a_i}) \\
&= \arg \max_{a_i=0}^{n} p(e_i \mid f_{a_i})
\end{aligned}$$

Historical Note

The **Viterbi algorithm** is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the **Viterbi path** – that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models.

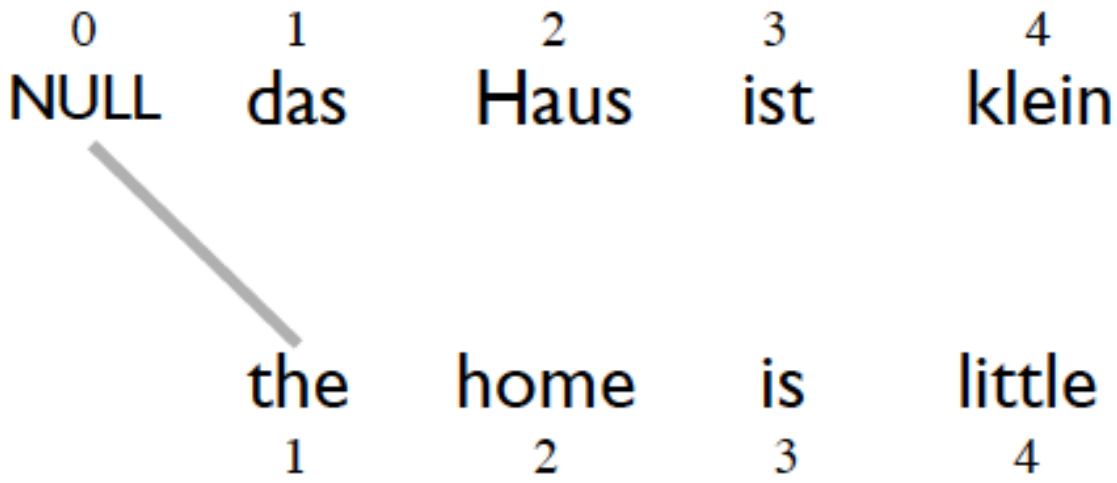
Andrew Viterbi

Professor at USC

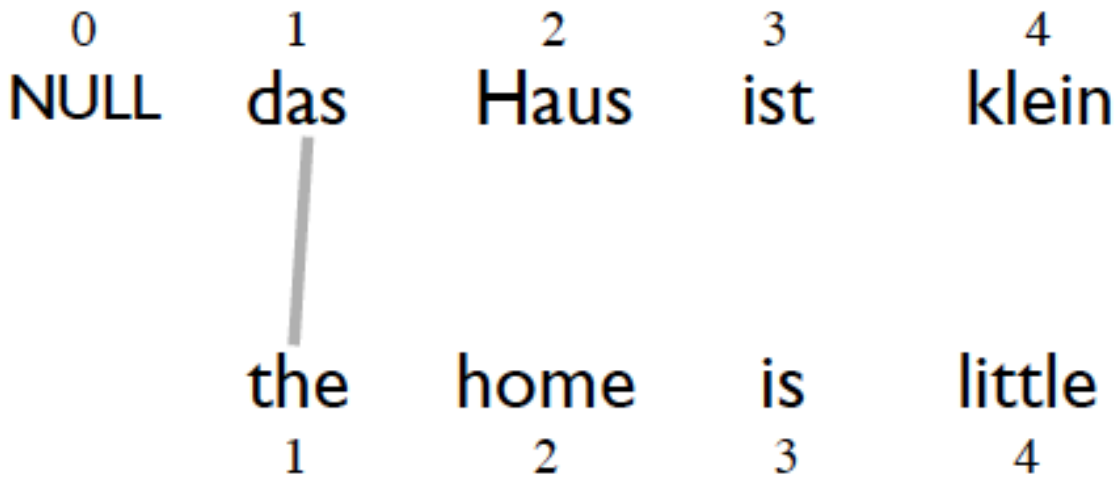
co-founder of Qualcomm

classmates with Fred Jelinek

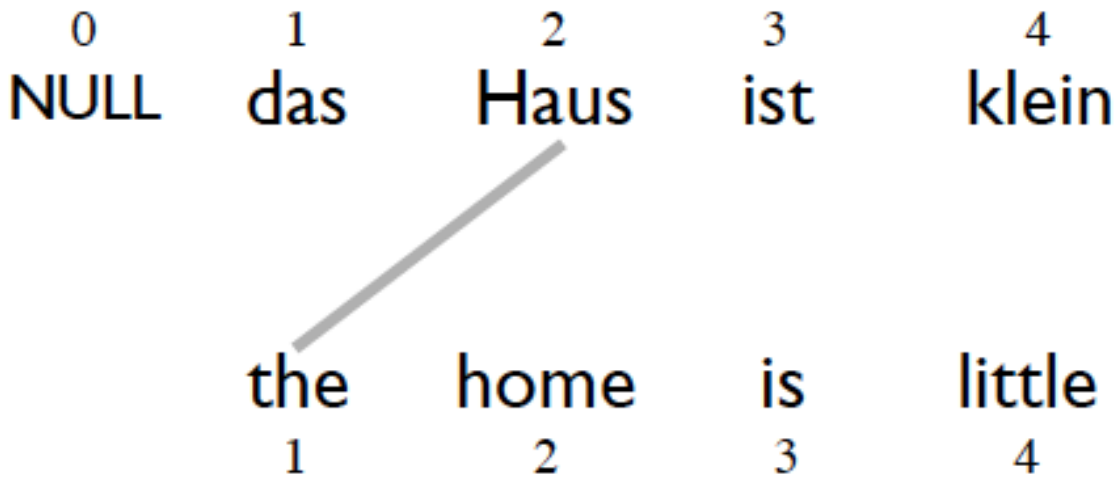
Finding the Viterbi Alignment



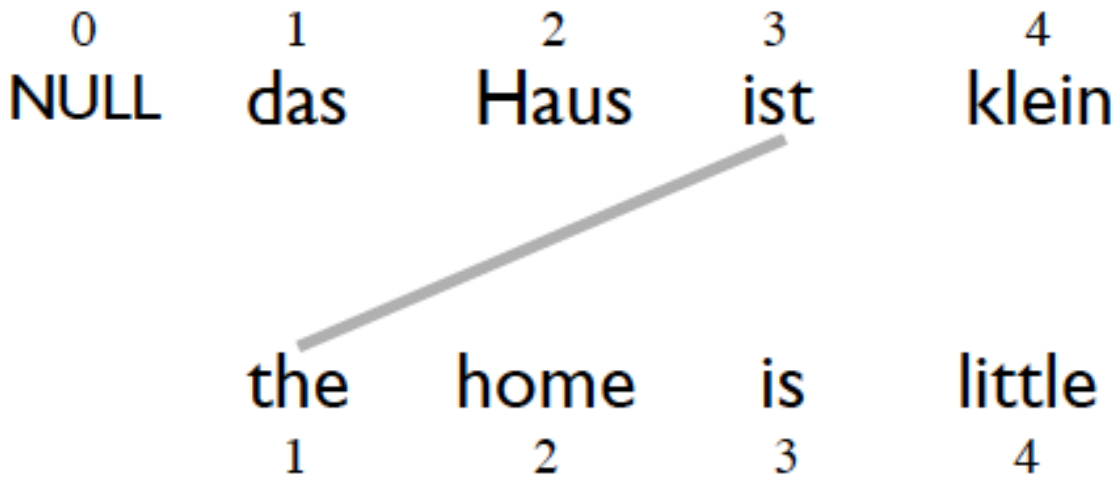
Finding the Viterbi Alignment



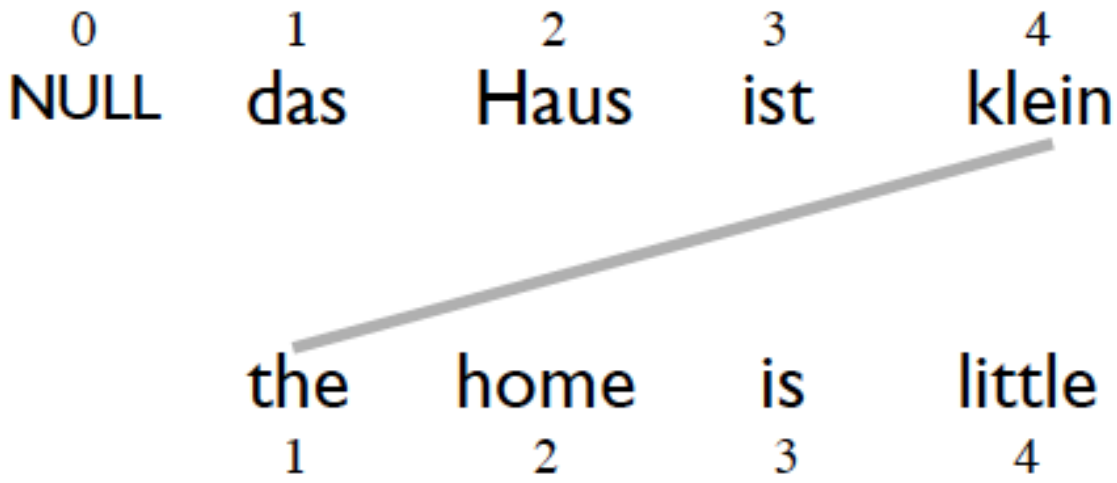
Finding the Viterbi Alignment



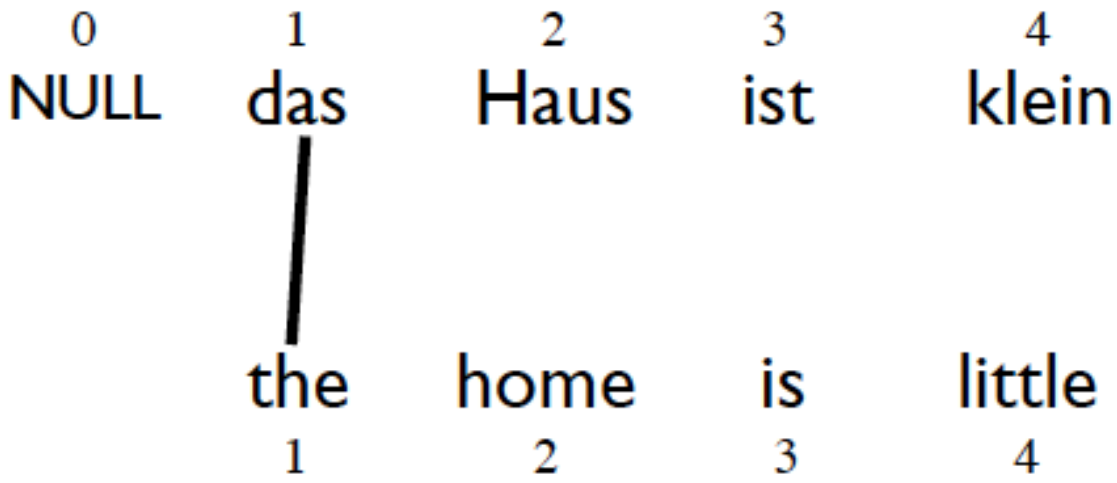
Finding the Viterbi Alignment



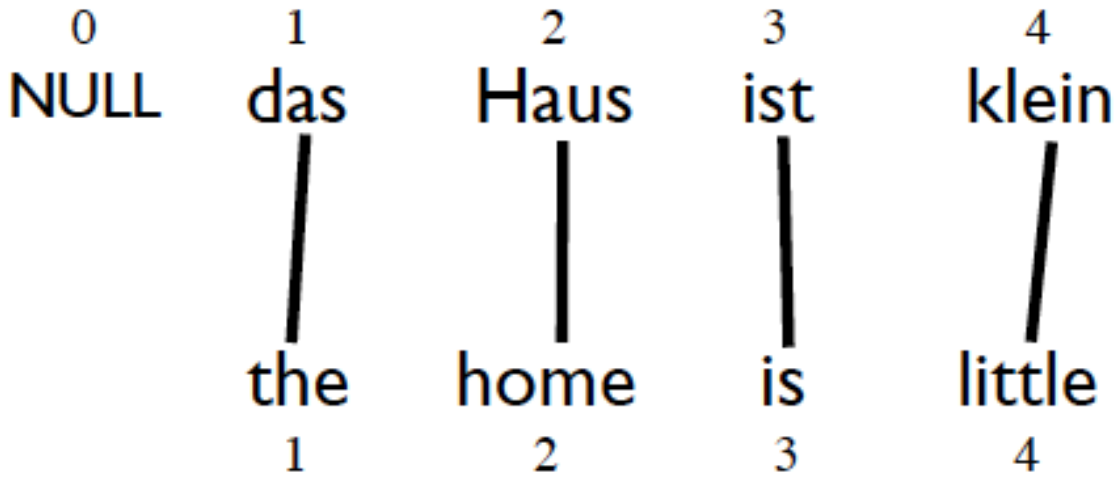
Finding the Viterbi Alignment



Finding the Viterbi Alignment

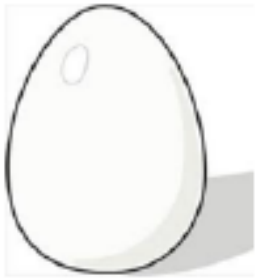


Finding the Viterbi Alignment



Learning Lexical Translation Models

- How do we learn the parameters $p(e \mid f)$
- "Chicken and egg" problem
  - If we had the alignments, we could estimate the parameters (MLE)
  - If we had parameters, we could find the most likely alignments



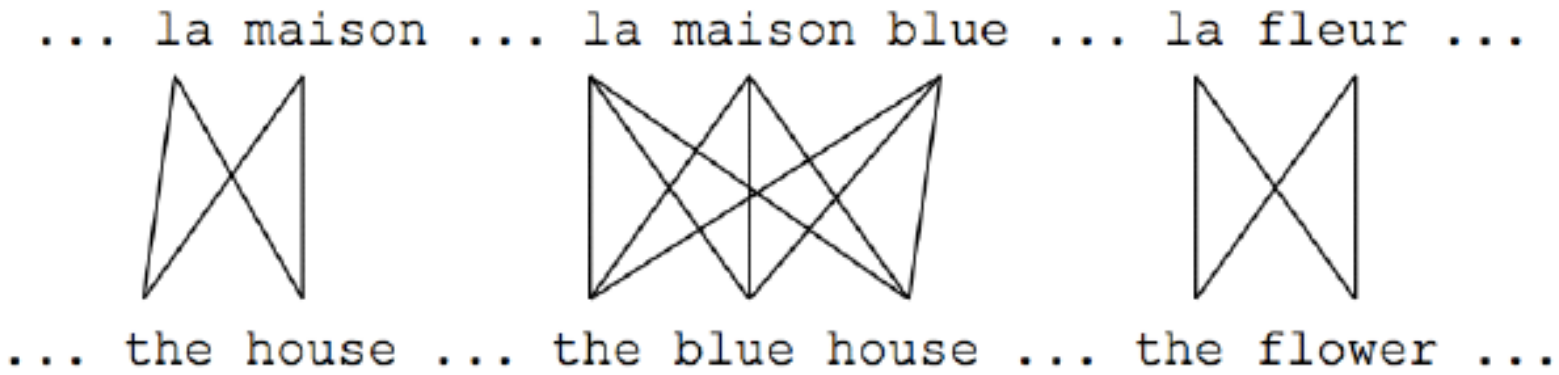
EM Algorithm

- pick some random (or uniform) parameters
- Repeat until you get bored (~ 5 iterations for lexical translation models)
  - using your current parameters, compute "expected" alignments for every target word token in the training data

$$p(a_i \mid \mathbf{e}, \mathbf{f})$$

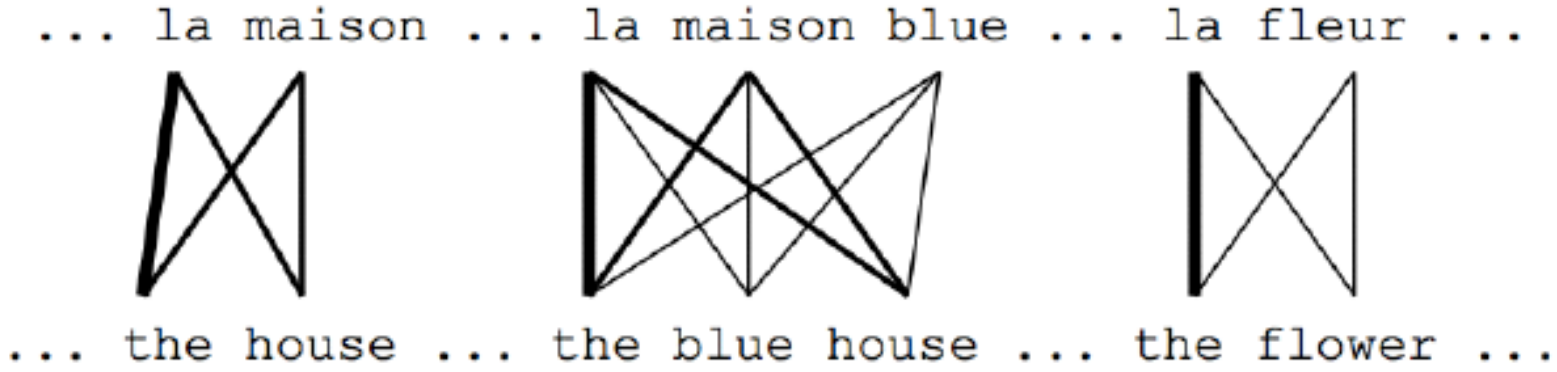
  - keep track of the expected number of times f translates into e throughout the whole corpus
  - keep track of the expected number of times that f is used as the source of any translation
  - use these expected counts as if they were "real" counts in the standard MLE equation

EM for Model 1



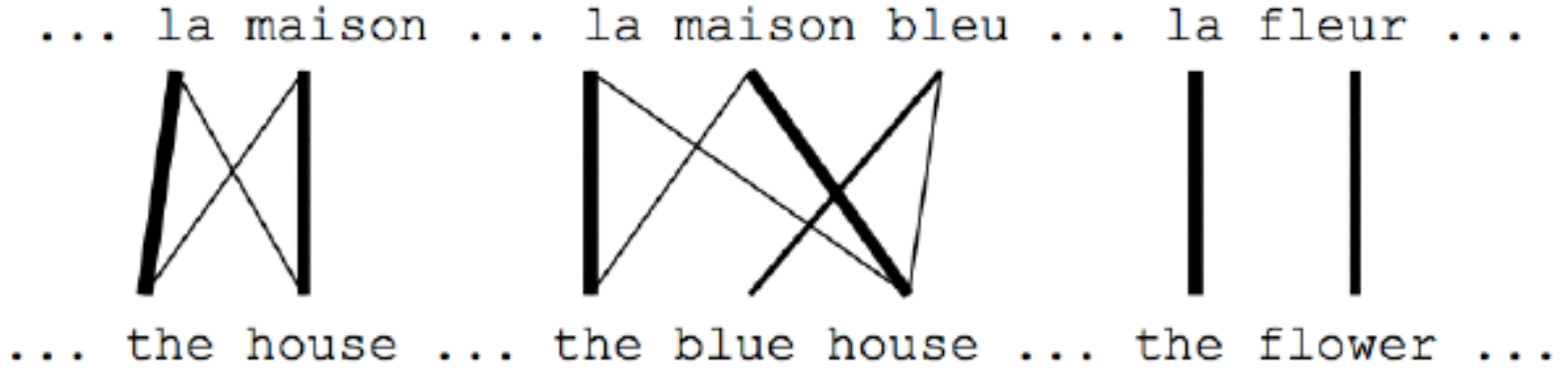
- Initial step: all alignments equally likely
- Model learns that, e.g., la is often aligned with the

EM for Model 1



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

EM for Model 1

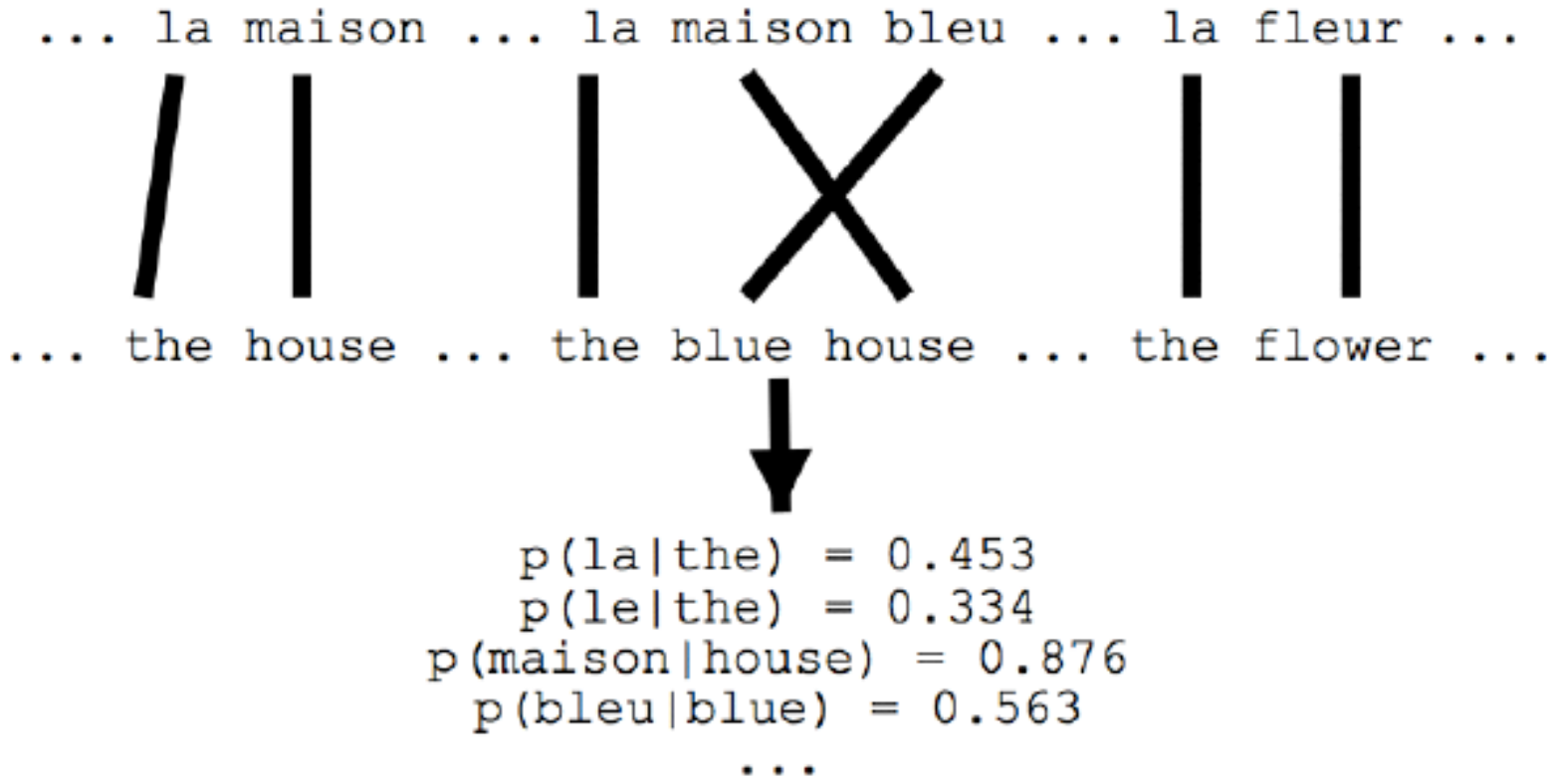


- After another iteration
- It becomes apparent that alignments, e.g., between fleur and flower are more likely (pigeon hole principle)

EM for Model 1

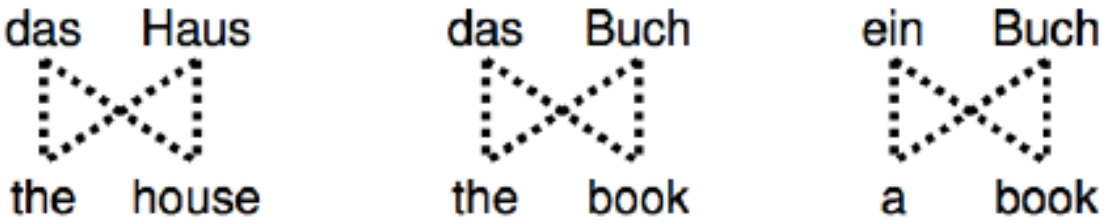
- Convergence
- Inherent hidden structure revealed by EM

EM for Model 1



- Parameter estimation from the aligned corpus

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

Can we do better?

$$p(\text{Translation}) = \sum_{\text{Alignment}} p(\text{Alignment}, \text{Translation})$$

$$= \sum_{\text{Alignment}} p(\text{Alignment}) \times p(\text{Translation} \mid \text{Alignment})$$

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

$$\prod_{i=1}^{m} p(e_i \mid f_{a_i}, f_{a_{i-1}})$$

$$\prod_{i=1}^{m} p(e_i \mid f_{a_i}, f_{a_i - 1})$$

$$\prod_{i=1}^{m} p(e_i \mid f_{a_i}, e_{i-1})$$

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

$$= \sum_{\mathbf{a} \in [0,n]^m} \underbrace{\prod_{i=1}^{m} \frac{1}{1+n}}_{p(\mathbf{a}|\mathbf{f},m)} \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

$$= \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} \frac{1}{1+n} p(e_i \mid f_{a_i})$$

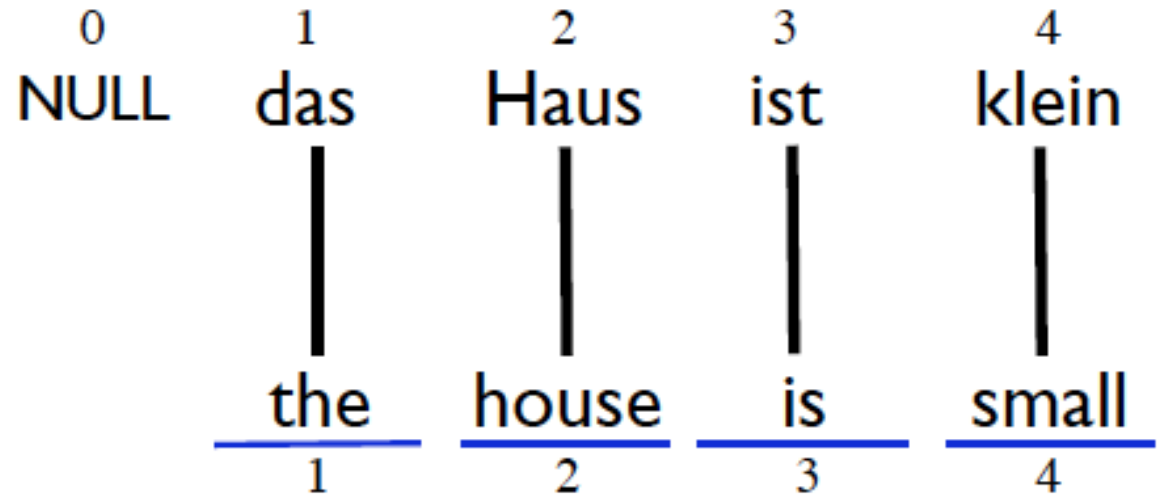
$$= \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i) \times p(e_i \mid f_{a_i})$$

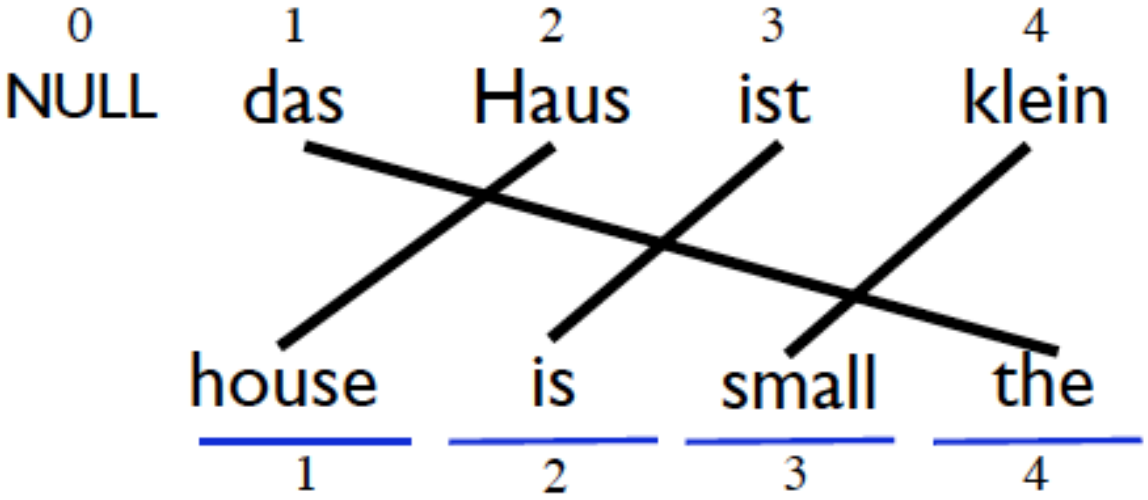
$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

$$= \sum_{\mathbf{a} \in [0,n]^m} \underbrace{\prod_{i=1}^{m} \frac{1}{1+n}}_{p(\mathbf{a} \mid \mathbf{f}, m)} \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

Can we do something better here?

$$= \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i) \times p(e_i \mid f_{a_i})$$


0
1
2
3
4
NULL
das
Haus
ist
klein
the
house
is
small
1
2
3
4

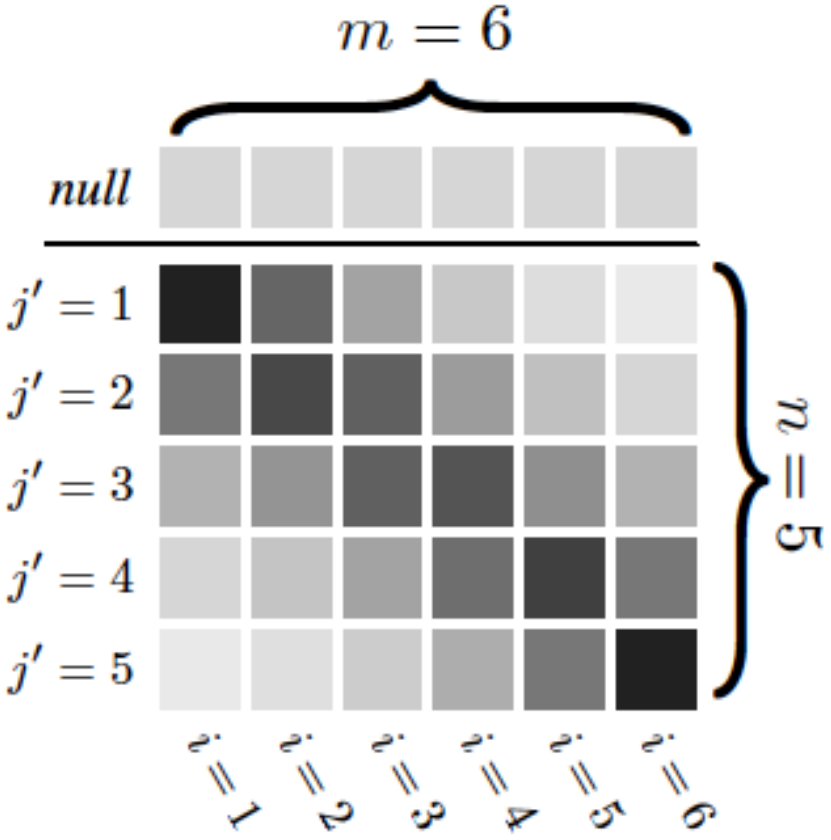

0
1
2
3
4
NULL
das
Haus
ist
klein
house
is
small
the
1
2
3
4

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i) \times p(e_i \mid f_{a_i})$$

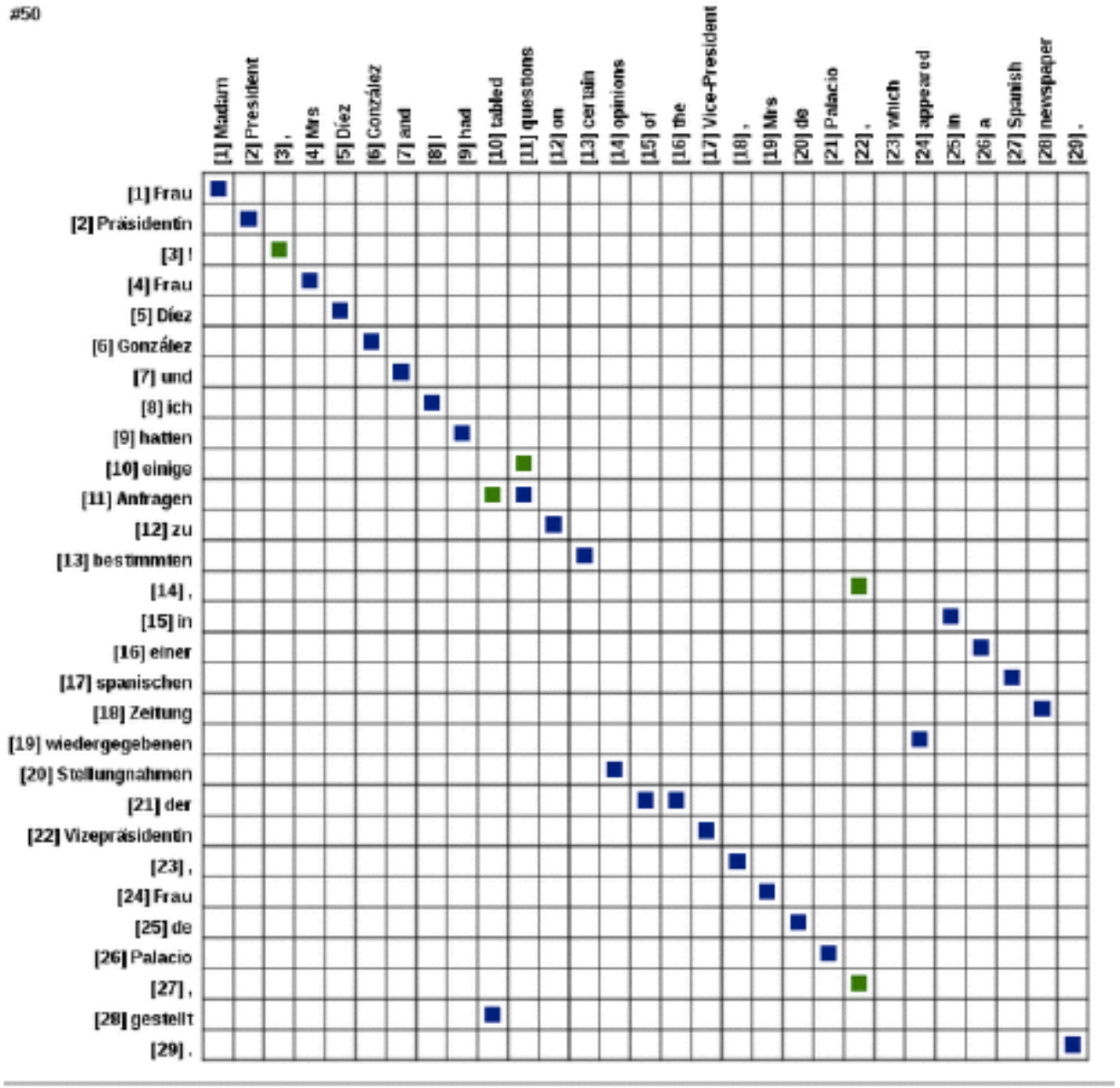
Model 2 $$= \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i \mid i, m, n) \times p(e_i \mid f_{a_i})$$

Model 2 $= \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i \mid i, m, n) \times p(e_i \mid f_{a_i})$

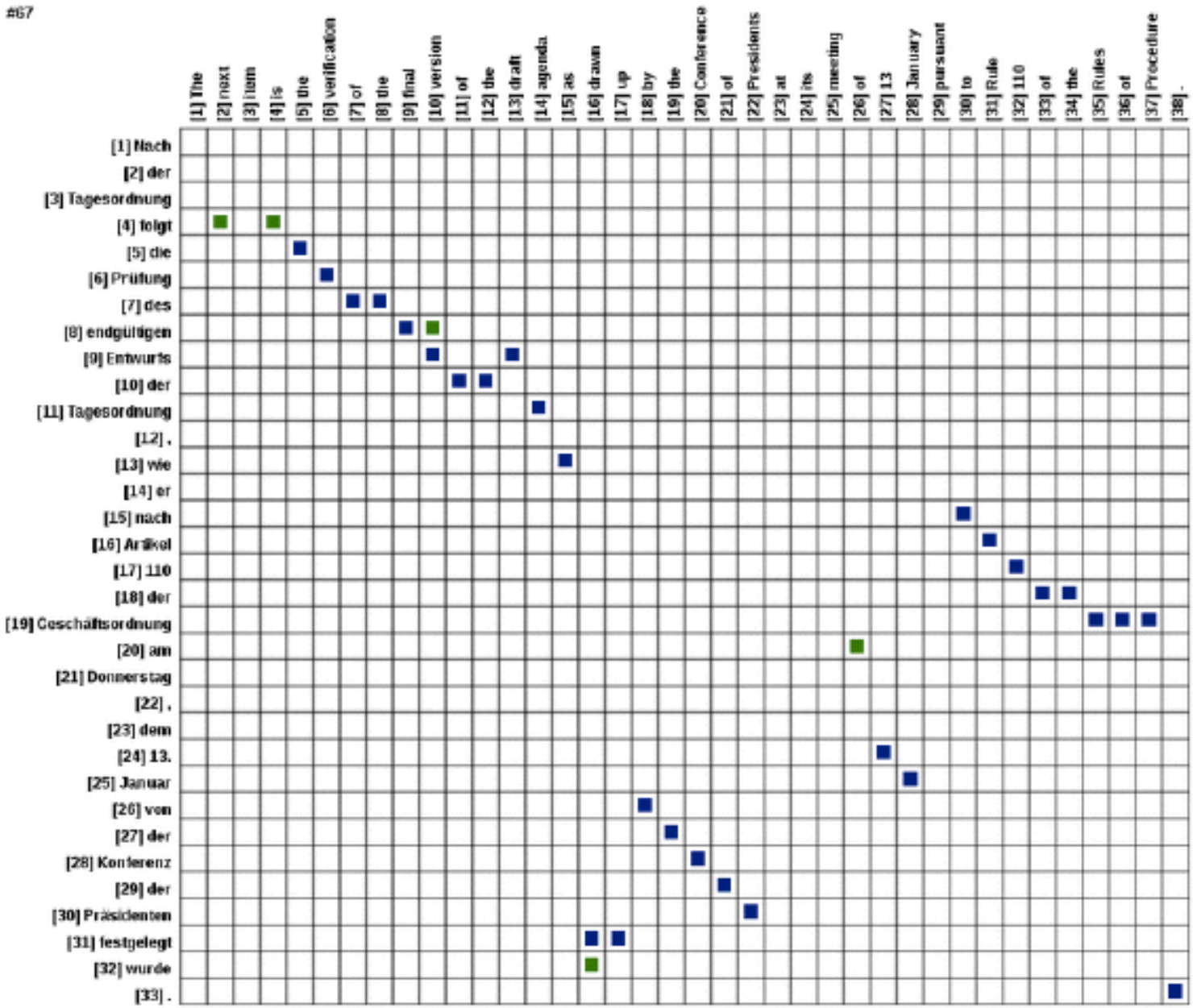
How much do we know when we only know the source & target lengths and the current position?



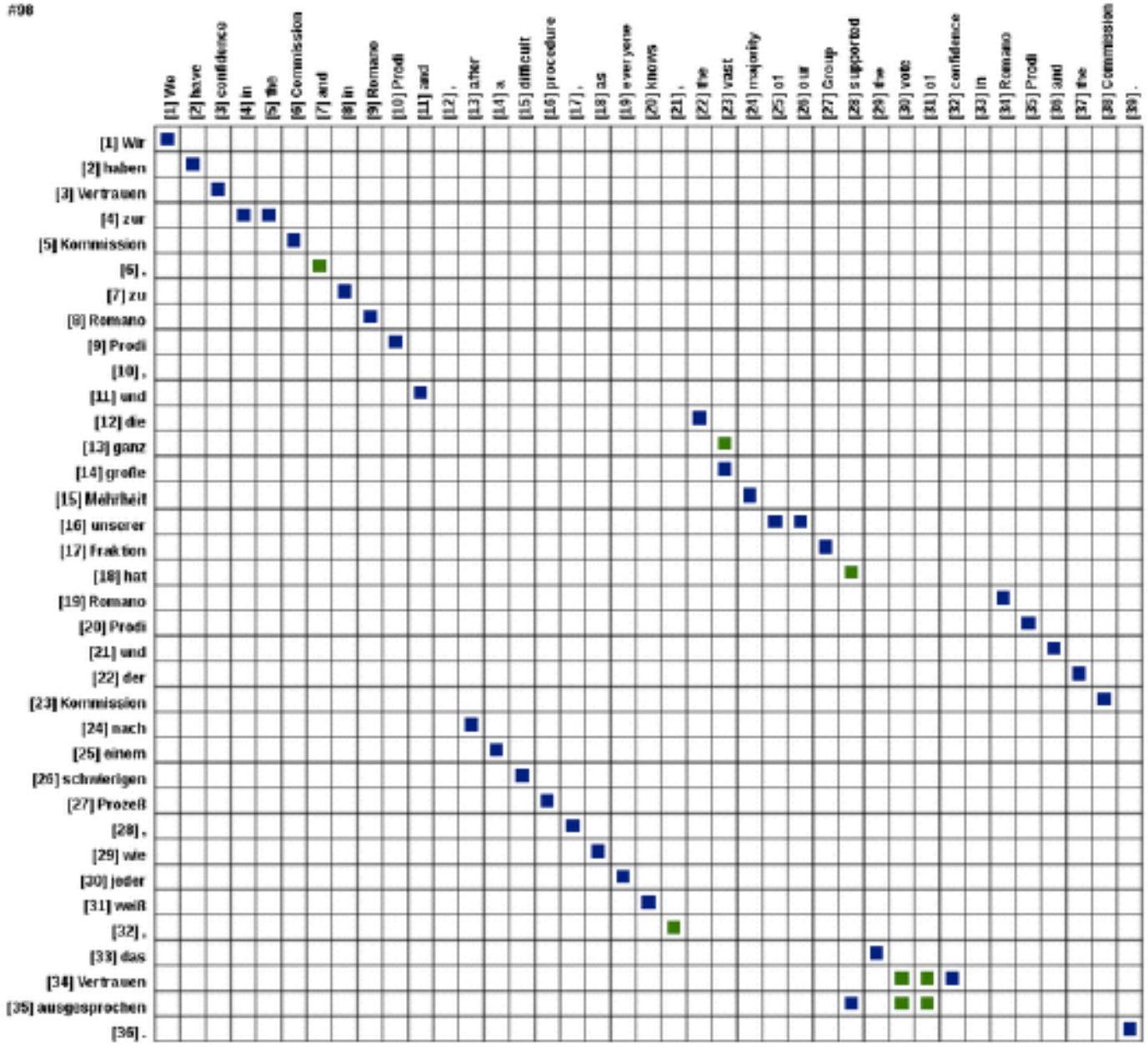
#50

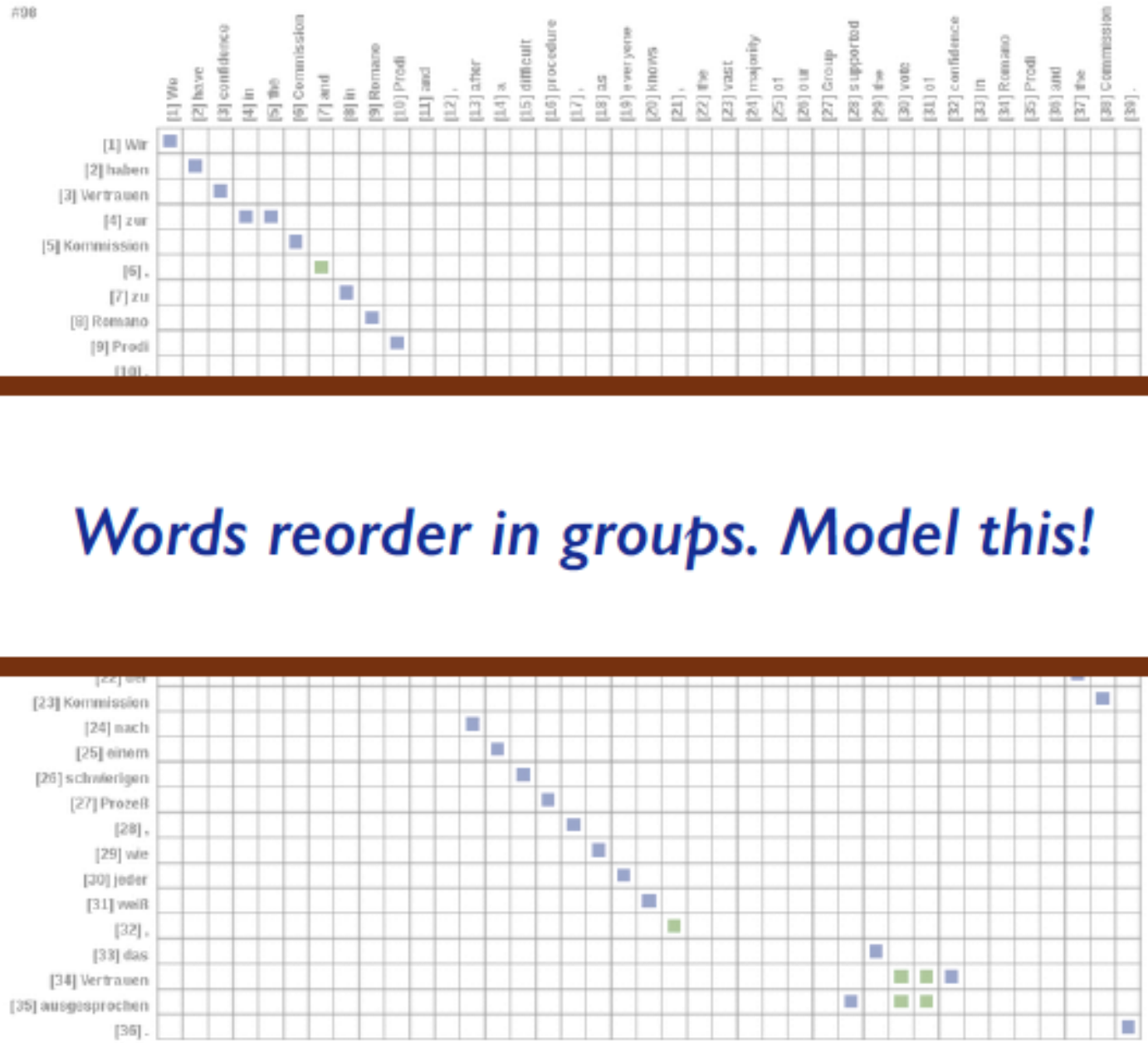

[1] Madam
[2] President
[3] ,
[4] Mrs
[5] Díez
[6] González
[7] and
[8] I
[9] had
[10] tabled
[11] questions
[12] on
[13] certain
[14] opinions
[15] of
[16] the
[17] Vice-President
[18] ,
[19] Mrs
[20] de
[21] Palacio
[22] ,
[23] which
[24] appeared
[25] in
[26] a
[27] Spanish
[28] newspaper
[29] .
[1] Frau
[2] Präsidentin
[3] !
[4] Frau
[5] Díez
[6] González
[7] und
[8] ich
[9] hatten
[10] einige
[11] Anfragen
[12] zu
[13] bestimmten
[14] ,
[15] in
[16] einer
[17] spanischen
[18] Zeitung
[19] wiedergegebenen
[20] Stellungnahmen
[21] der
[22] Vizepräsidentin
[23] ,
[24] Frau
[25] de
[26] Palacio
[27] ,
[28] gestellt
[29] .

#67


[1] The
[2] next
[3] item
[4] is
[5] the
[6] verification
[7] of
[8] the
[9] final
[10] version
[11] of
[12] the
[13] draft
[14] agenda
[15] as
[16] drawn
[17] up
[18] by
[19] the
[20] Conference
[21] of
[22] Presidents
[23] at
[24] its
[25] meeting
[26] of
[27] 13
[28] January
[29] pursuant
[30] to
[31] Rule
[32] 110
[33] of
[34] the
[35] Rules
[36] of
[37] Procedure
[38] .
[1] Nach
[2] der
[3] Tagesordnung
[4] folgt
[5] die
[6] Prüfung
[7] des
[8] endgültigen
[9] Entwurfs
[10] der
[11] Tagesordnung
[12] ,
[13] wie
[14] er
[15] nach
[16] Artikel
[17] 110
[18] der
[19] Geschäftsordnung
[20] am
[21] Donnerstag
[22] ,
[23] dem
[24] 13.
[25] Januar
[26] von
[27] der
[28] Konferenz
[29] der
[30] Präsidenten
[31] festgelegt
[32] wurde
[33] .

#98


[1] Wir
[2] haben
[3] Vertrauen
[4] zur
[5] Kommission
[6] .
[7] zu
[8] Romano
[9] Prodi
[10] ,
[11] und
[12] die
[13] ganz
[14] große
[15] Mehrheit
[16] unserer
[17] Fraktion
[18] hat
[19] Romano
[20] Prodi
[21] und
[22] der
[23] Kommission
[24] nach
[25] einem
[26] schwierigen
[27] Prozeß
[28] ,
[29] wie
[30] jeder
[31] weiß
[32] ,
[33] das
[34] Vertrauen
[35] ausgesprochen
[36] .



Words reorder in groups. Model this!

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i) \times p(e_i \mid f_{a_i})$$

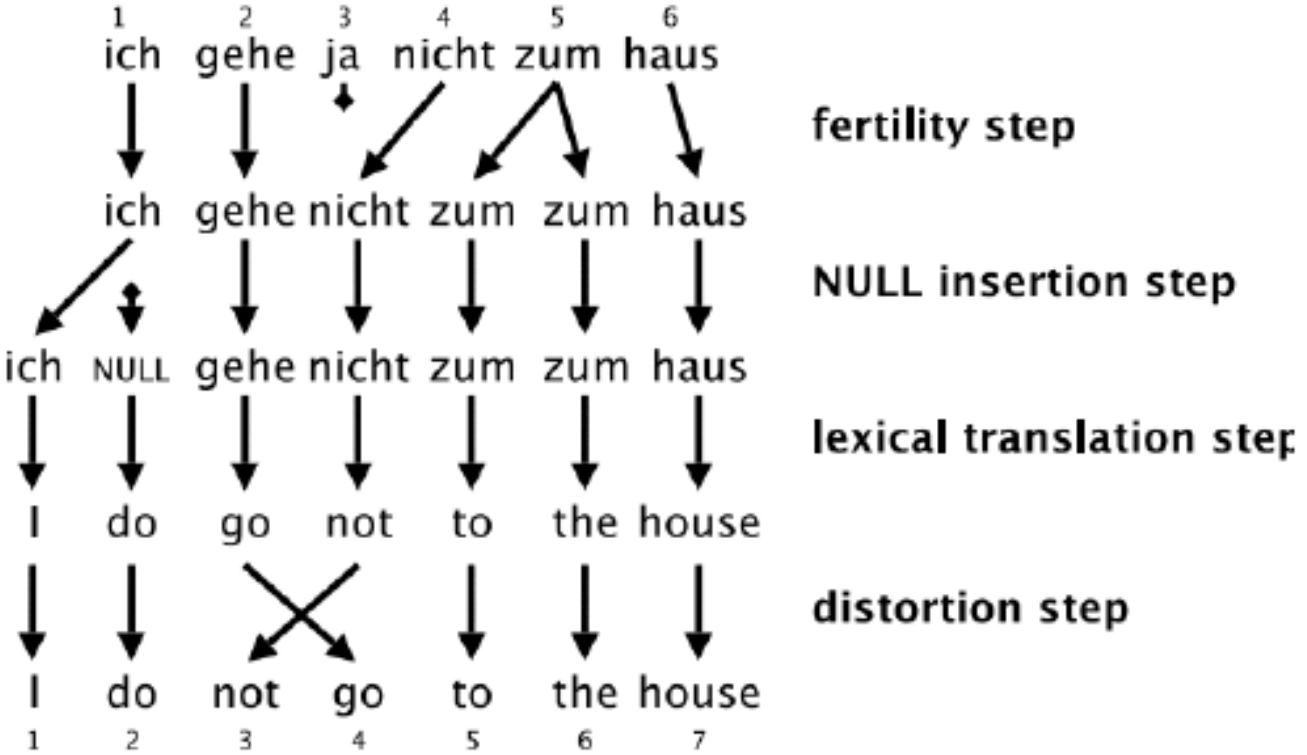
$$\textbf{Model 2} \quad = \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i \mid i, m, n) \times p(e_i \mid f_{a_i})$$

$$\textbf{HMM} \quad = \sum_{\mathbf{a} \in [0,n]^m} \prod_{i=1}^{m} p(a_i \mid a_{i-1}) \times p(e_i \mid f_{a_i})$$

Fertility Models

- The models we have considered so far have been efficient
- This efficiency has come at a modeling cost:
  - What is to stop the model from "translating" a word 0, 1, 2, or 100 times?
- We introduce *fertility models* to deal with this

IBM Model 3

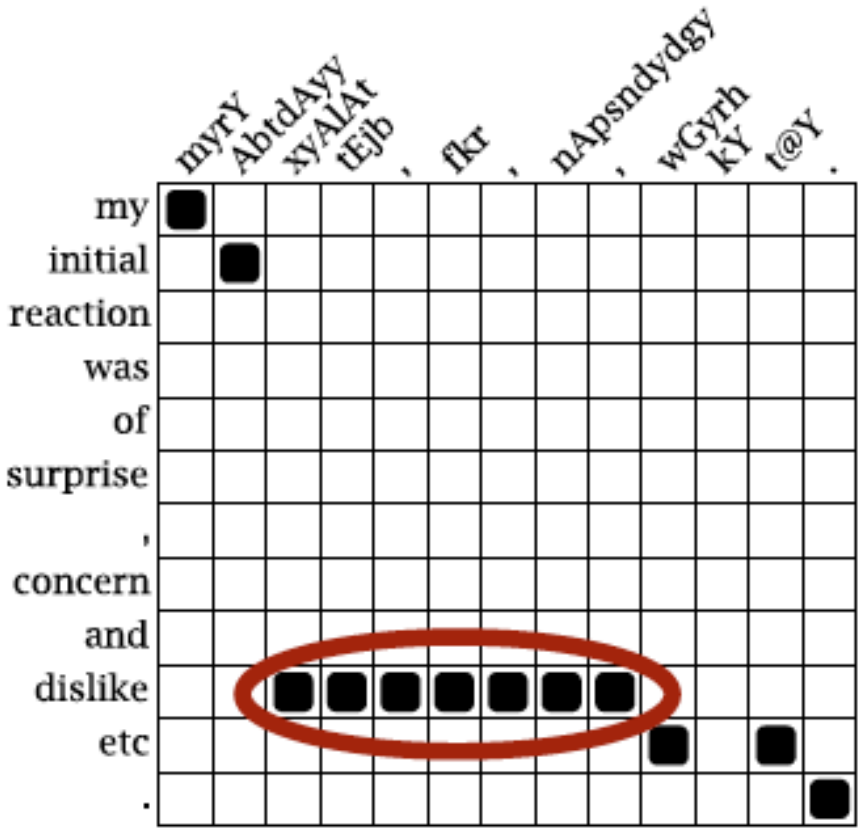

1 2 3 4 5 6
ich gehe ja nicht zum haus
ich gehe nicht zum zum haus
ich NULL gehe nicht zum zum haus
I do go not to the house
I do not go to the house
1 2 3 4 5 6 7
fertility step
NULL insertion step
lexical translation step
distortion step

Fertility

$$p(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a} \in [0,n]^m} p(\mathbf{a} \mid \mathbf{f}, m) \times \prod_{i=1}^{m} p(e_i \mid f_{a_i})$$

- Fertility models mean that we can no longer exploit conditional independencies to write $p(\mathbf{a} \mid \mathbf{f}, m)$ as a series of local alignment decisions.
- *How do we compute the statistics required for EM training?*

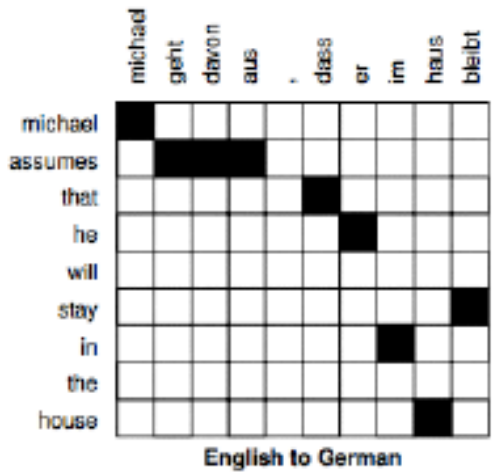
Pitfalls of Conditional Models



IBM Model 4 alignment

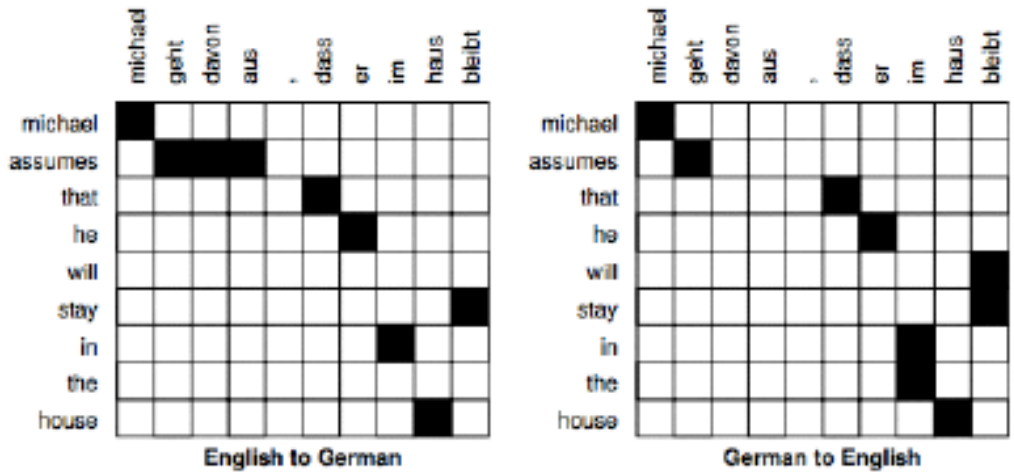
A few tricks...

p(f|e)



A few tricks...

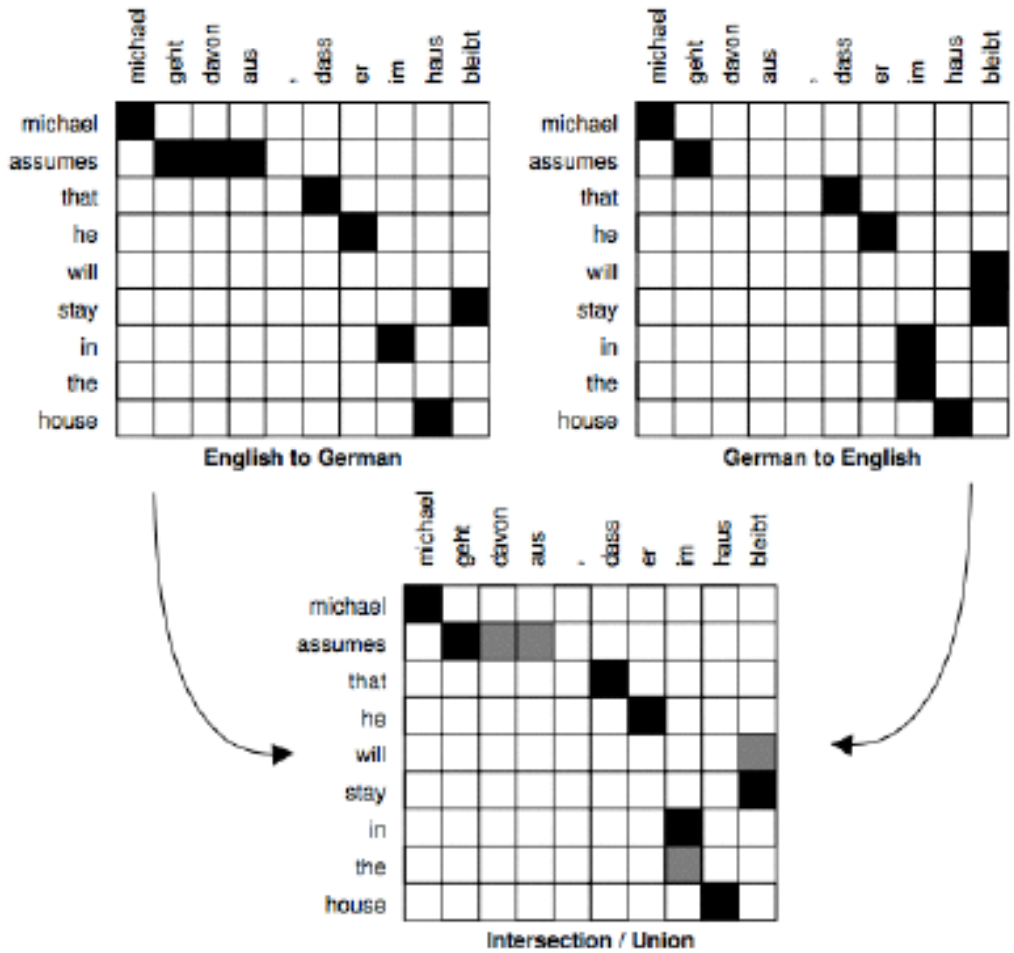
p(f|e)



p(e|f)

A few tricks...

p(f|e)



p(e|f)

Phrase-based Translation Models

Translational Equivalence

*Er hat die Prüfung **bestanden**, jedoch nur knapp*

He insisted on the test, but just barely.

He passed the test, but just barely.

How do lexical translation models deal with contextual information?

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \overline{\mathbf{e}}, \overline{\mathbf{f}} \rangle \in \mathbf{a}} p(\overline{\mathbf{f}} \mid \overline{\mathbf{e}})$$

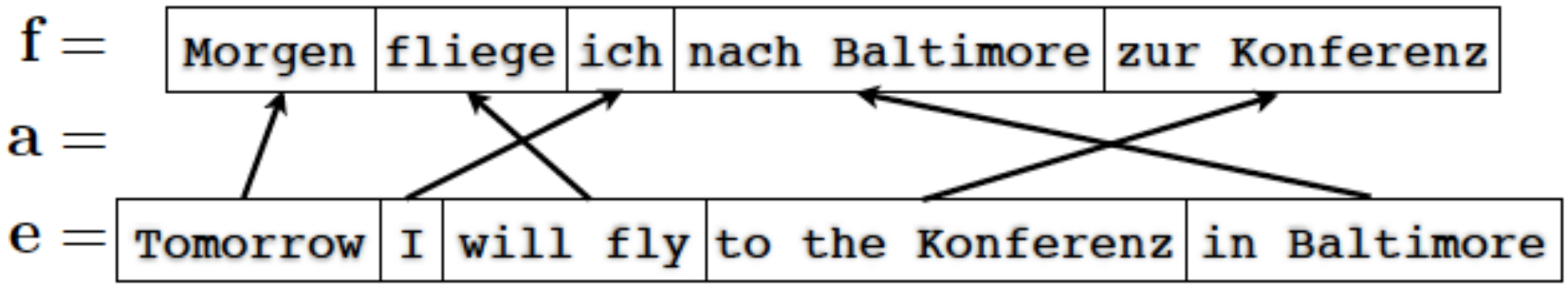
$\mathbf{f} =$ `Morgen fliege ich nach Baltimore zur Konferenz`

$\mathbf{e} =$ `Tomorrow I will fly to the Konferenz in Baltimore`

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

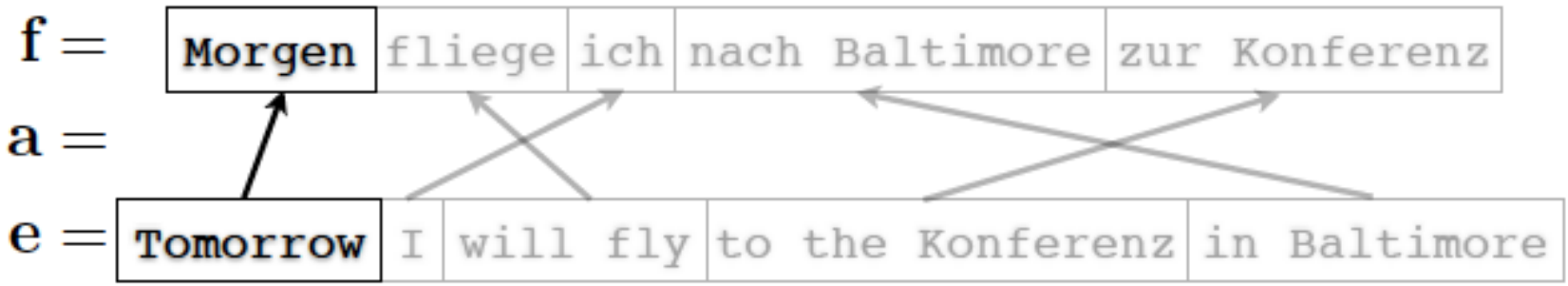
$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$



Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$

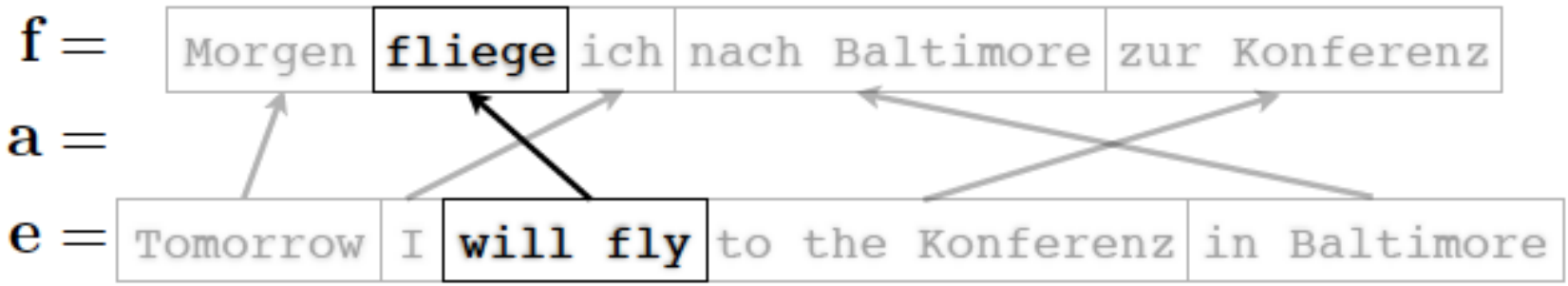


p(Morgen|Tomorrow)

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$

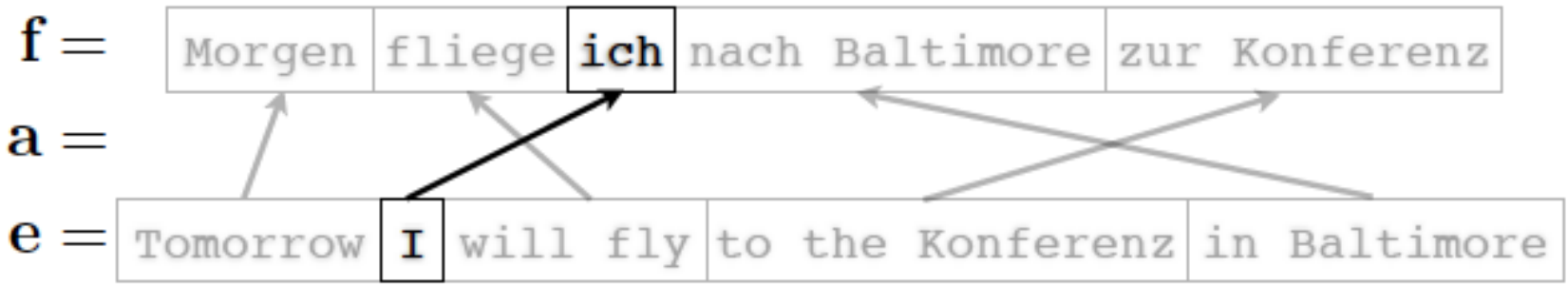


p(Morgen|Tomorrow) x p(fliege|will fly)

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$

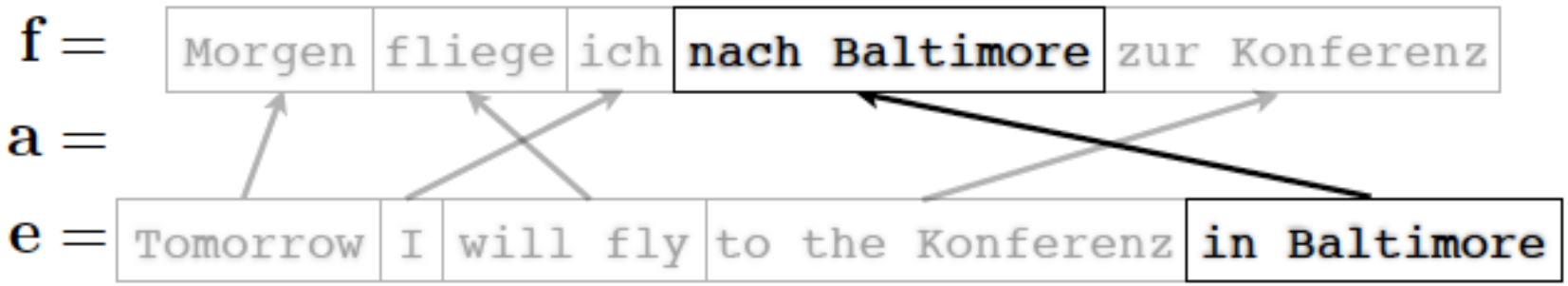


p(Morgen|Tomorrow) x p(fliege|will fly) x p(ich|I)

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$



p(Morgen|Tomorrow) x p(fliege|will fly) x p(ich|I) x ...

Translation model

- With a **latent variable**, we introduce a decomposition into **phrases** which translate **independently**:

$$p(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$

Marginalize to get p(f|e):

$$p(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} p(\mathbf{a}) \prod_{\langle \bar{\mathbf{e}}, \bar{\mathbf{f}} \rangle \in \mathbf{a}} p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$$

Phrases

- Contiguous strings of words
- Phrases are not necessarily syntactic constituents
- Usually have maximum limits
- Phrases subsume words (words are phrases)

Phrase Tables

$\bar{\mathbf{f}}$	$\bar{\mathbf{e}}$	$p(\bar{\mathbf{f}} \mid \bar{\mathbf{e}})$
das Thema	the issue	0.41
	the point	0.72
	the subject	0.47
	the thema	0.99
es gibt	there is	0.96
	there are	0.72
morgen	tomorrow	0.9
fliege ich	will I fly	0.63
	will fly	0.17
	I will fly	0.13

p(a)

- Two responsibilities
  - **Divide the source sentence into phrases**
    - Standard approach: uniform distribution over all possible segmentations
    - How many segmentations are there?
  - **Reorder the phrases**
    - Standard approach: Markov model on phrases (parameterized with log-linear model)

Learning Phrases

- Latent segmentation variable
- Latent phrasal inventory
- Parallel data
  - EM?

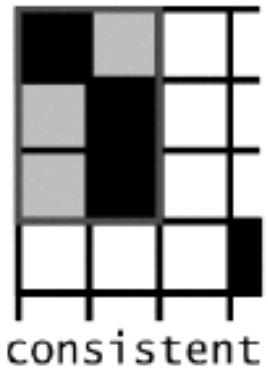
Computational problem: summing over all segmentations and alignments is #P-complete $p(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} p(\mathbf{a}) \prod_{\langle \overline{\mathbf{e}}, \overline{\mathbf{f}} \rangle \in \mathbf{a}} p(\overline{\mathbf{f}} \mid \overline{\mathbf{e}})$

Modeling problem: MLE has a degenerate solution.

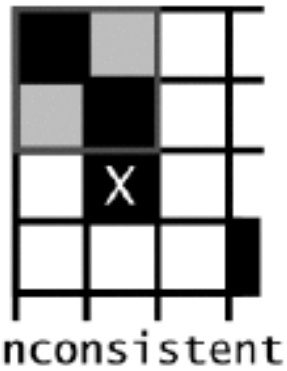
Learning Phrases

- Three stages
  - word alignment
  - extraction of phrases
  - estimation of phrase probabilities

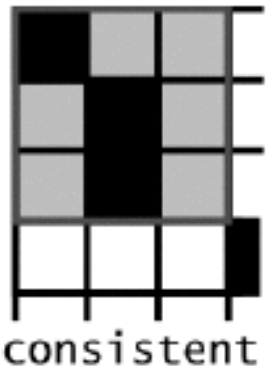
Consistent Phrases



ok



violated
one alignment
point outside



ok
unaligned
word is fine

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

Scoring Phrase Pairs

- Phrase pair extraction: collect all phrase pairs from the data
- Phrase pair scoring: assign probabilities to phrase translations
- Score by relative frequency:

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

Decoding in Phrase-based Models

Translation Process

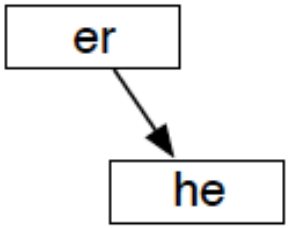
- Task: translate this sentence from German into English

er **geht** **ja** **nicht** **nach** **hause**

Translation Process

- Task: translate this sentence from German into English

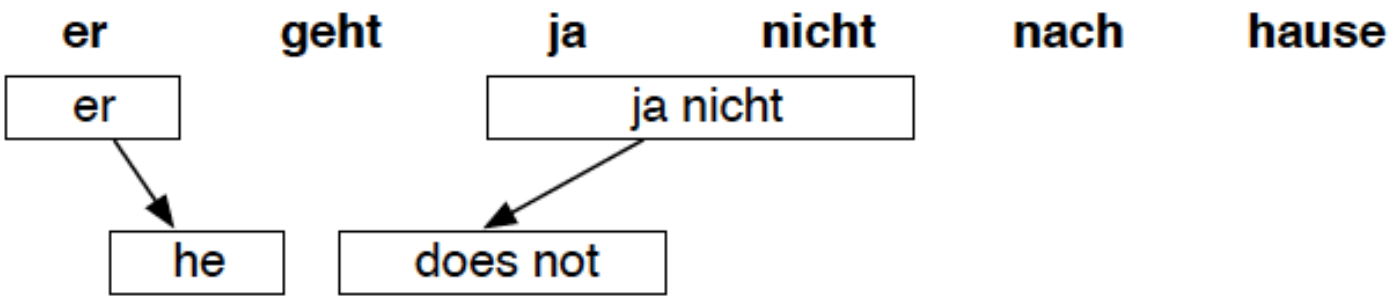
er **geht** **ja** **nicht** **nach** **hause**



- Pick phrase in input, translate

Translation Process

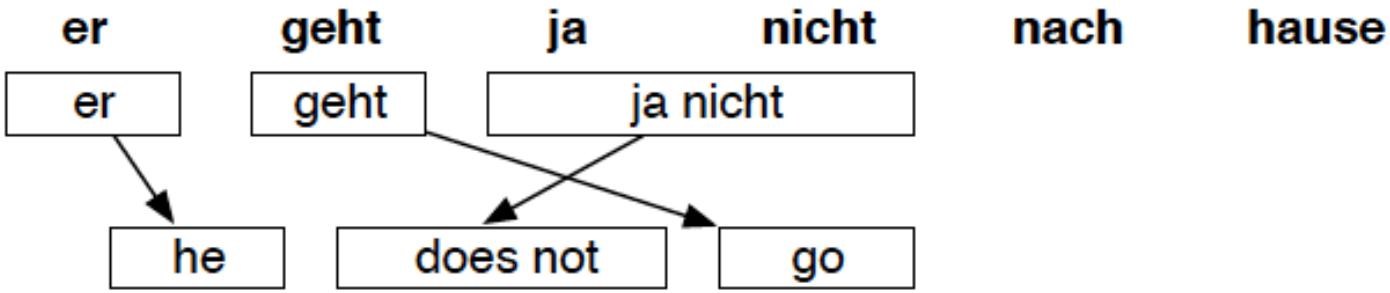
- Task: translate this sentence from German into English



- Pick phrase in input, translate
  - it is allowed to pick words out of sequence reordering
  - phrases may have multiple words: many-to-many translation

Translation Process

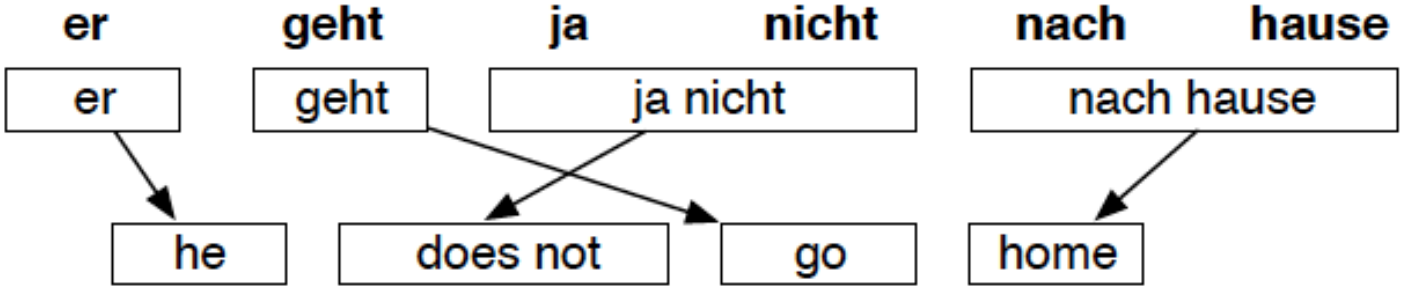
- Task: translate this sentence from German into English



- Pick phrase in input, translate

Translation Process

- Task: translate this sentence from German into English



- Pick phrase in input, translate

Computing Translation Probability

- Probabilistic model for phrase-based translation:

$$\mathbf{e}_{\text{best}} = \text{argmax}_{\mathbf{e}} \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)\ d(start_i - end_{i-1} - 1)\ p_{\text{LM}}(\mathbf{e})$$

- Score is computed incrementally for each partial hypothesis

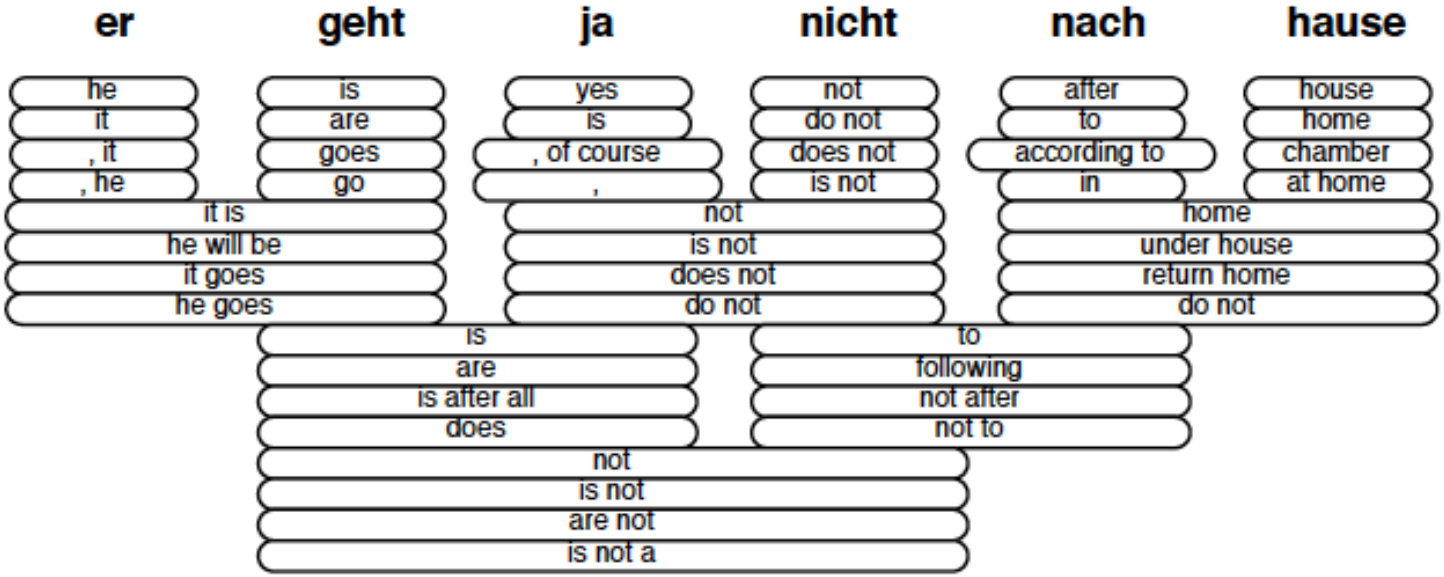
- Components

  Phrase translation Picking phrase $\bar{f}_i$ to be translated as a phrase $\bar{e}_i$
  → look up score $\phi(\bar{f}_i|\bar{e}_i)$ from phrase translation table

  Reordering Previous phrase ended in end_{i-1}, current phrase starts at $start_i$
  → compute $d(start_i - end_{i-1} - 1)$

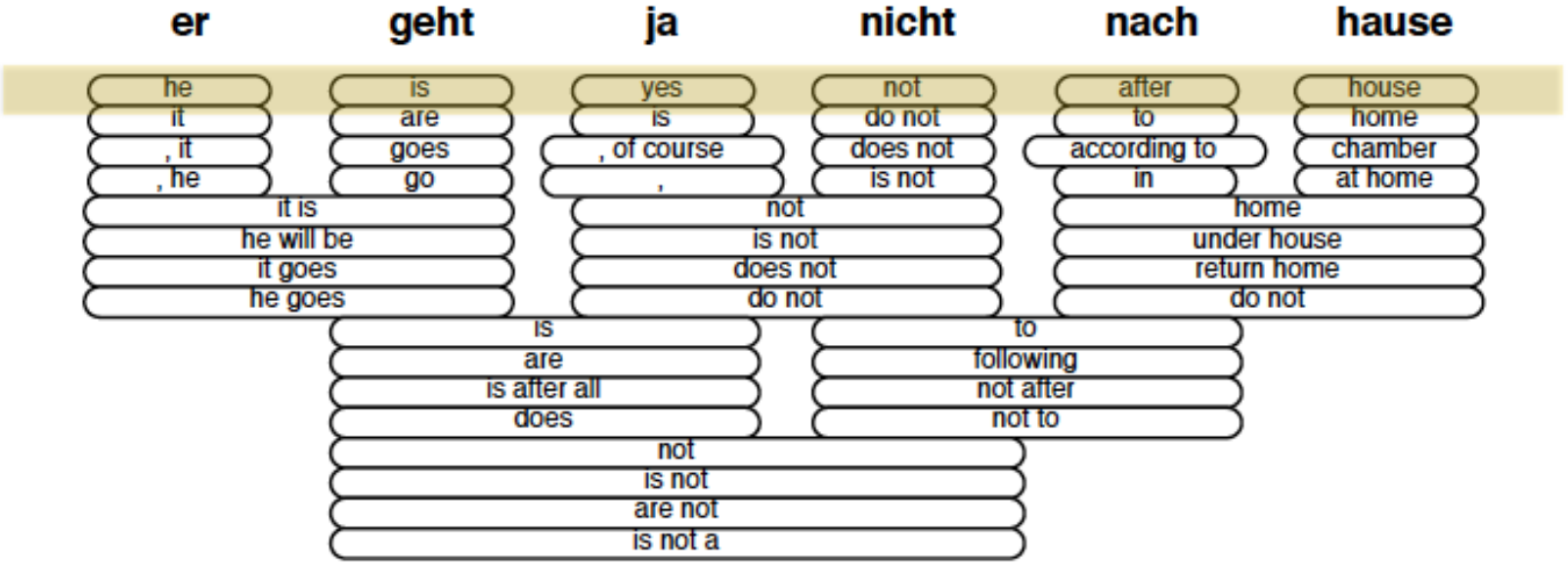
  Language model For n-gram model, need to keep track of last $n-1$ words
  → compute score $p_{\text{LM}}(w_i|w_{i-(n-1)}, ..., w_{i-1})$ for added words w_i

Translation Options



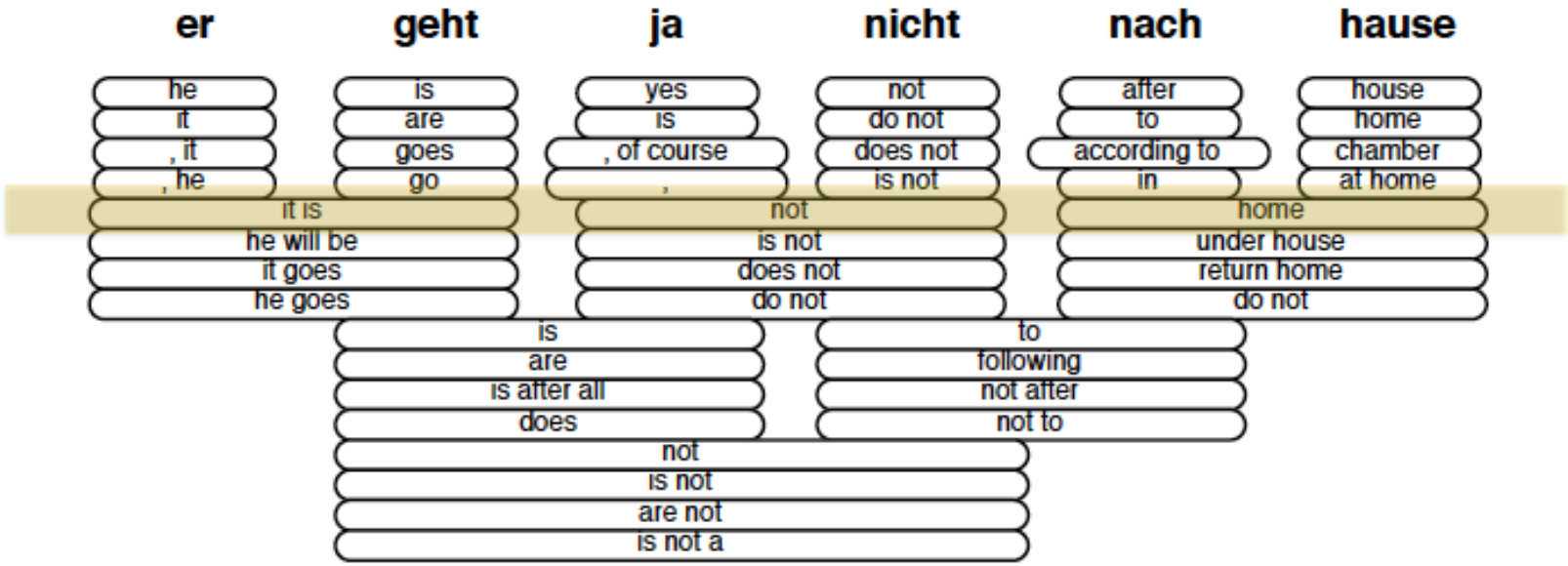
- Many translation options to choose from
  - in Europarl phrase table: 2727 matching phrase pairs for this sentence
  - by pruning to the top 20 per phrase, 202 translation options remain

Translation Options



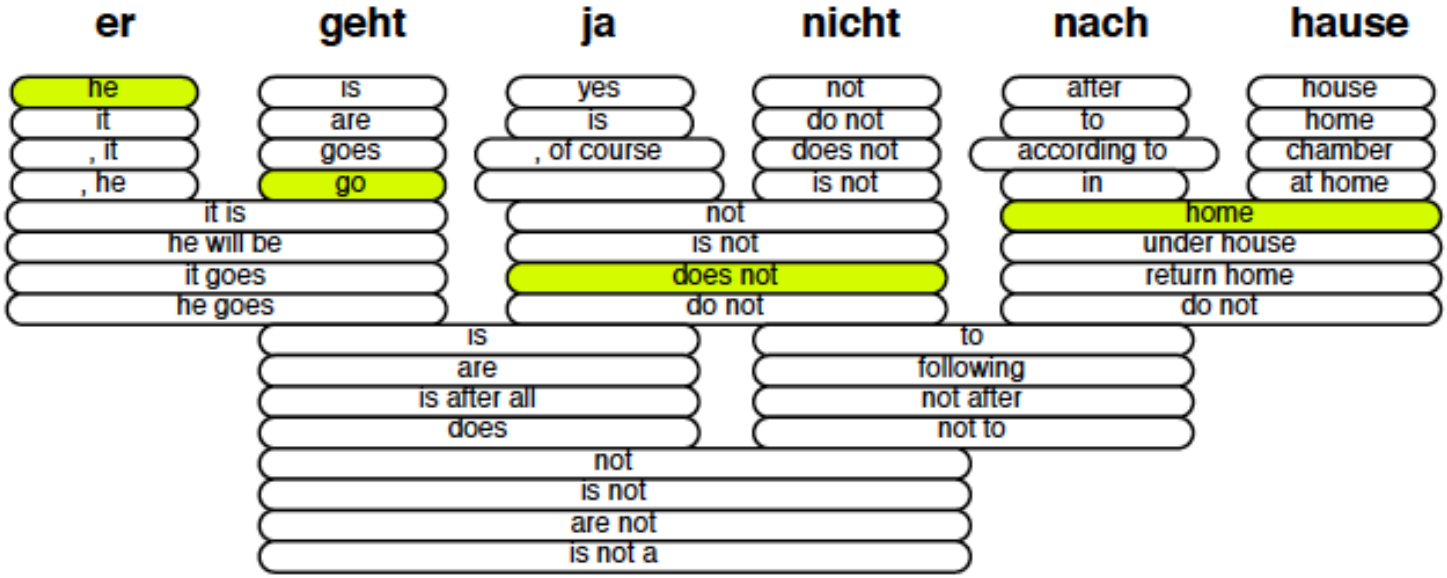
- Many translation options to choose from
  - in Europarl phrase table: 2727 matching phrase pairs for this sentence
  - by pruning to the top 20 per phrase, 202 translation options remain

Translation Options



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

Translation Options



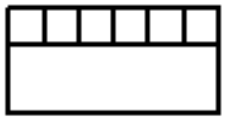
- The machine translation decoder does not know the right answer
  - picking the right translation options
  - arranging them in the right order

→ Search problem solved by heuristic beam search

Decoding: Precompute Translation Options

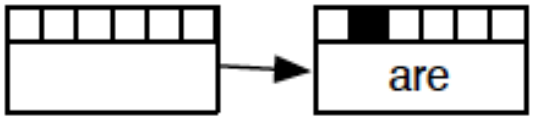
consult phrase translation table for all input phrases

Decoding: Start with Initial Hypothesis



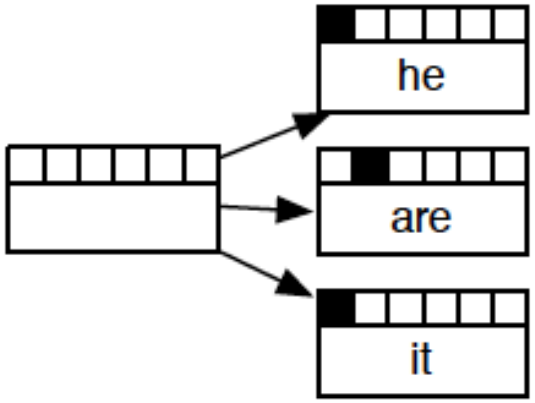
initial hypothesis: no input words covered, no output produced

Decoding: Hypothesis Expansion



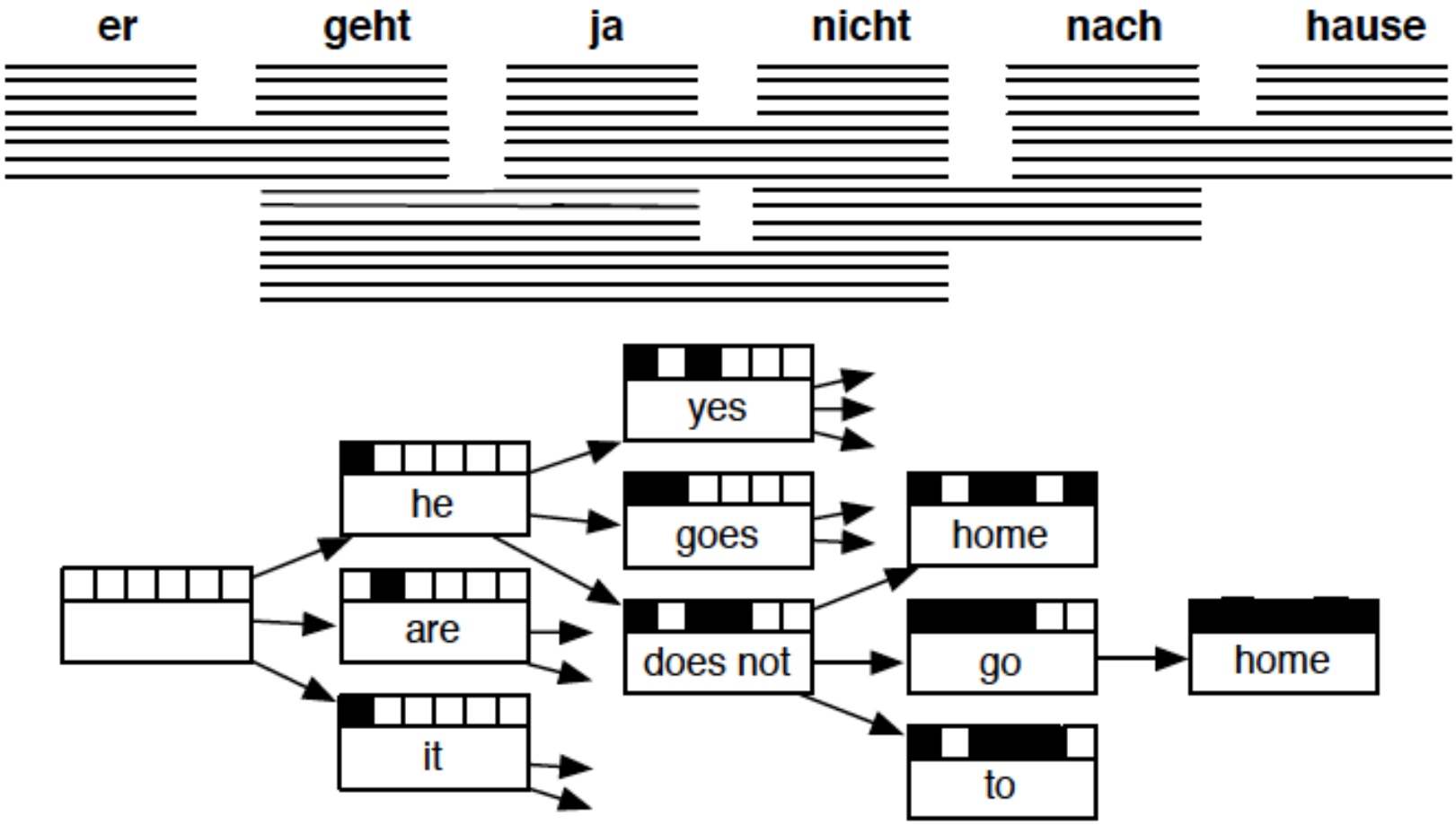
pick any translation option, create new hypothesis

Decoding: Hypothesis Expansion



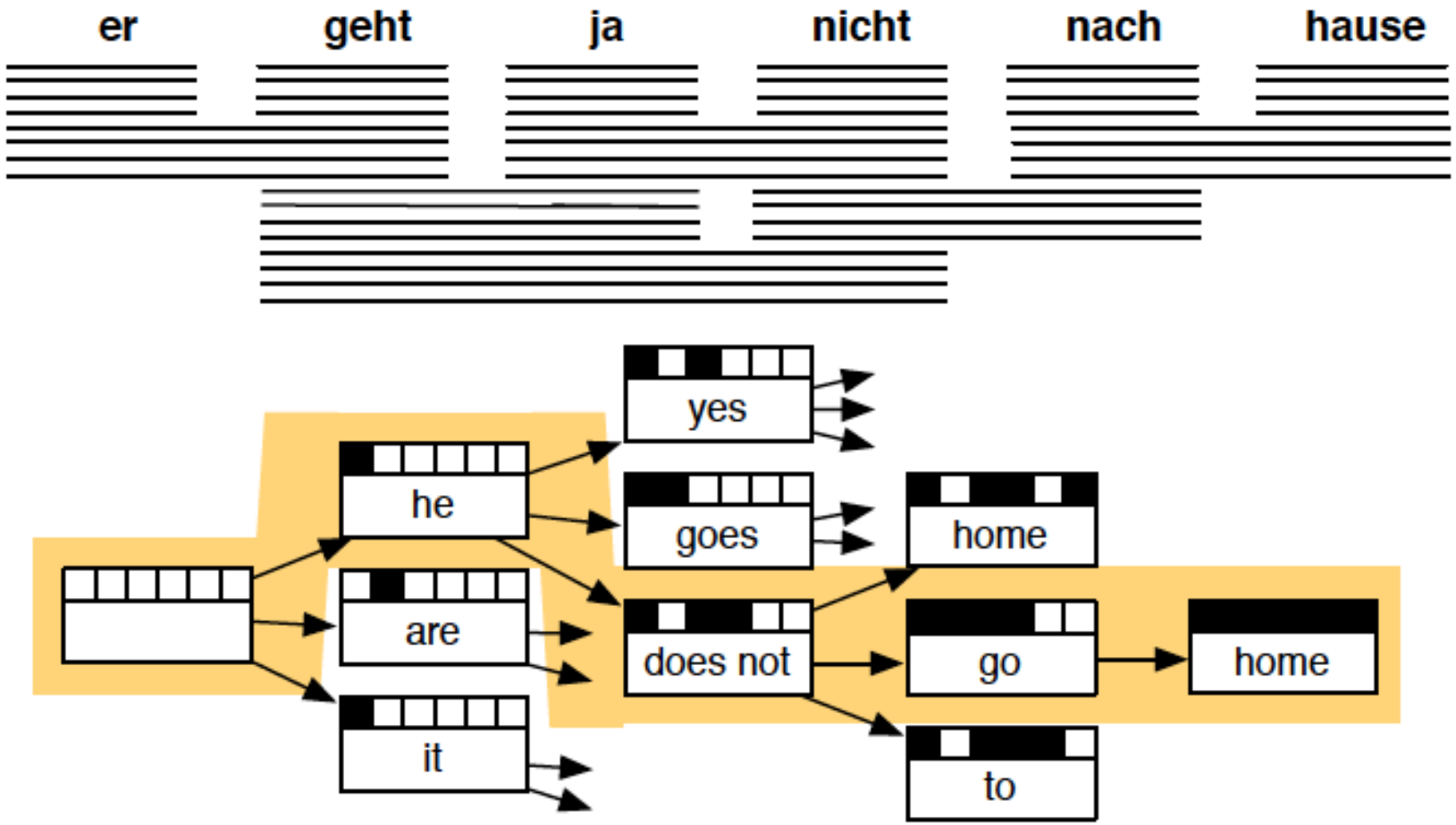
create hypotheses for all other translation options

Decoding: Hypothesis Expansion



also create hypotheses from created partial hypothesis

Decoding: Find Best Path



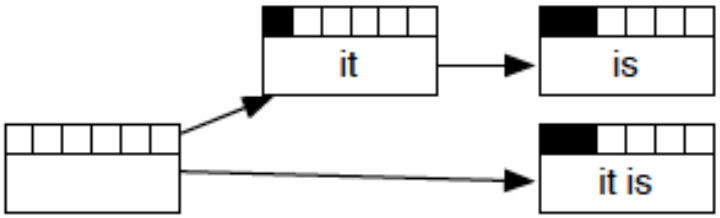
backtrack from highest scoring complete hypothesis

Size of the search space

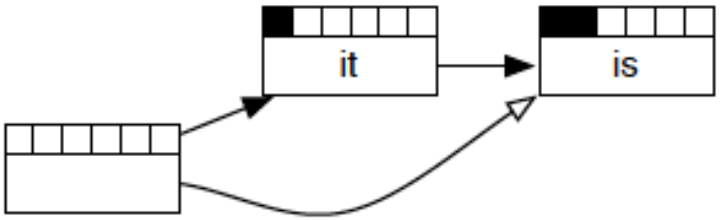
- Decoding is an NP-complete problem
- The search space is exponential
- How can we reduce the search?
- Dynamic programming (risk free)
- Heuristic search (risky)

Recombination

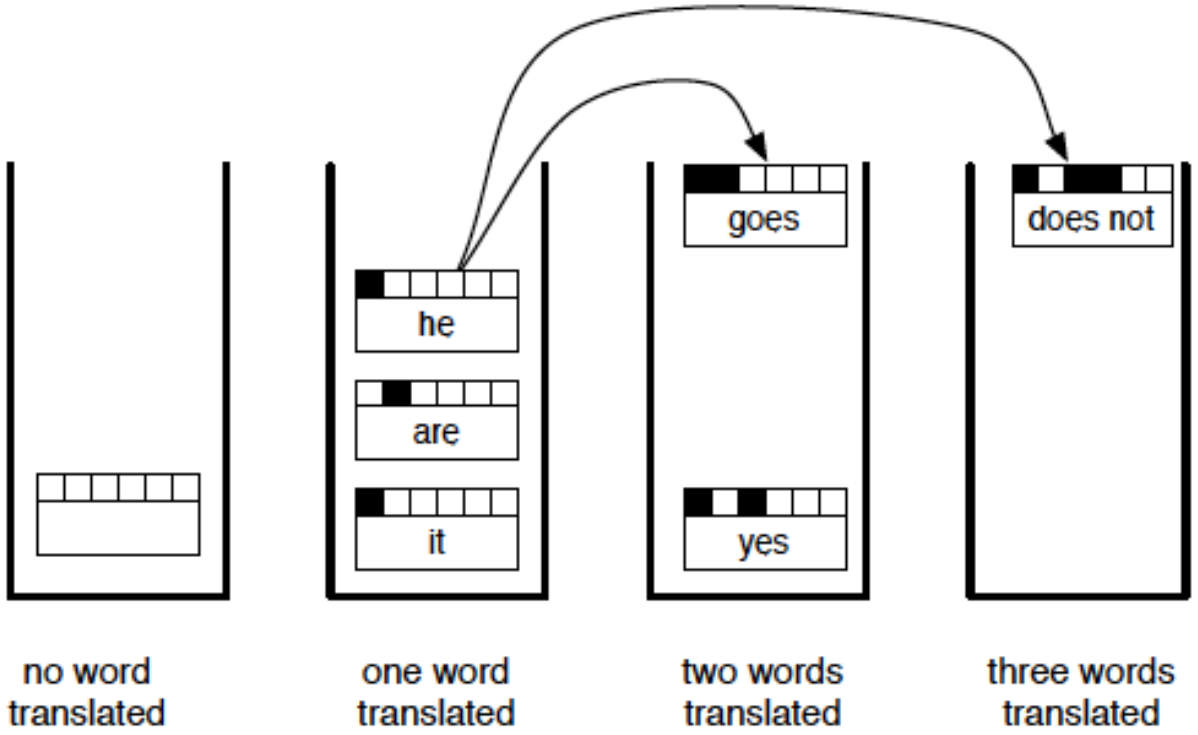
- Two hypothesis paths lead to two matching hypotheses
  - same number of foreign words translated
  - same English words in the output
  - different scores



- Worse hypothesis is dropped



Stacks



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

Search Errors

- We are using a **heuristic search** to prune the search space
- There are no guarantees of admissibility (like in A* search)
- We may therefore prune out a partial hypothesis that would have lead to the most probable translation, if we hadn't pruned it early on

Model Errors

- Model errors are different than search errors
- If the model scores an incorrect translation higher than higher than the correct translation, then it is the model's fault not the search strategy's fault

Other Decoding Algorithms

- A* search
- Greedy hill-climbing
- Using finite state transducers (standard toolkits)

Discriminative Modeling

Noisy Channels Again

$$
\begin{aligned}
\mathbf{e}^* &= \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g}) \\
&= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})} \\
&= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e}) \\
&= \arg\max_{\mathbf{e}} \log p(\mathbf{g} \mid \mathbf{e}) + \log p(\mathbf{e})
\end{aligned}
$$

Noisy Channels Again

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} p(\mathbf{e} \mid \mathbf{g})$$

$$= \arg\max_{\mathbf{e}} \frac{p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})}{p(\mathbf{g})}$$

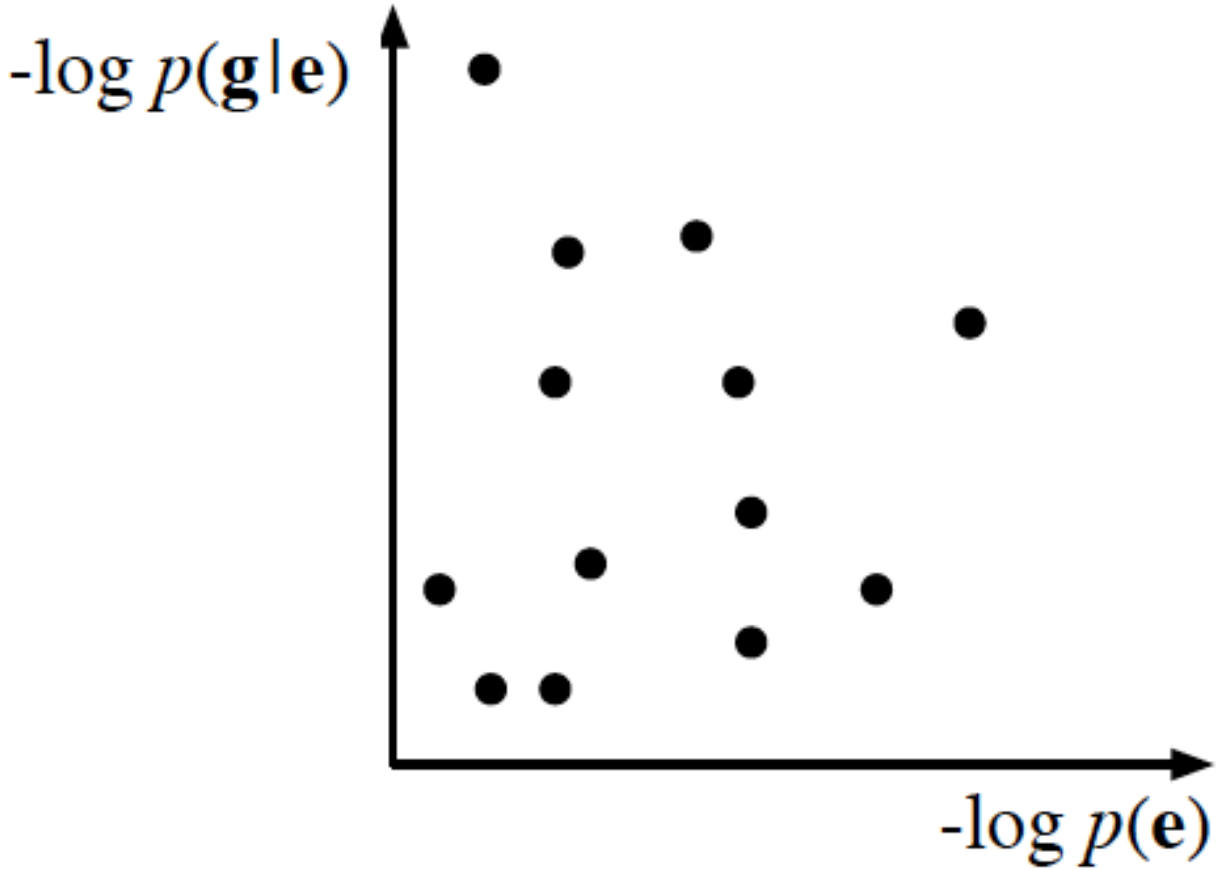
$$= \arg\max_{\mathbf{e}} p(\mathbf{g} \mid \mathbf{e}) \times p(\mathbf{e})$$

$$= \arg\max_{\mathbf{e}} \log p(\mathbf{g} \mid \mathbf{e}) + \log p(\mathbf{e})$$

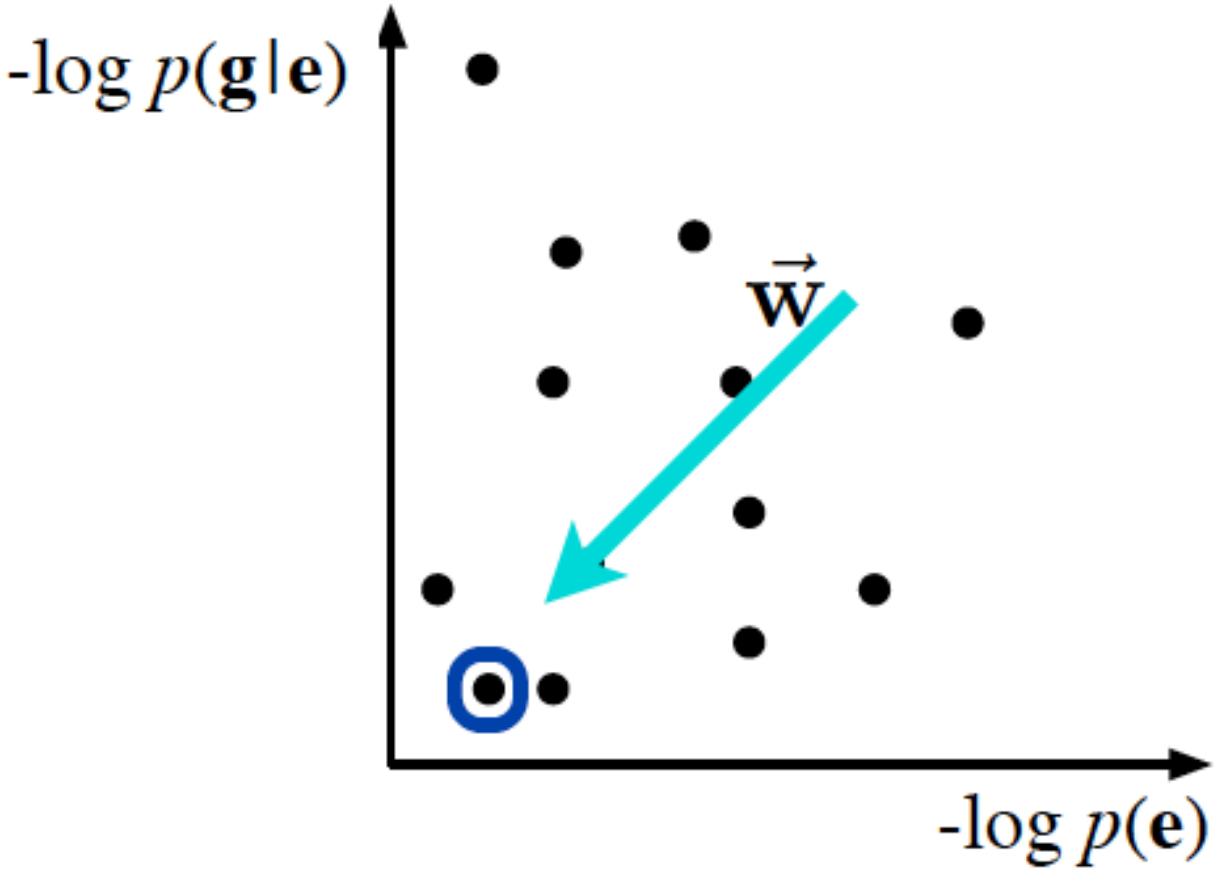
$$= \arg\max_{\mathbf{e}} \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}^\top}_{\mathbf{w}^\top} \underbrace{\begin{bmatrix} \log p(\mathbf{g} \mid \mathbf{e}) \\ \log p(\mathbf{e}) \end{bmatrix}}_{\mathbf{h}(\mathbf{g},\mathbf{e})}$$

Log-linear Model

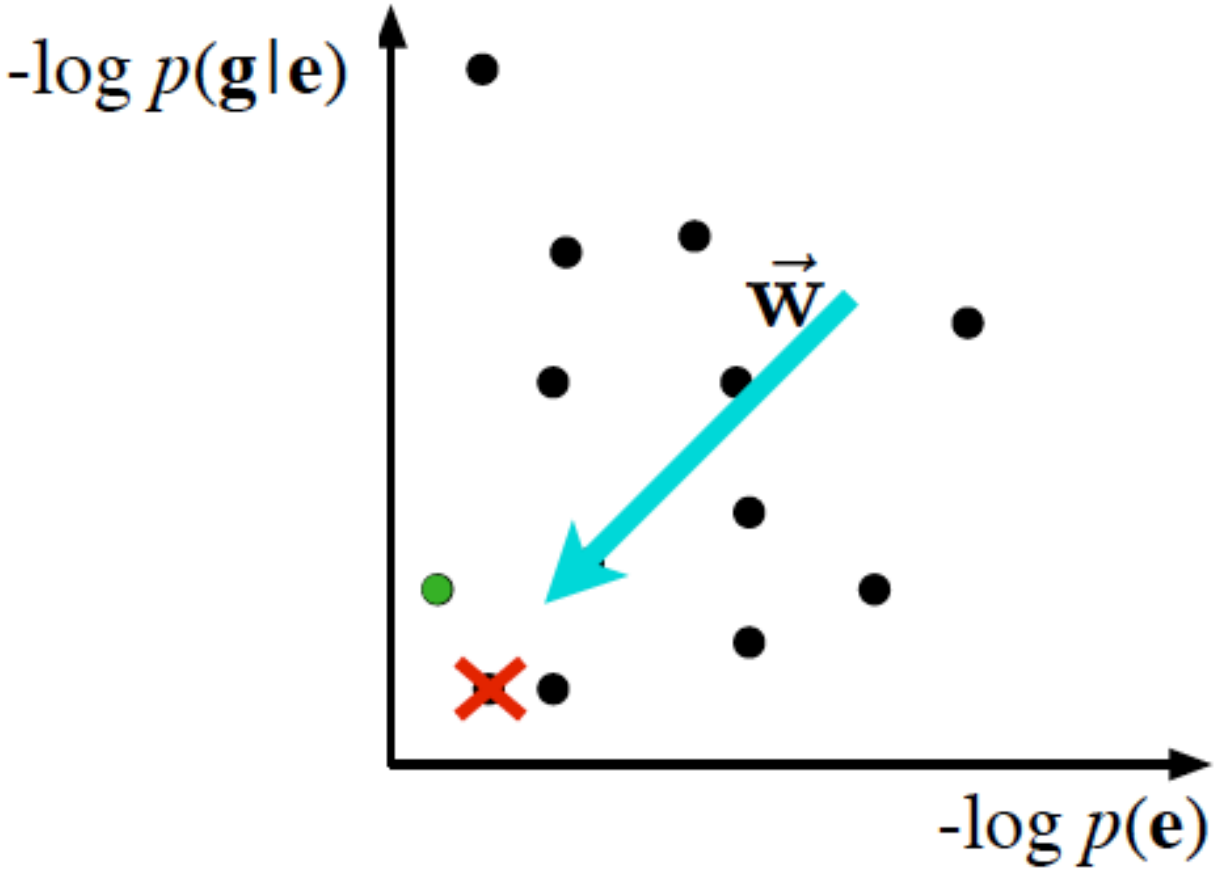
The Noisy Channel



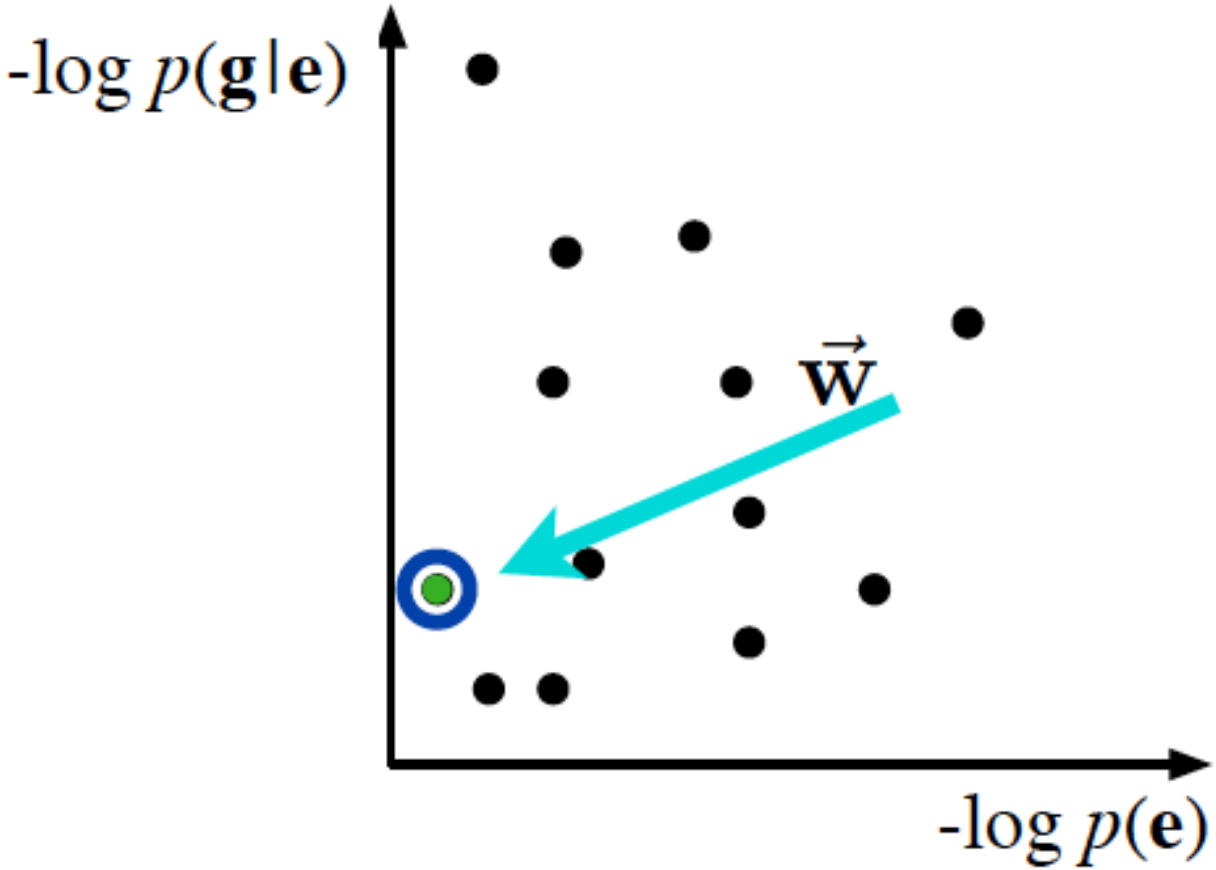
As a Linear Model



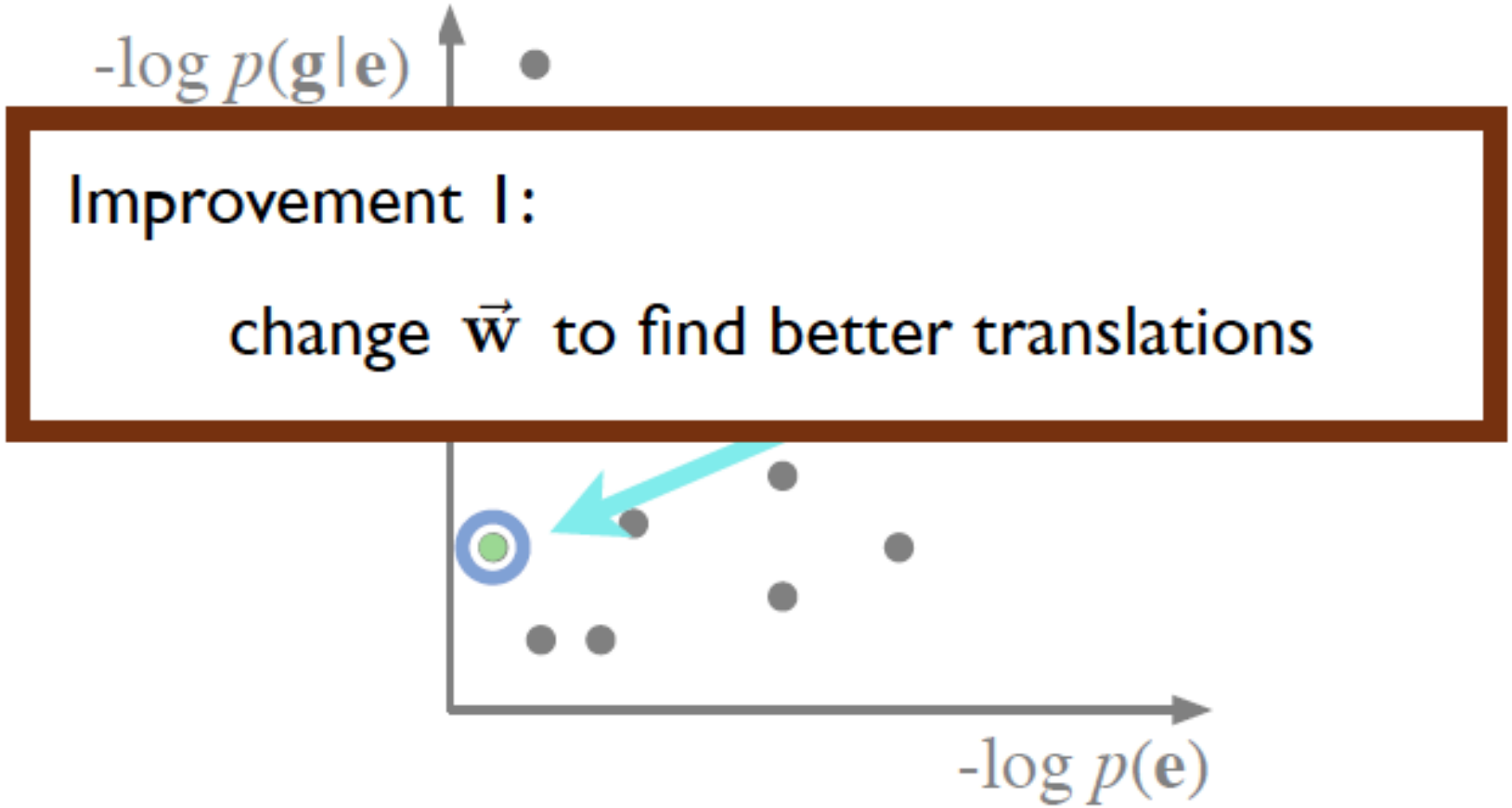
As a Linear Model



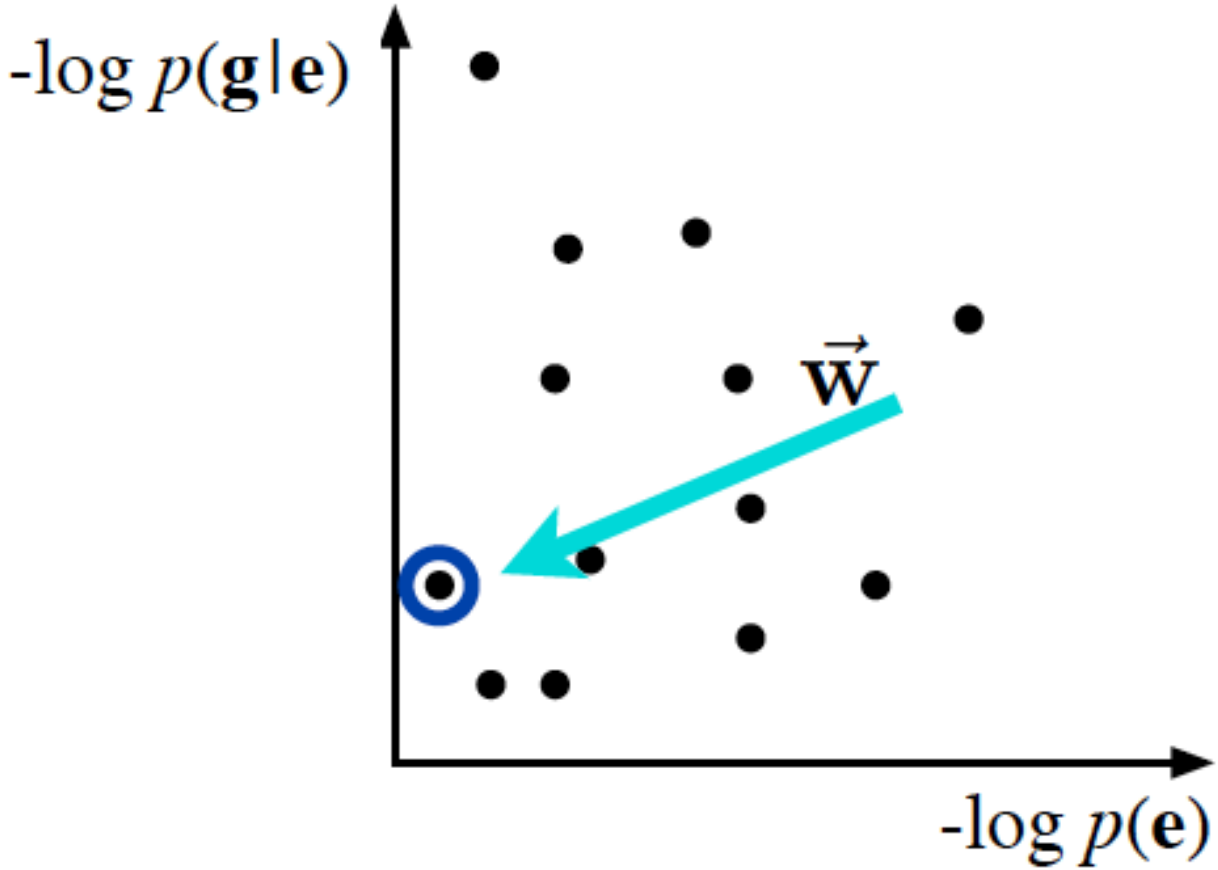
As a Linear Model



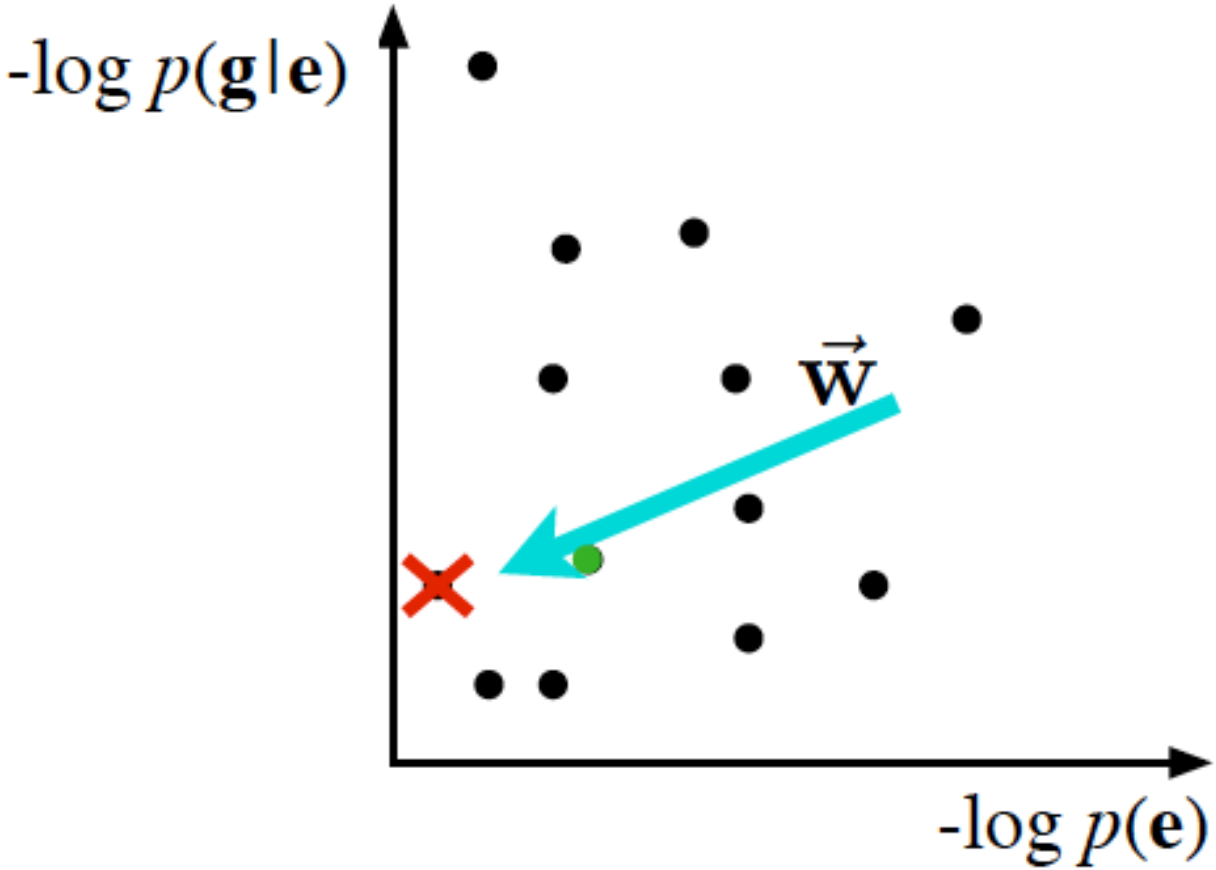
As a Linear Model



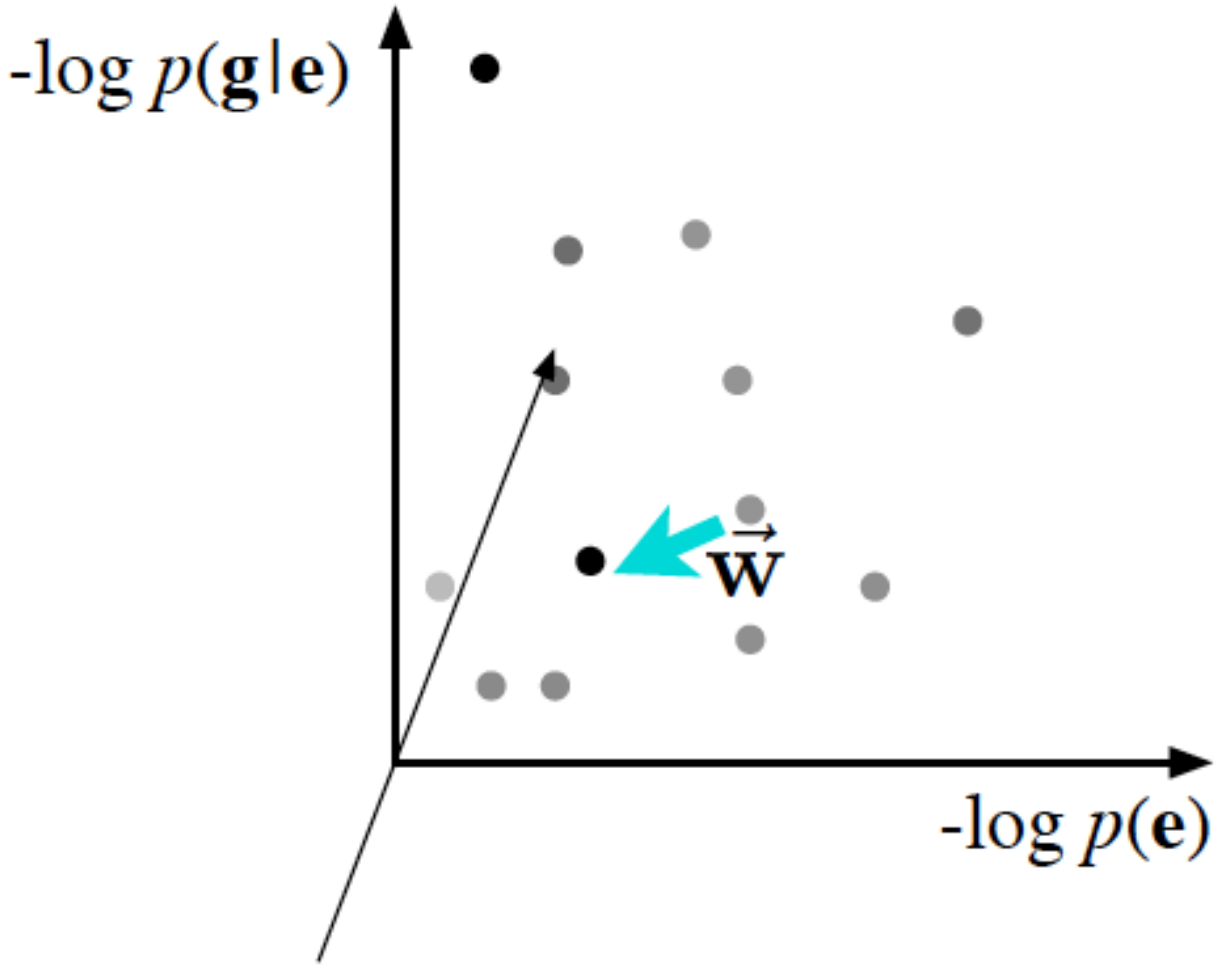
As a Linear Model



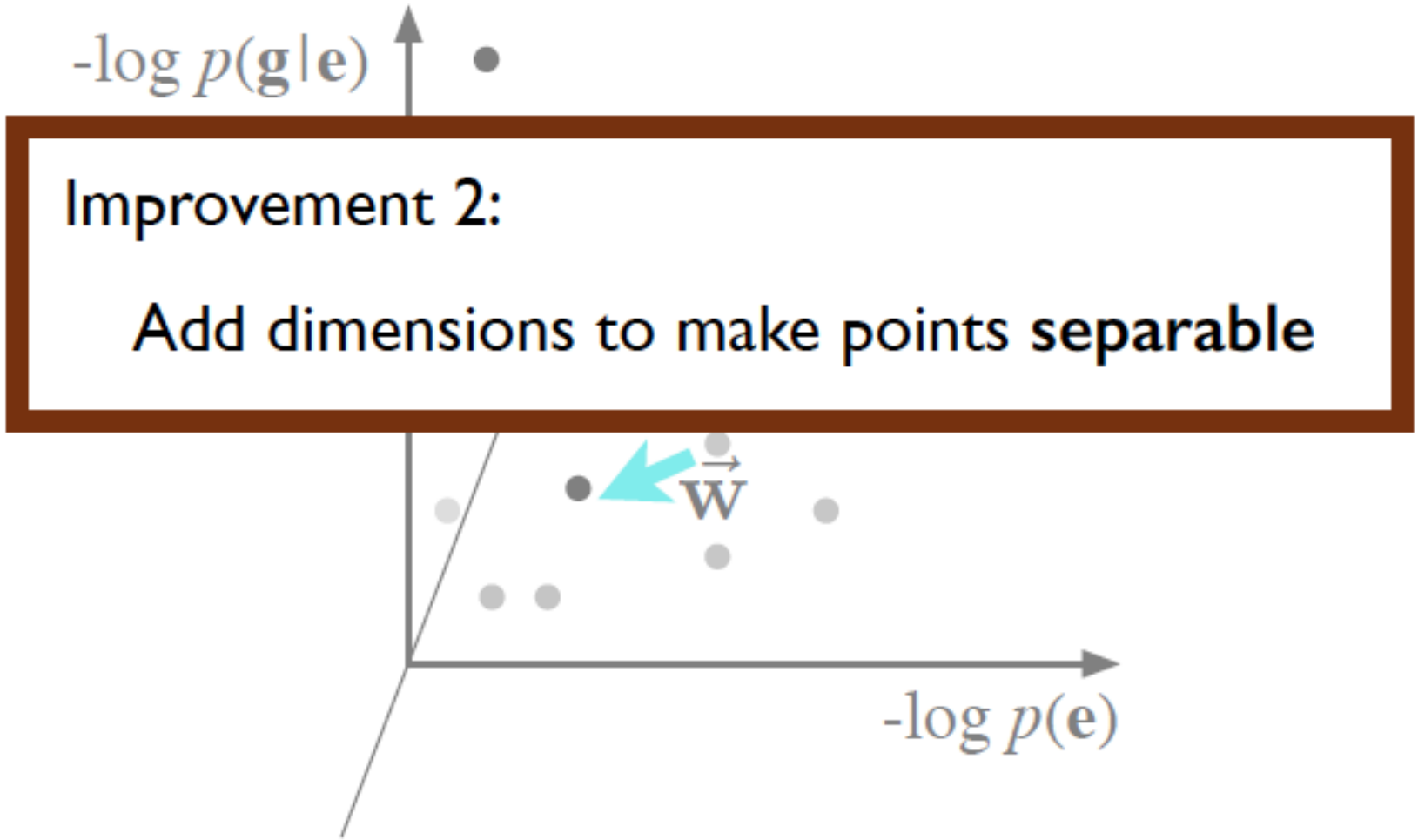
As a Linear Model



As a Linear Model



As a Linear Model



Linear Models

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \mathbf{w}^\top \mathbf{h}(\mathbf{g}, \mathbf{e})$$

- Improve the modeling capacity of the noisy channel in two ways
  - Reorient the weight vector
  - Add new dimensions (***new features***)
- Questions
  - What features? $\mathbf{h}(\mathbf{g}, \mathbf{e})$
  - How do we set the weights? $\mathbf{w}$

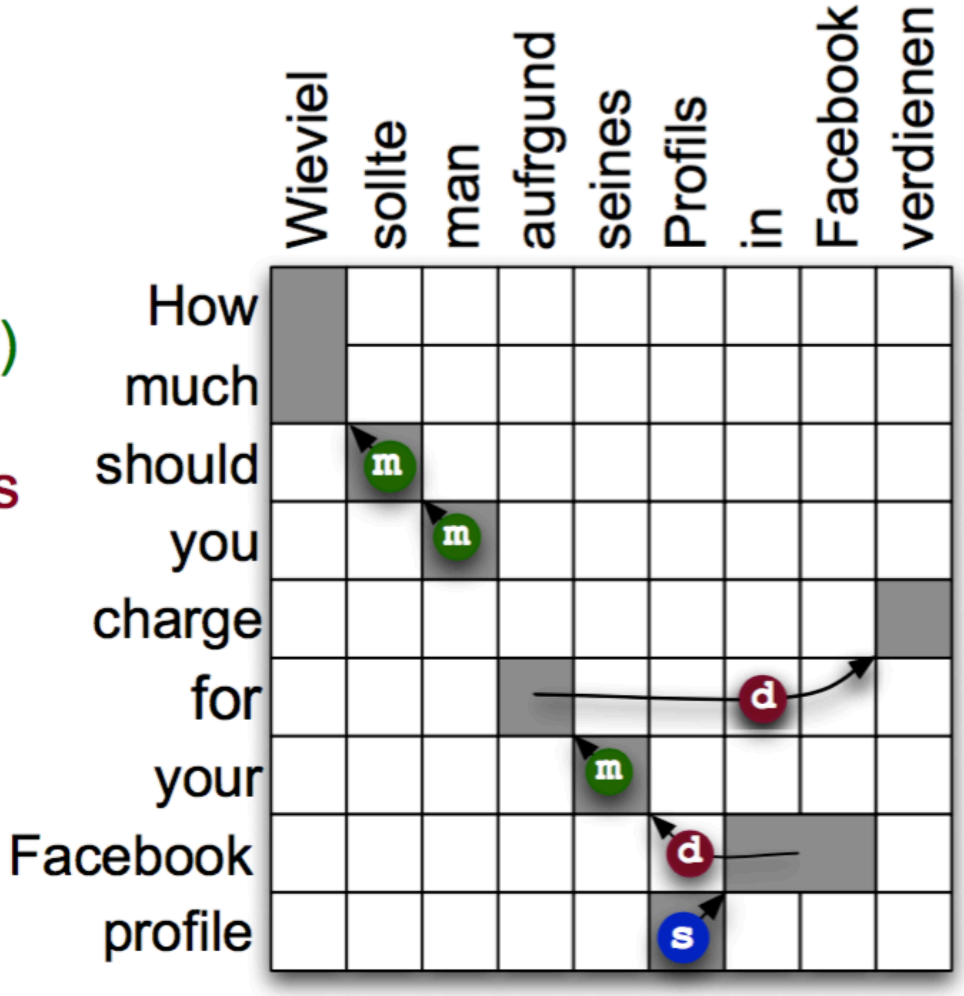
Standard Features

- **Target side features**
  - log p(e) [*n*-gram language model]
  - Number of words in hypothesis
  - Non-English character count
- **Source + target features**
  - log relative frequency e|*f* of each rule [log #(e,f) - log #(f)]
  - log relative frequency *f*|e of each rule [log #(e,f) - log #(e)]
  - "lexical translation" log probability e|*f* of each rule [≈ log p_{model1}(e|f)]
  - "lexical translation" log probability *f*|e of each rule [≈ log p_{model1}(f|e)]
- **Other features**
  - Count of rules/phrases used
  - Reordering pattern probabilities

Lexicalized Reordering

m: monotone (keep order)
s: swap order
d: become discontinuous

Reordering features are probability estimates of s, d, and m



Learning Weights

- Try to match the reference translation *exactly*
  - **Conditional random field**
    - Maximize the conditional probability of the reference translations
    - "Average" over the different latent variables

- These methods give "full credit" when the model *exactly* produces the reference and no credit otherwise
- **What is the problem with this?**

Cost-Sensitive Training

- Assume we have a **cost function** that gives a score for how good/bad a translation is

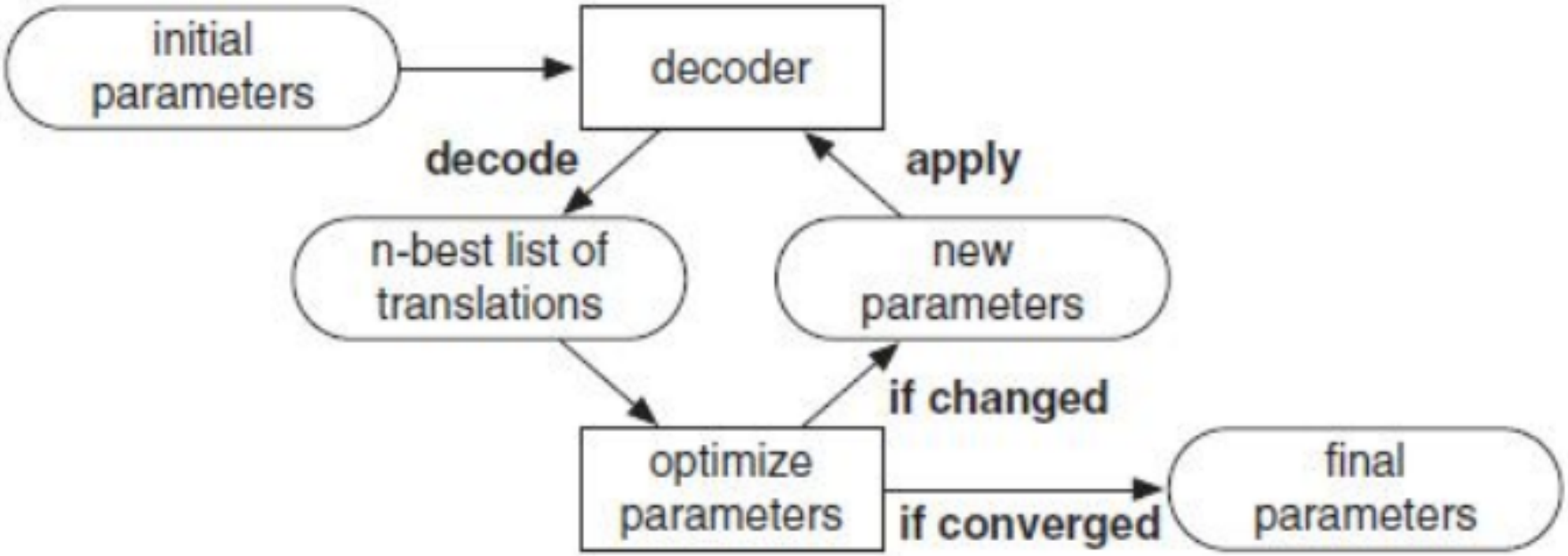
$$\ell(\hat{\mathbf{e}}, \mathcal{E}) \mapsto [0, 1]$$

- Optimize the weight vector by making reference to this function

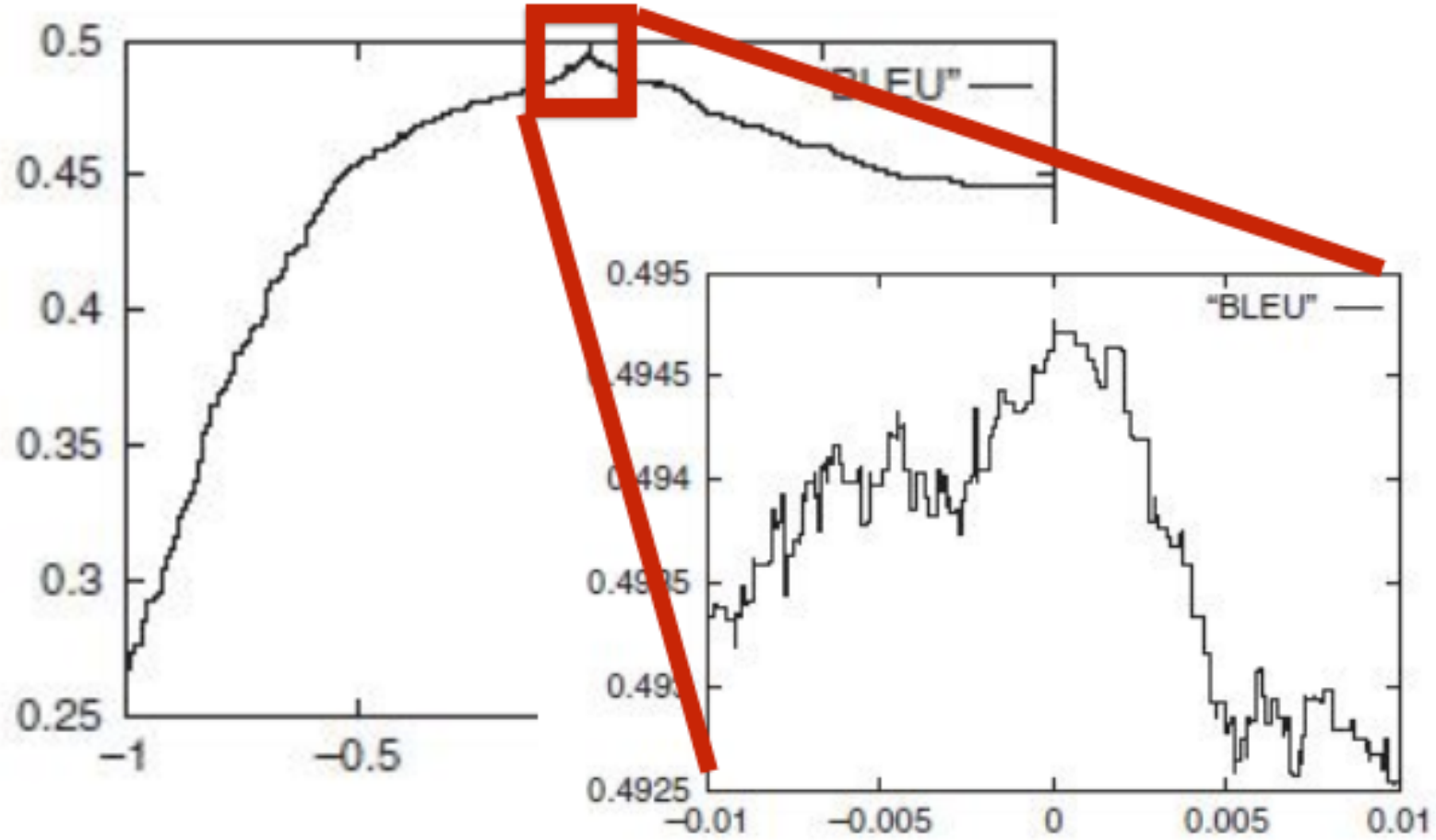
MERT

- **M**inimum **E**rror **R**ate **T**raining
- Directly optimize for an automatic evaluation metric instead of likelihood
- Maximize the BLEU score on a **held out development set**
- Iteratively update the parameters by re-scoring n-best lists and comparing the highest scoring translation to the reference

Iterative parameter tuning



The effect on BLEU varying one parameter



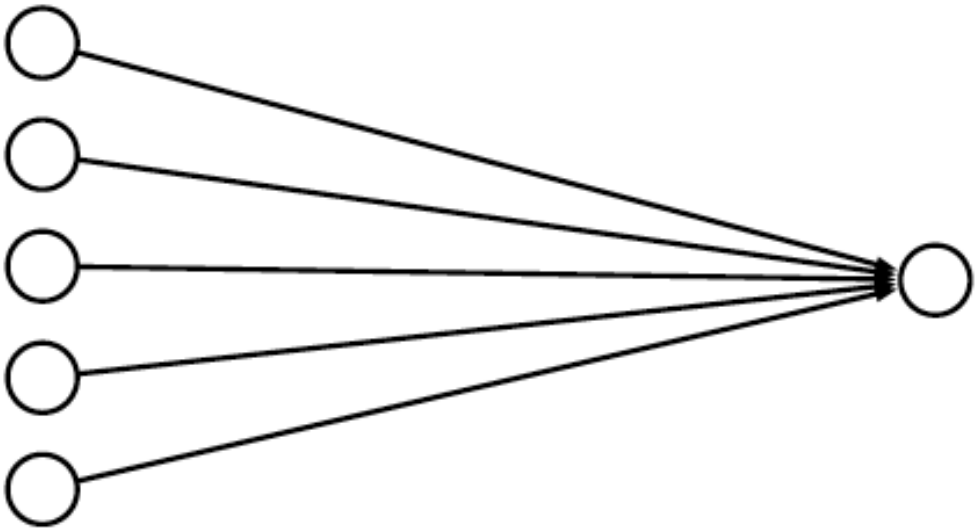
Neural
Machine Translation

Linear Models

- We used before weighted linear combination of feature values h_j and weights λ_j

$$\text{score}(\lambda, \mathbf{d}_i) = \sum_j \lambda_j \; h_j(\mathbf{d}_i)$$

- Such models can be illustrated as a "network"



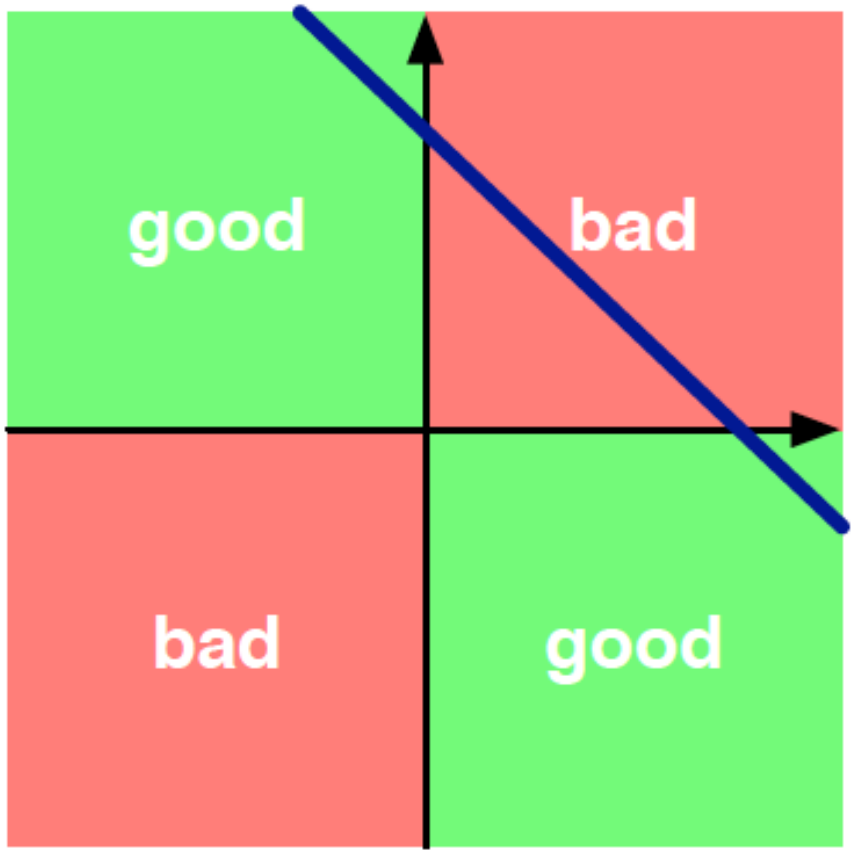
Limits of Linearity

- We can give each feature a weight

- But not more complex value relationships, e.g,
  - any value in the range [0;5] is equally good
  - values over 8 are bad
  - higher than 10 is not worse

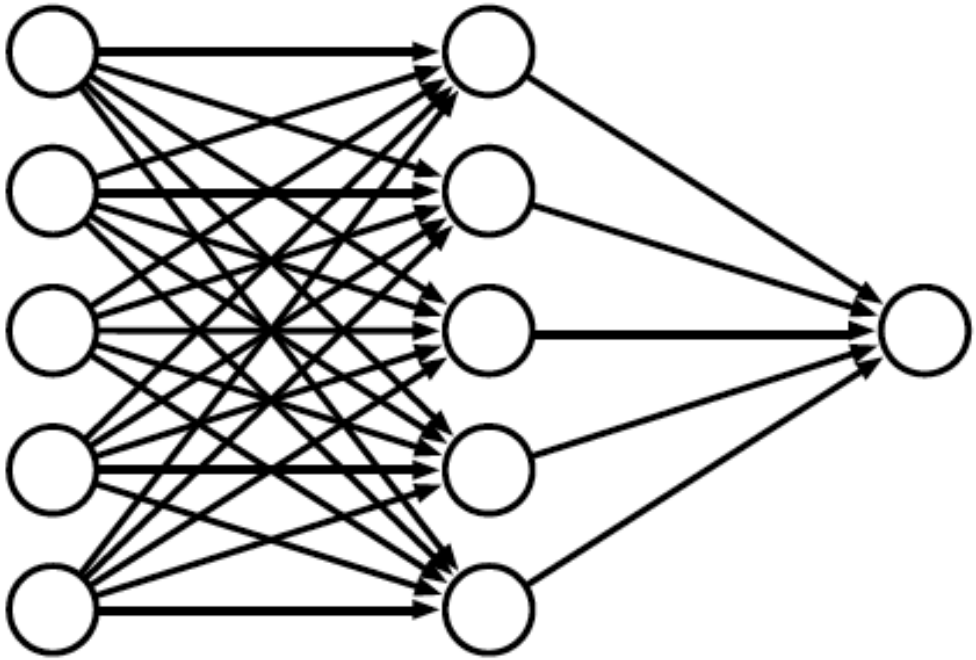
XOR

- Linear models cannot model XOR



Multiple Layers

- Add an intermediate ("hidden") layer of processing (each arrow is a weight)



- Have we gained anything so far?

Non-Linearity

- Instead of computing a linear combination

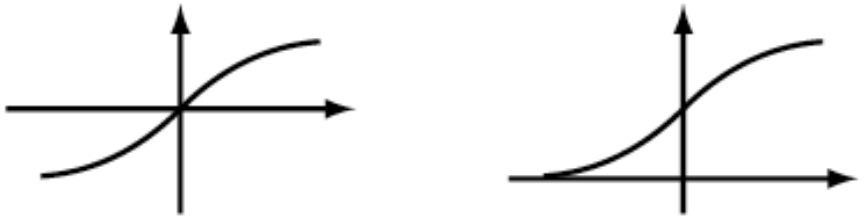
$$\text{score}(\lambda, \mathbf{d}_i) = \sum_j \lambda_j \; h_j(\mathbf{d}_i)$$

- Add a non-linear function

$$\text{score}(\lambda, \mathbf{d}_i) = f\Big(\sum_j \lambda_j \; h_j(\mathbf{d}_i)\Big)$$

- Popular choices

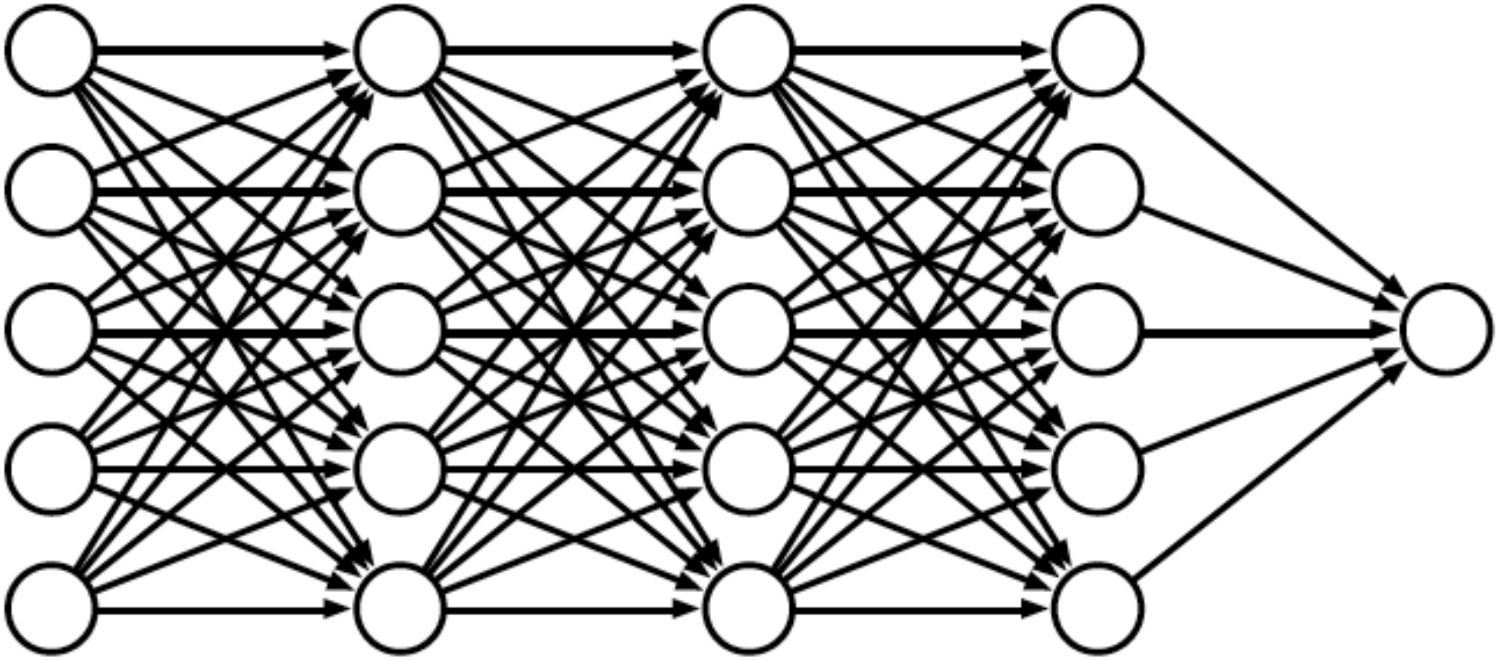
tanh(x) sigmoid(x) = $\frac{1}{1+e^{-x}}$



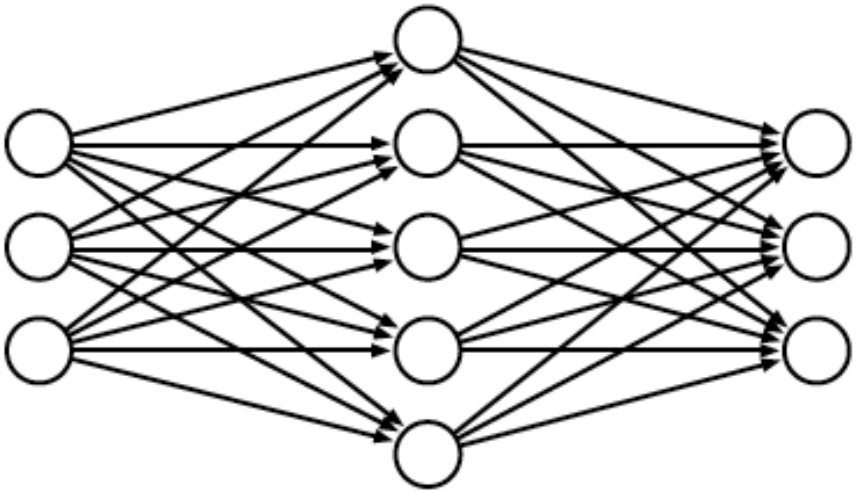
(sigmoid is also called the "logistic function")

Deep Learning

- More layers = deep learning



Neural Networks for Classification

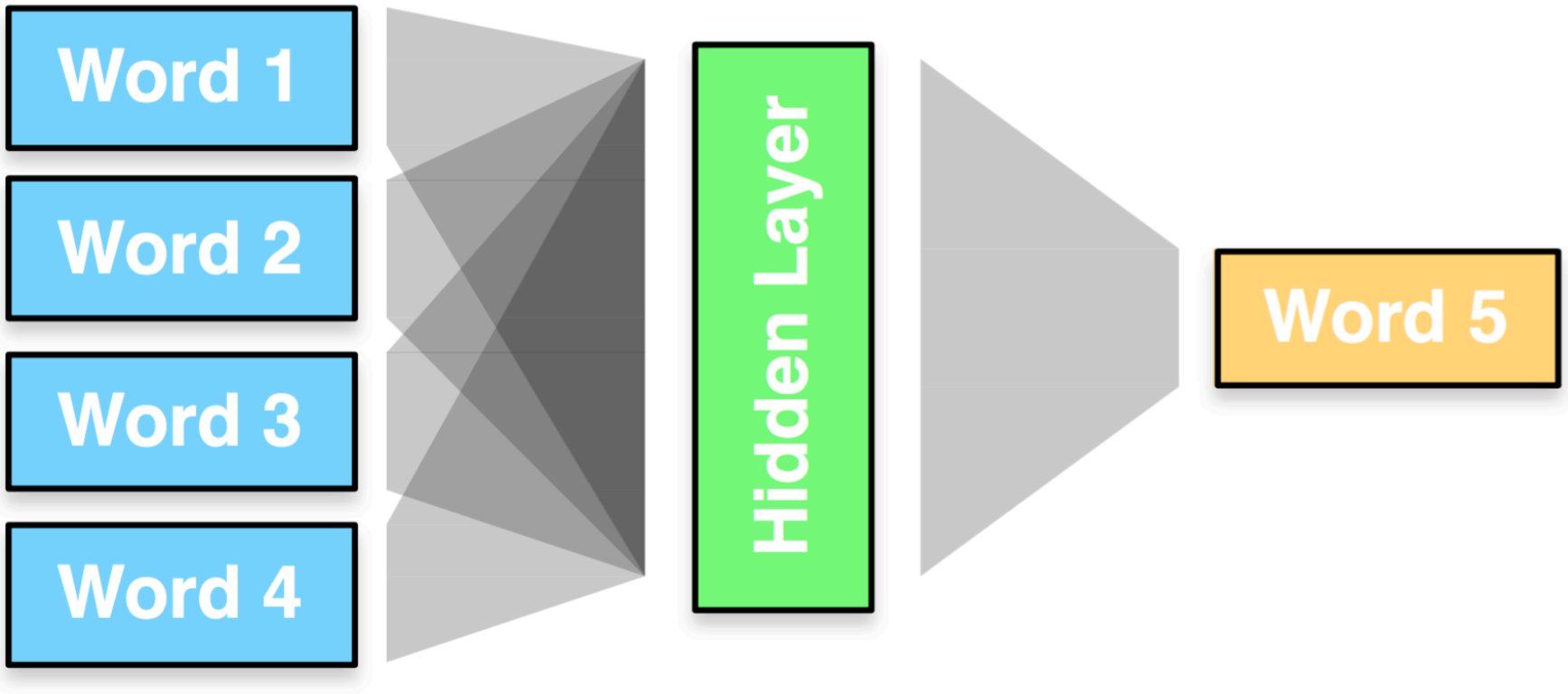


- Predict class: one output node per class
- Training data output: "One-hot vector", e.g., $\vec{y} = (0, 0, 1)^T$
- Prediction
  - predicted class is output node y_i with highest value
  - obtain posterior probability distribution by soft-max

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

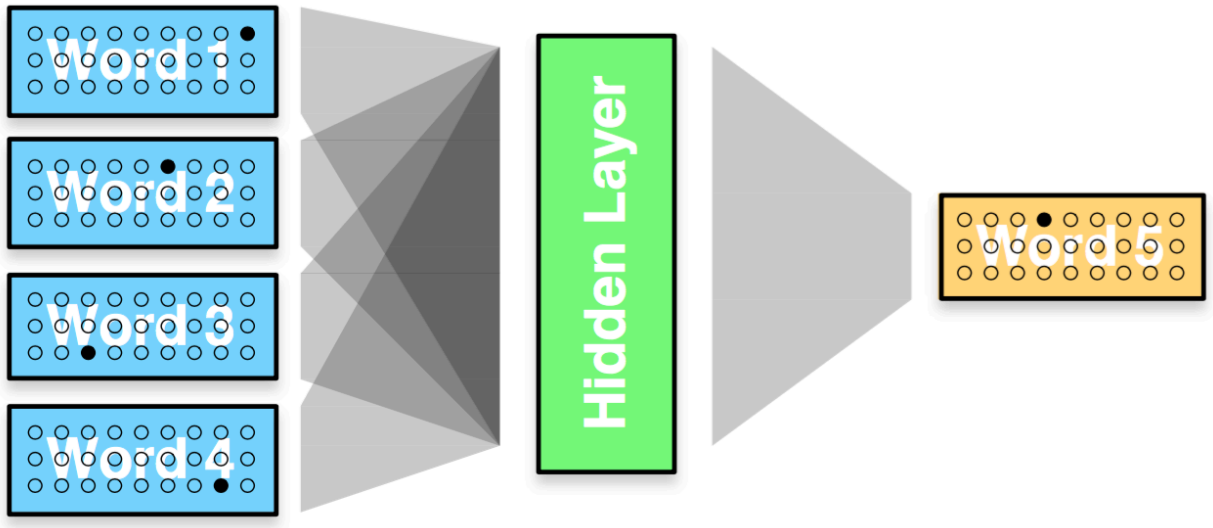
Neural N-gram Language Models

$$p(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-n+1}, \ldots, e_{i-1})$$

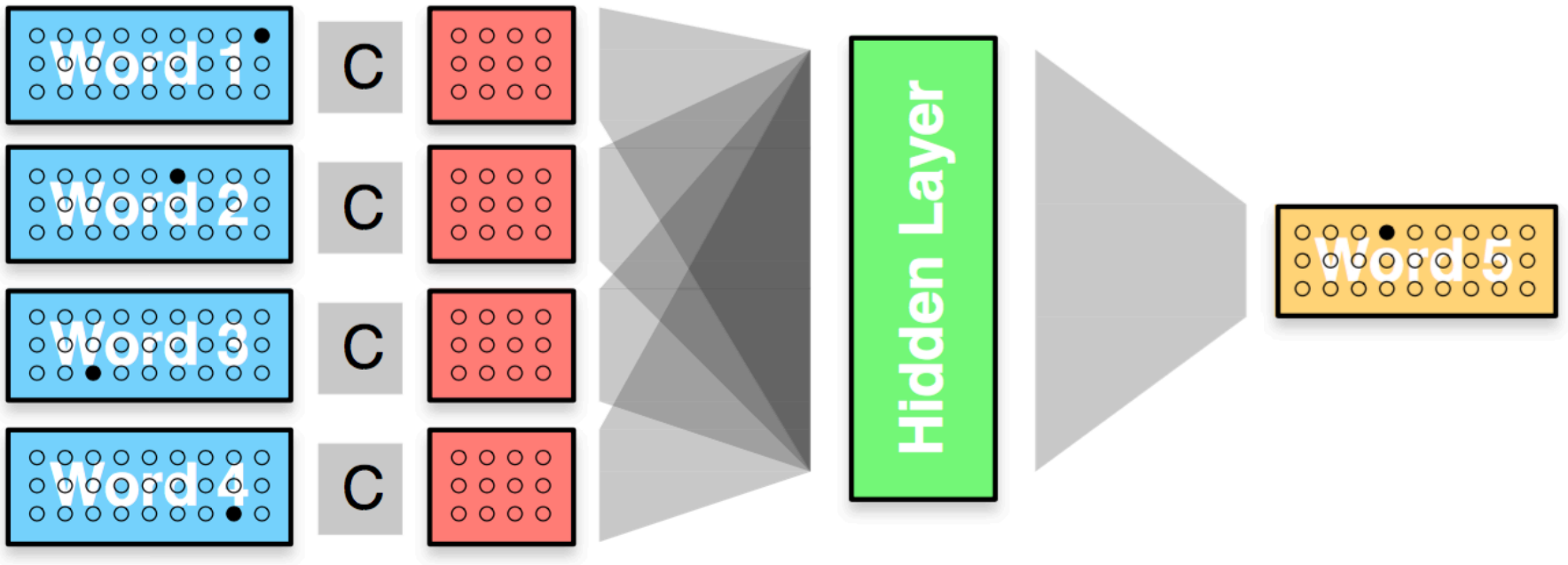


Word Representation

- Words are represented with a one-hot vector, e.g.,
  - dog = (0,0,0,0,1,0,0,0,0,....)
  - cat = (0,0,0,0,0,0,0,1,0,....)
  - eat = (0,1,0,0,0,0,0,0,0,....)
- That's a large vector!
- Remedies
  - limit to, say, 20,000 most frequent words, rest are OTHER

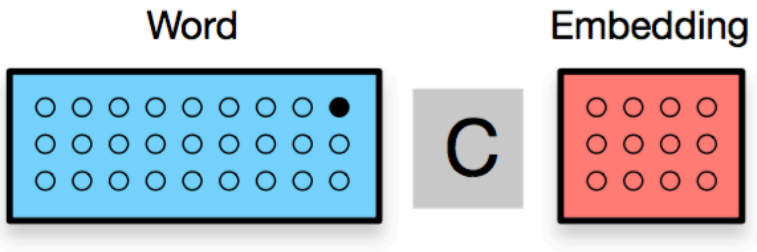


Word embeddings



- Map each word first into a lower-dimensional real-valued space
- Shared weight matrix C

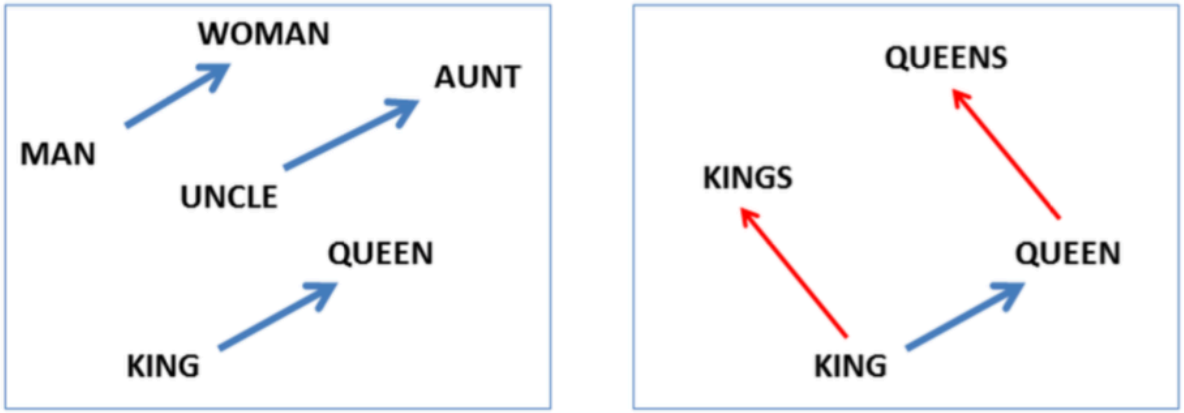
Word embeddings



- By-product: embedding of word into continuous space
- Similar contexts $\rightarrow$ similar embedding
- Recall: distributional semantics

Word embeddings

Word embeddings

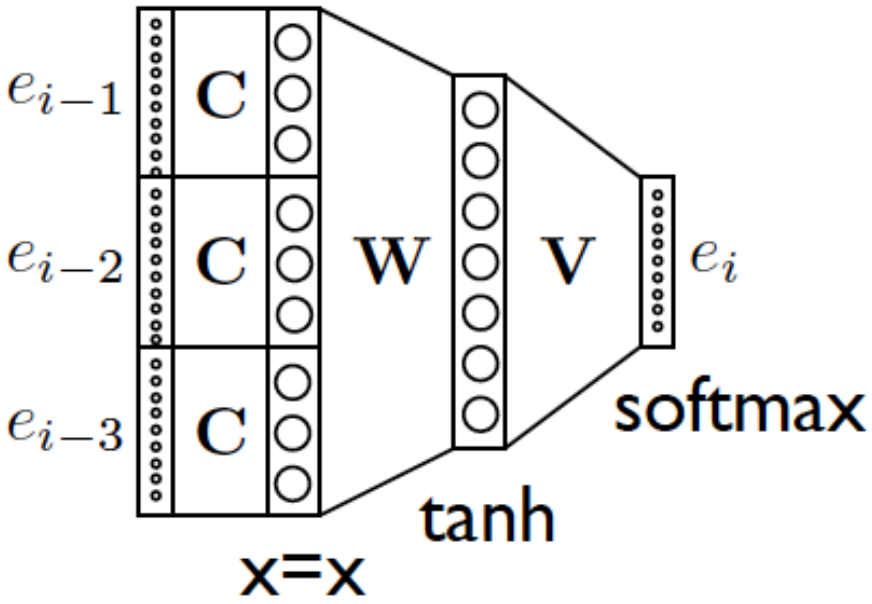


- Morphosyntactic regularities (Mikolov et al., 2013)
  - adjectives base form vs. comparative, e.g., good, better
  - nouns singular vs. plural, e.g., year, years
  - verbs present tense vs. past tense, e.g., see, saw
- Semantic regularities
  - clothing is to shirt as dish is to bowl
  - evaluated on human judgment data of semantic similarities

Bengio et al. (2003)

$$p(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-n+1}, \ldots, e_{i-1})$$

$$p(e_i \mid e_{i-n+1}, \ldots, e_{i-1}) =$$

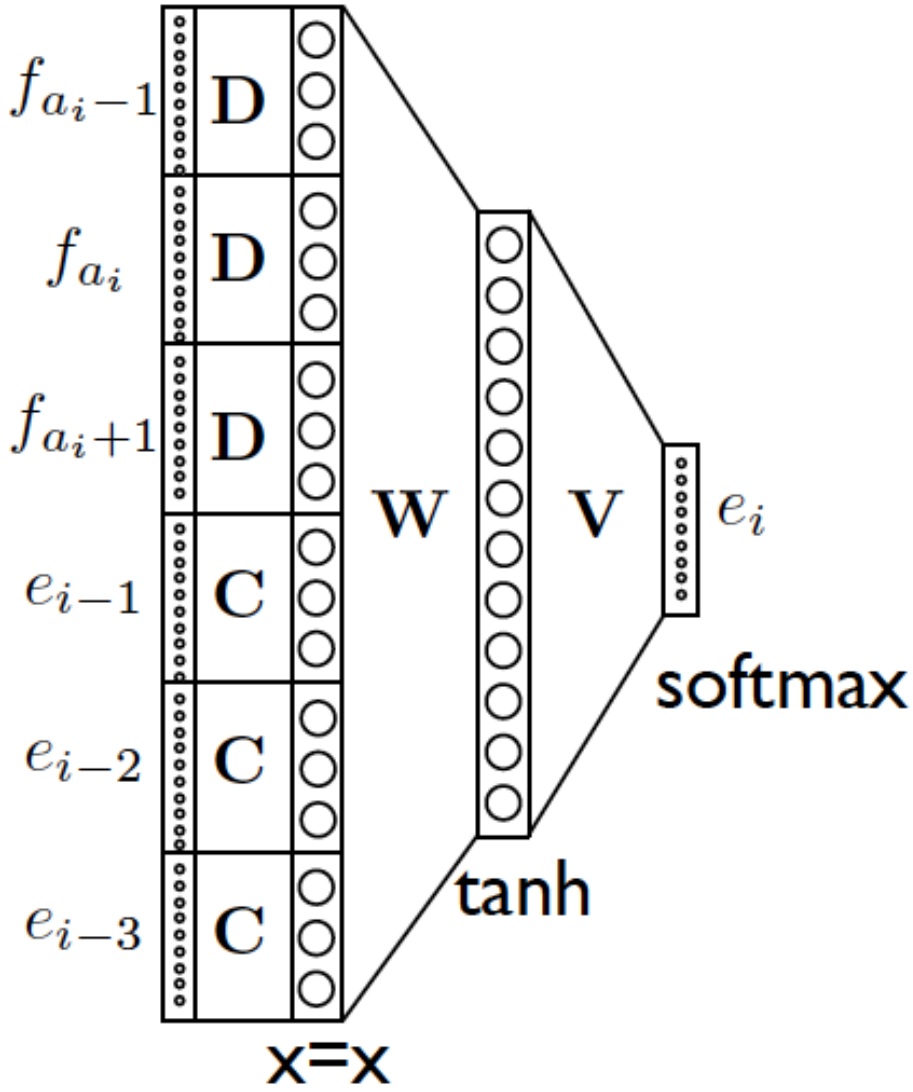


Devlin et al. (2014)

- Turn Bengio et al. (2003) into a translation model
- Conditional model; generate the next English word conditioned on
  - The previous *n* English words you generated
  - The aligned source word, and its *m* neighbors

$$p(\mathbf{e} \mid \mathbf{f}, \mathbf{a}) = \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1})$$

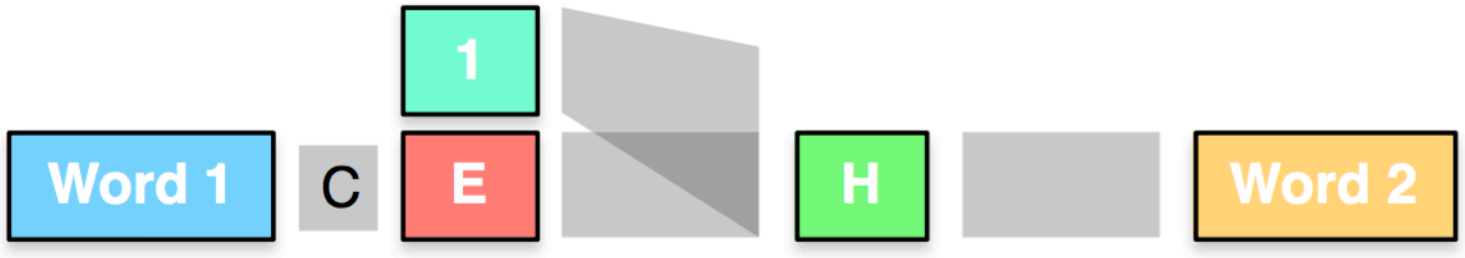
$$p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1}) =$$



Summary

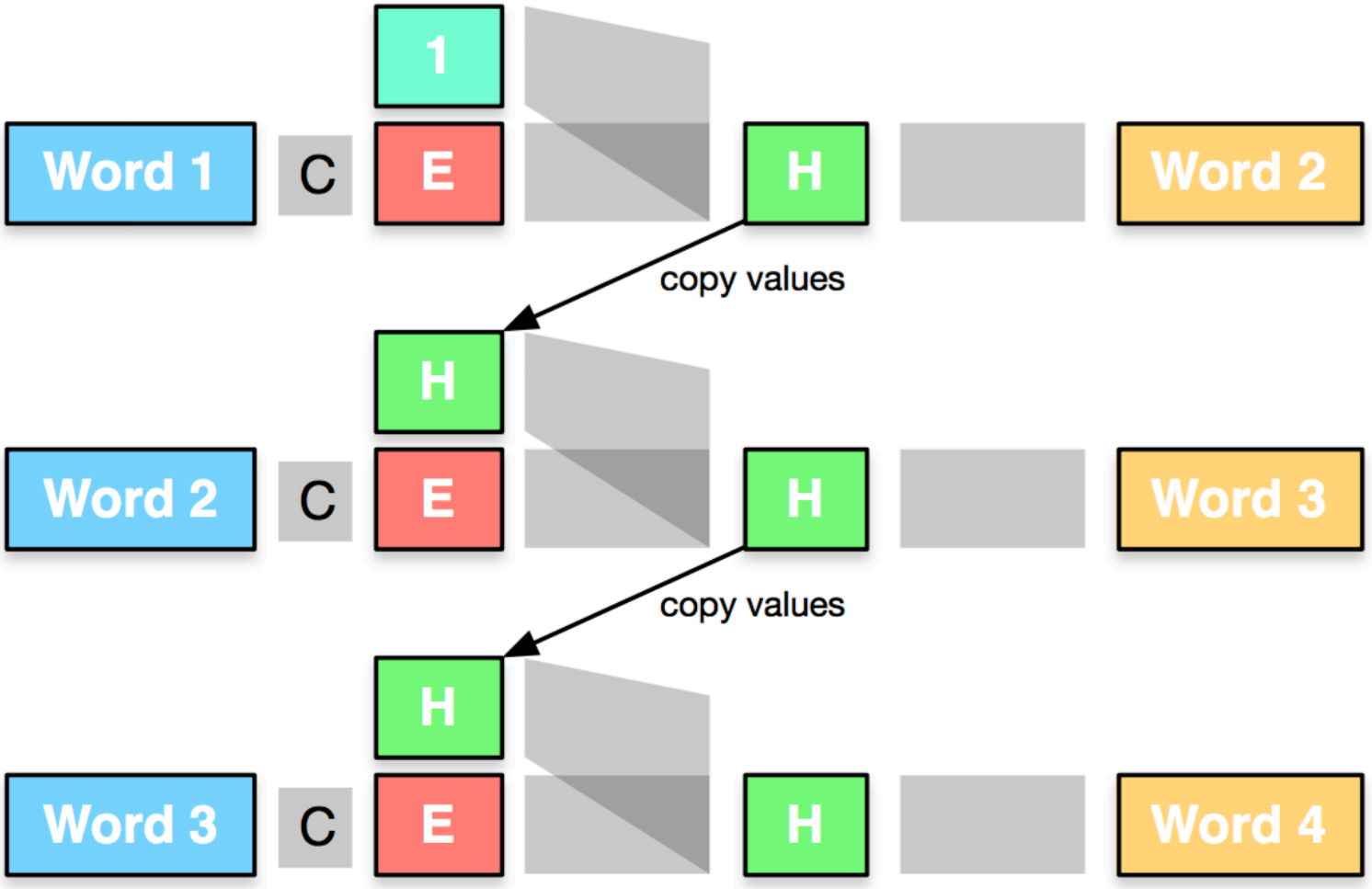
- Two problems in standard statistical models
  - We don't condition on enough stuff
  - We don't know what features to use when we condition on lots of structure
- Punchline: **Neural networks let us condition on a lot of stuff without an exponential growth in parameters**

Recurrent NN

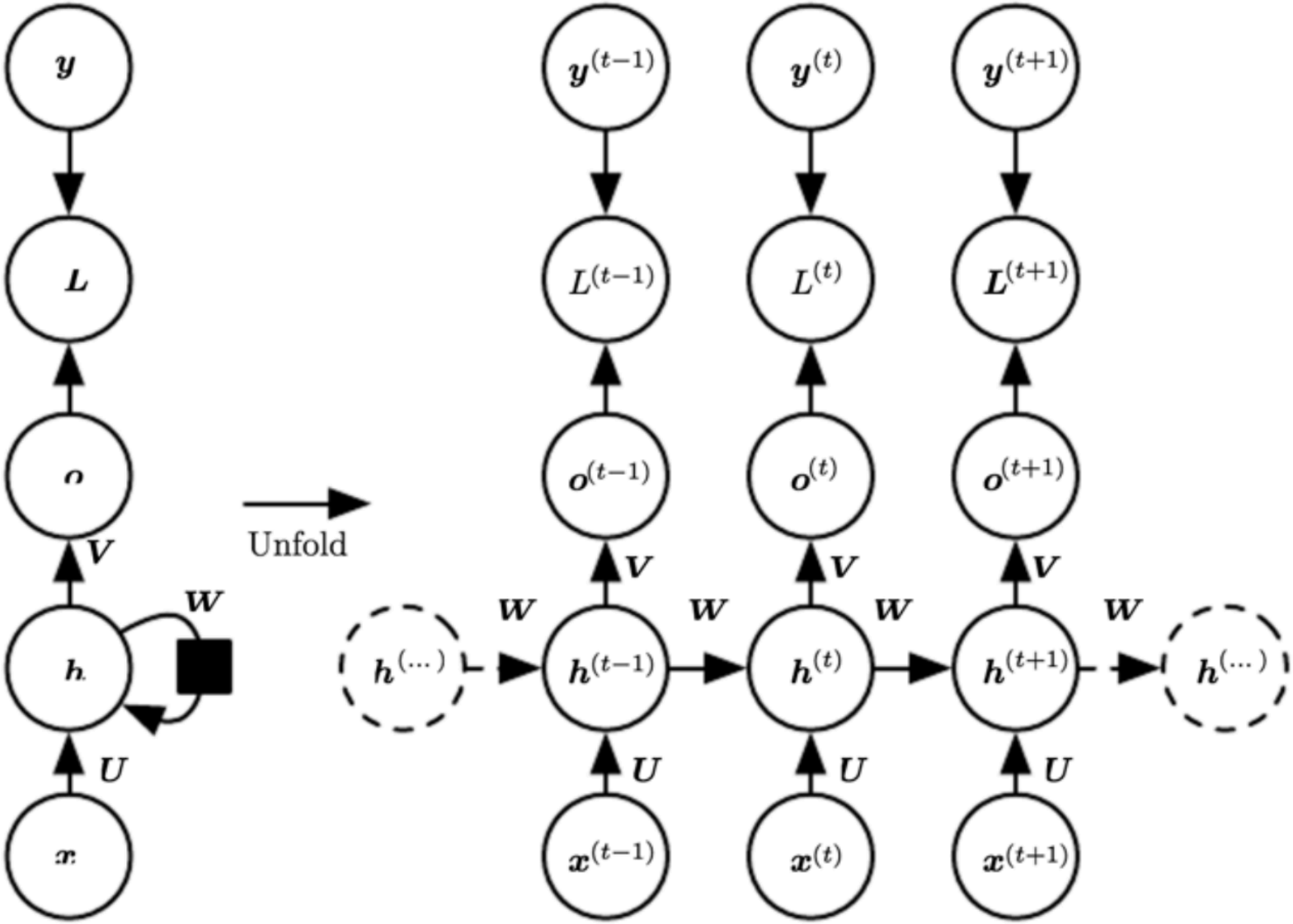


- Start: predict second word from first
- Mystery layer with nodes all with value 1

Recurrent NN



Recurrent NN

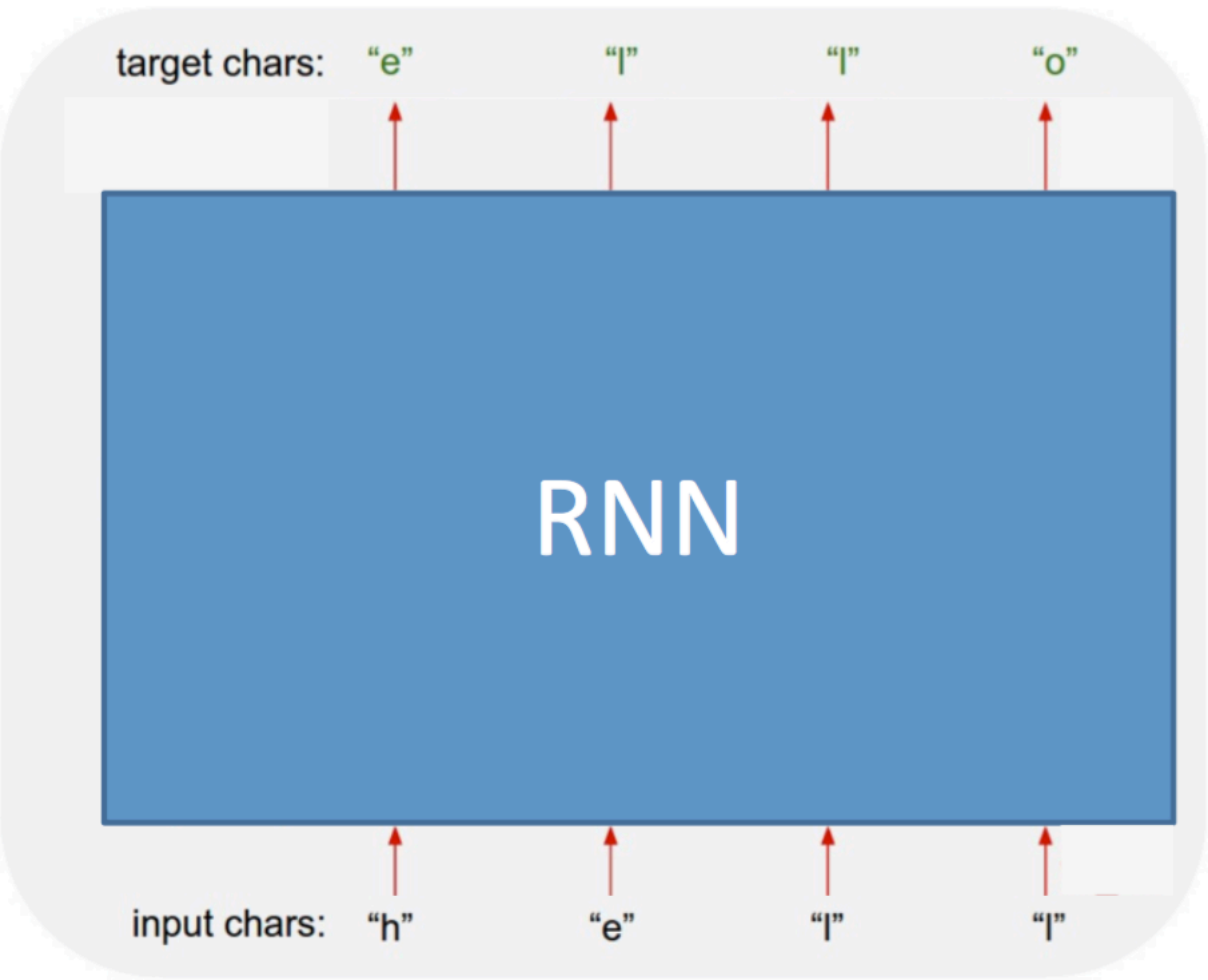


RNN Language Model

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

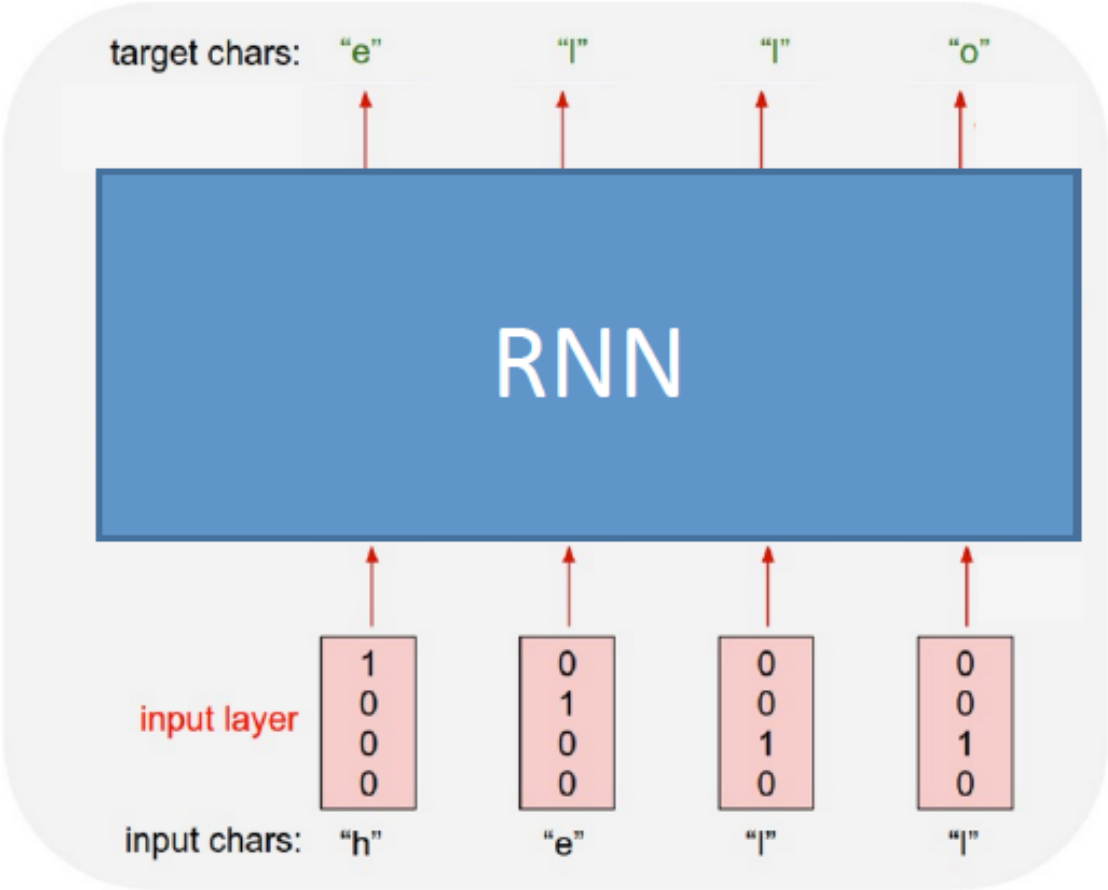


RNN – *Input Layer*

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



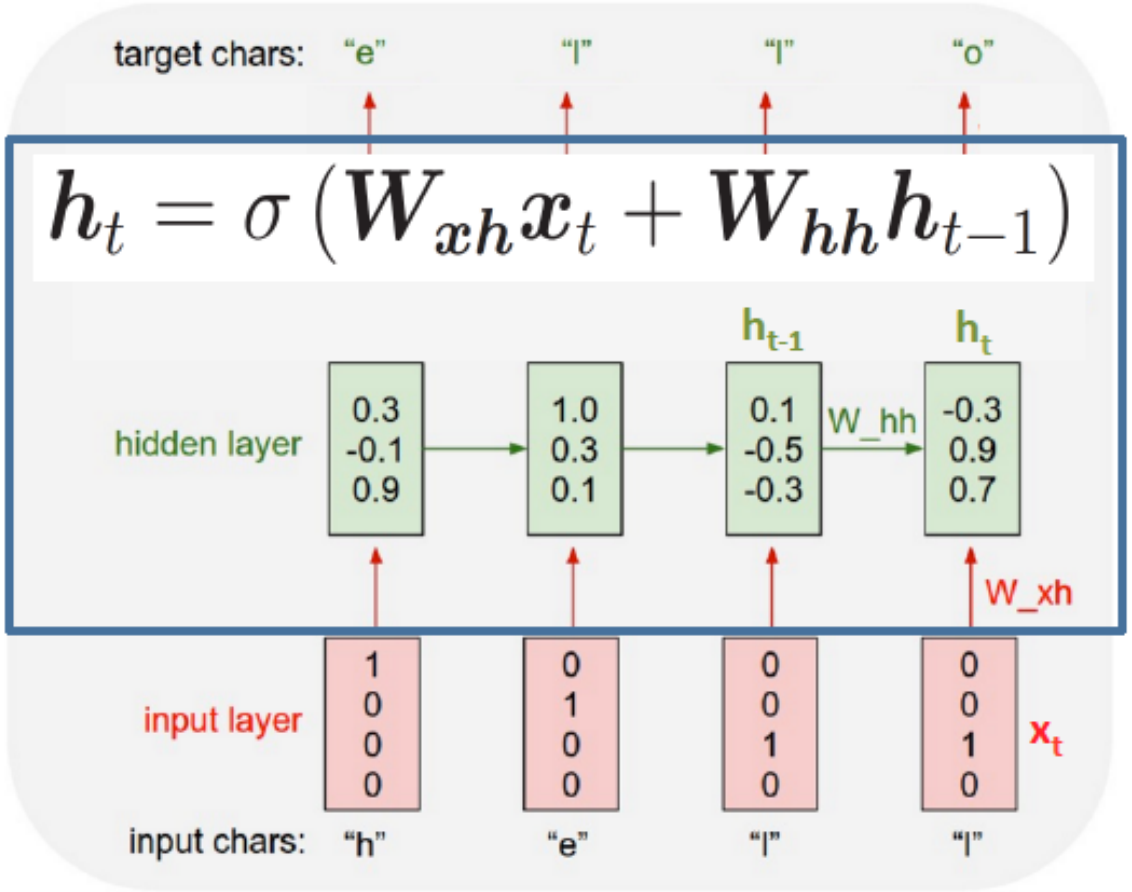
(Picture adapted from Andrej Karparthy)

RNN – *Hidden Layer*

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



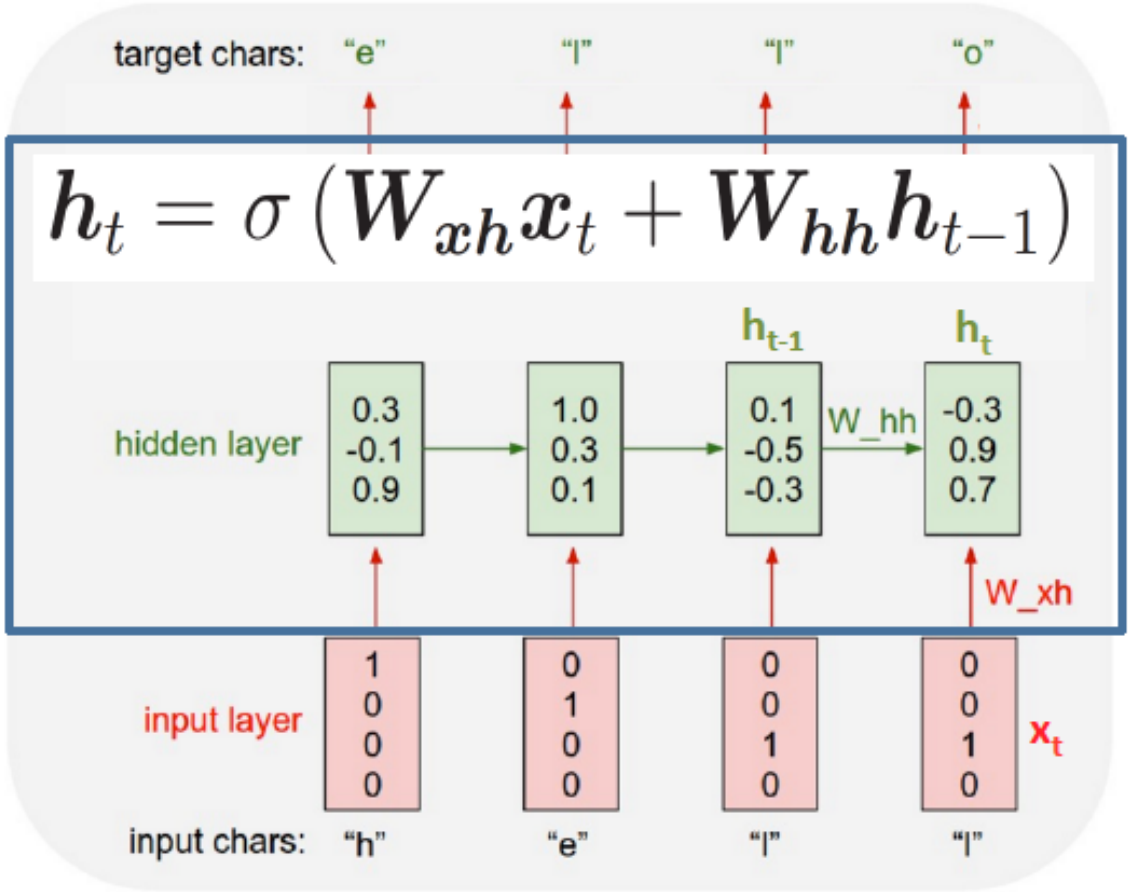
(Picture adapted from Andrej Karparthy)

RNN – *Hidden Layer*

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



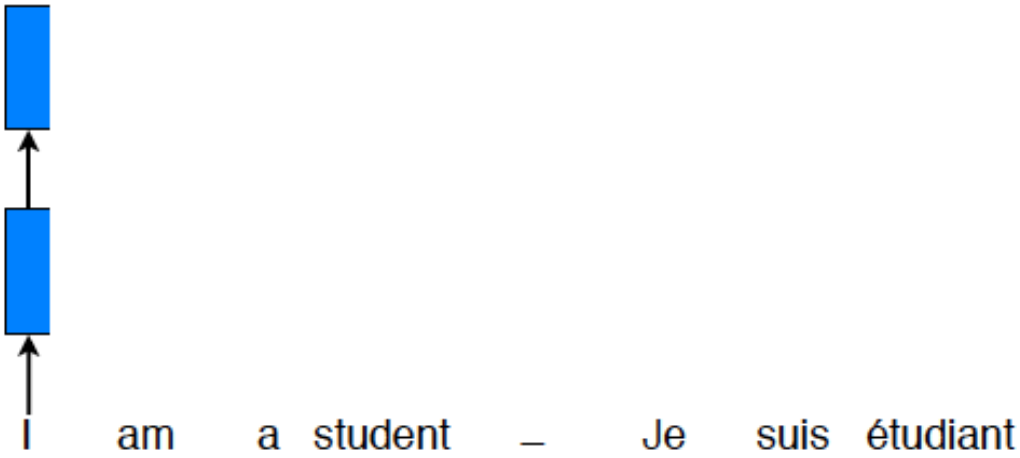
RNNs to represent sequences!

Neural Machine Translation (NMT)

I am a student _ Je suis étudiant

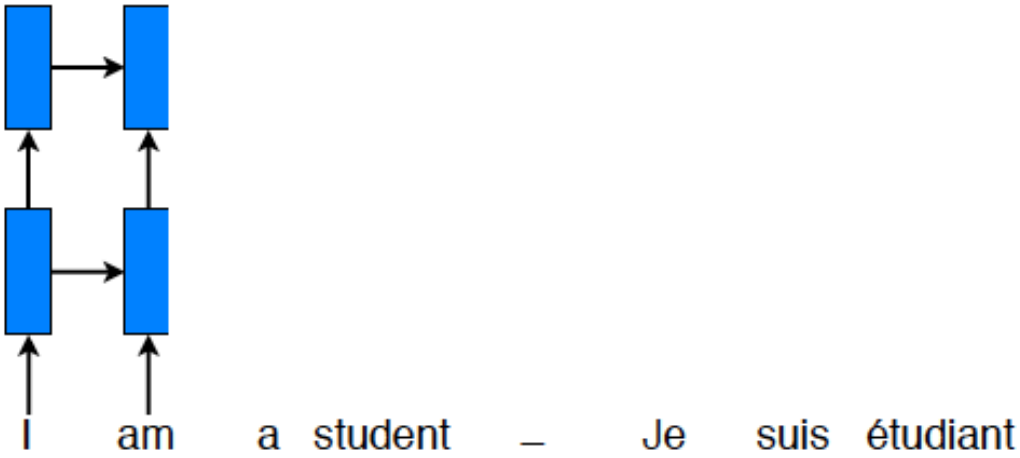
- Model P(target | source) directly.

Neural Machine Translation (NMT)



- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)

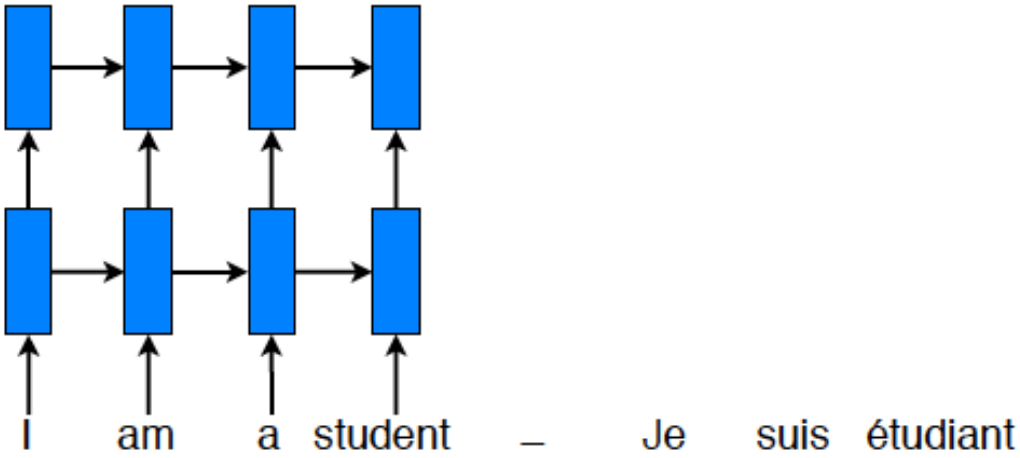


- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)

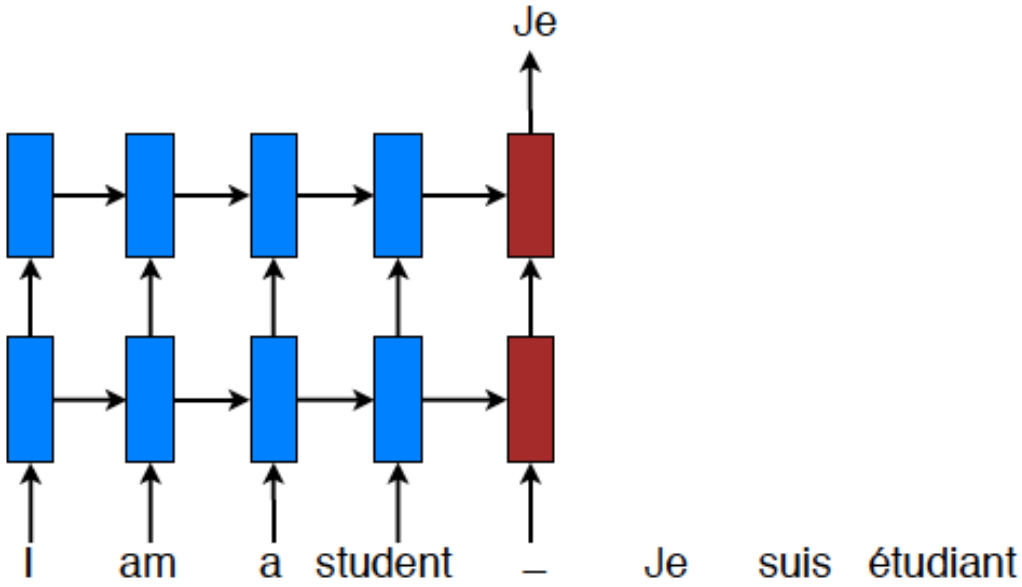
- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)



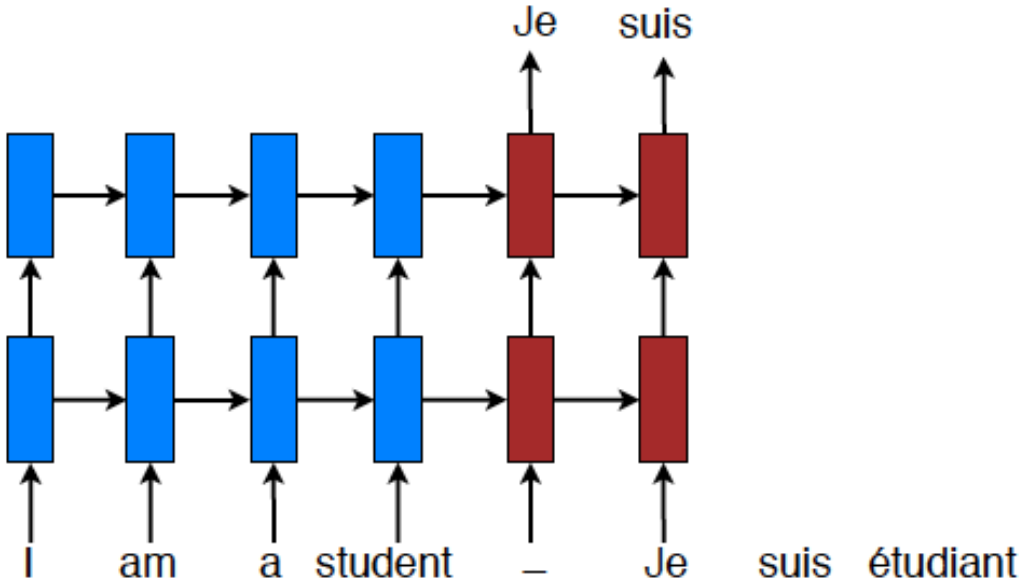
- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)



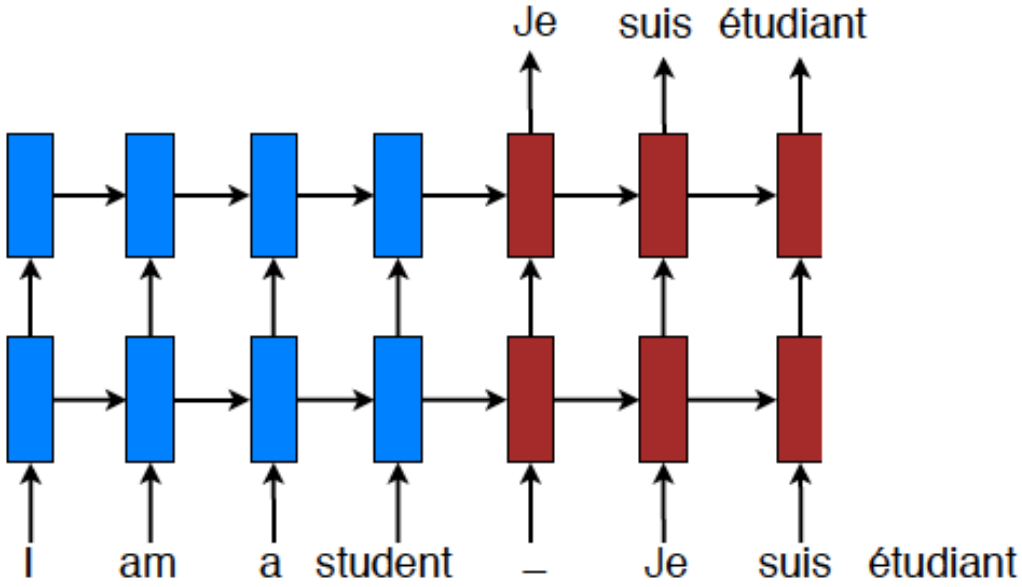
- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)



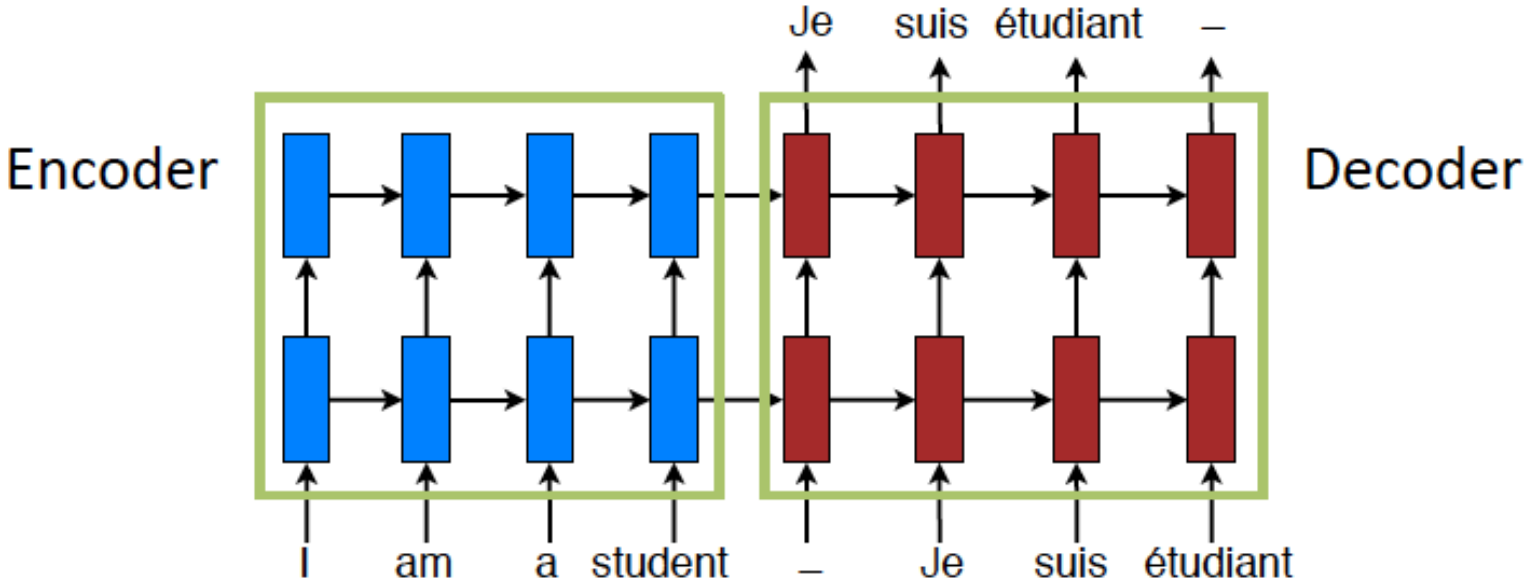
- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)



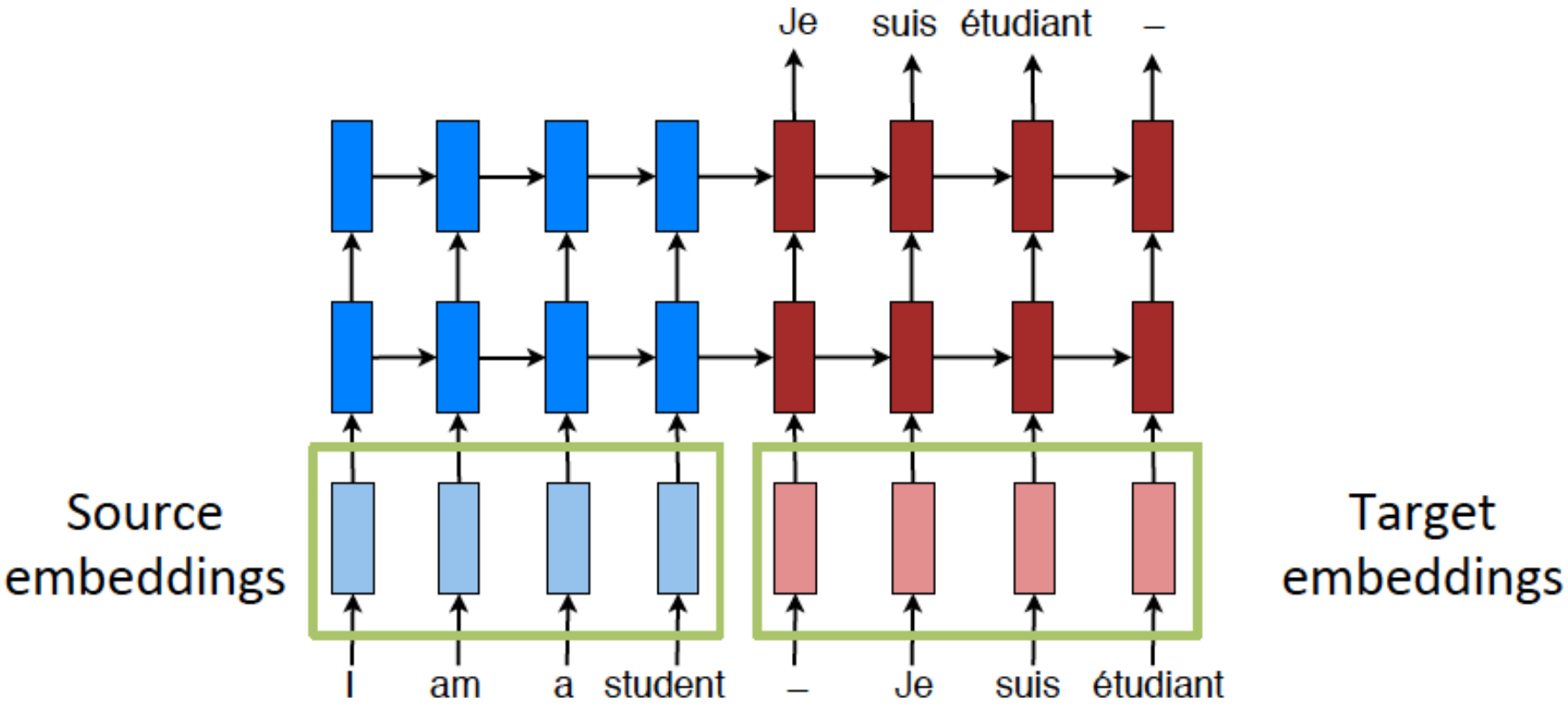
- RNNs trained end-to-end (Sutskever et al., 2014).

Neural Machine Translation (NMT)



- RNNs trained end-to-end (Sutskever et al., 2014).
- Encoder-decoder approach.

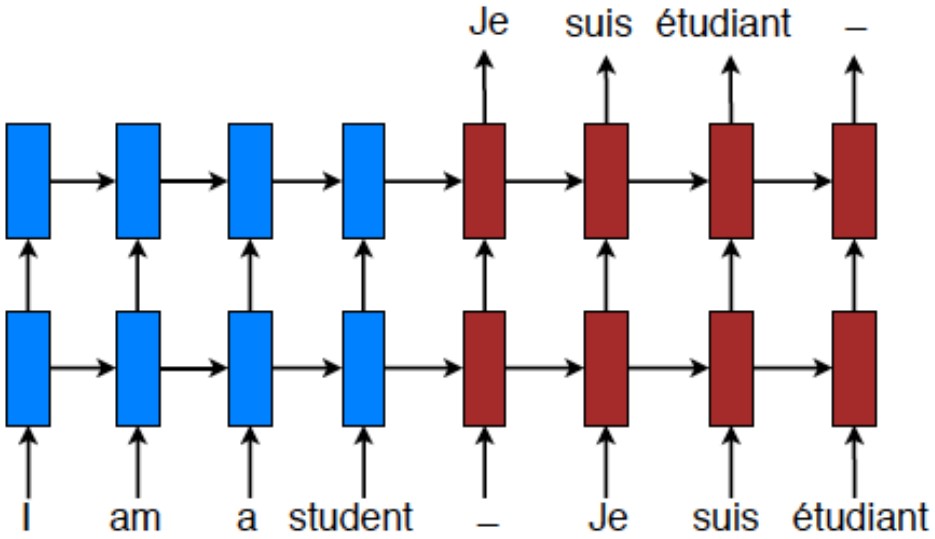
Word Embeddings



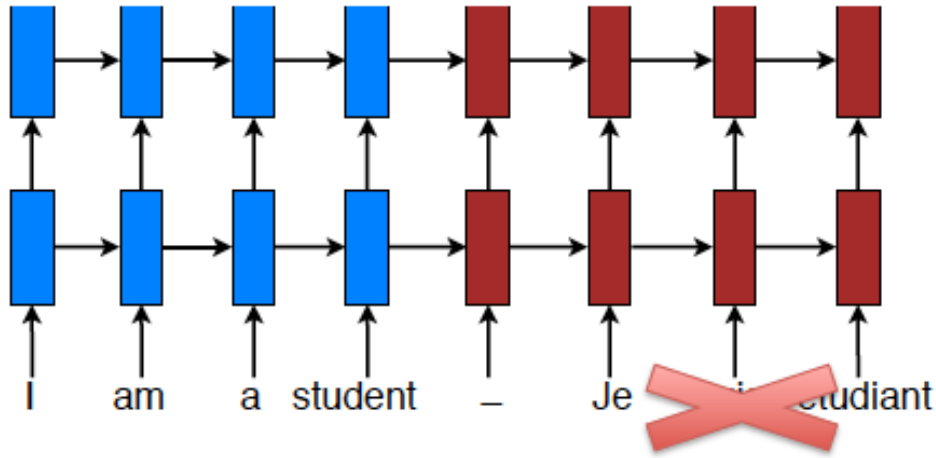
- Randomly initialized, one for each language.
  - Learnable parameters.

Training vs. Testing

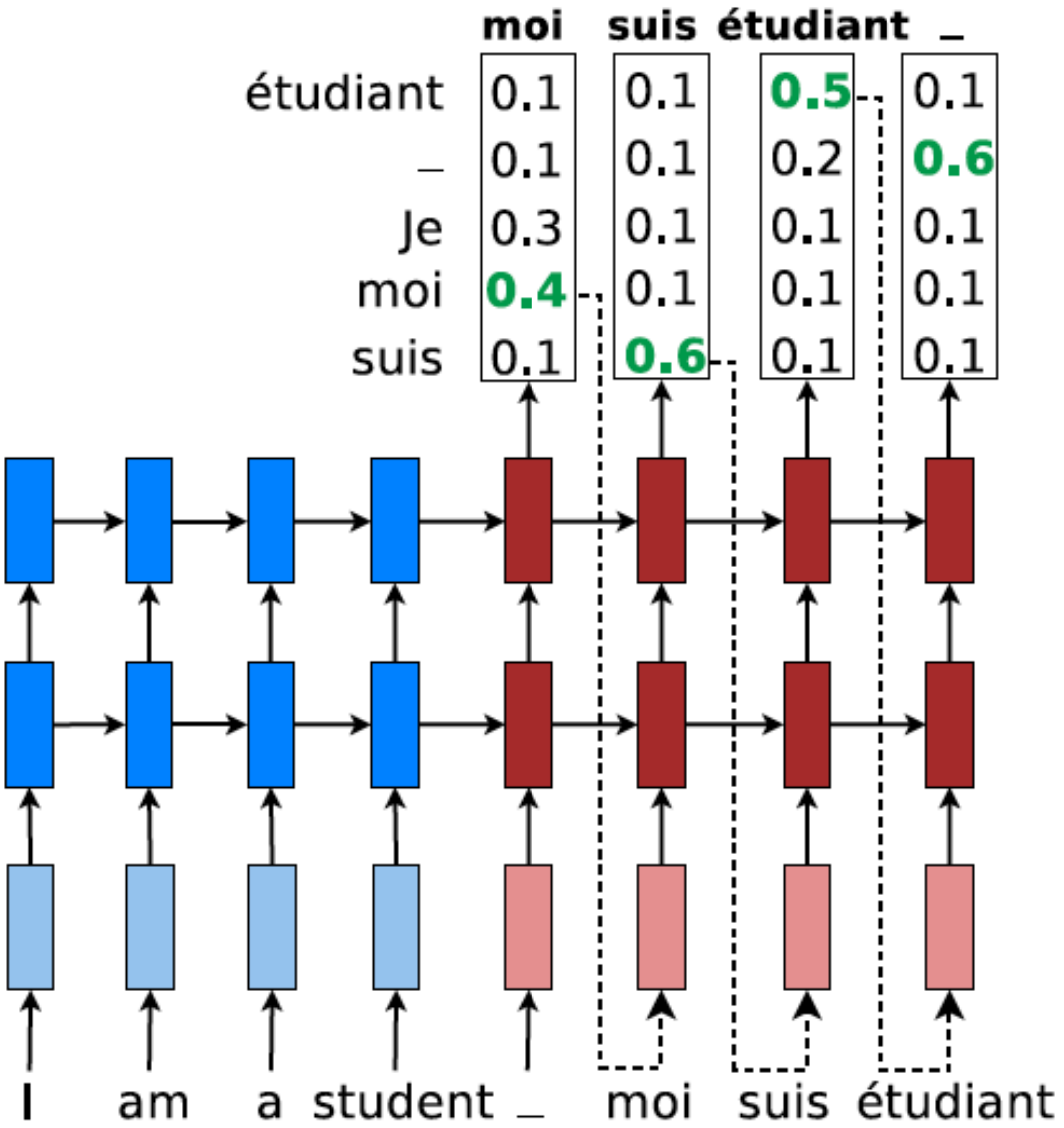
- *Training*
  - Correct translations are available.



- *Testing*
  - Only source sentences are given.



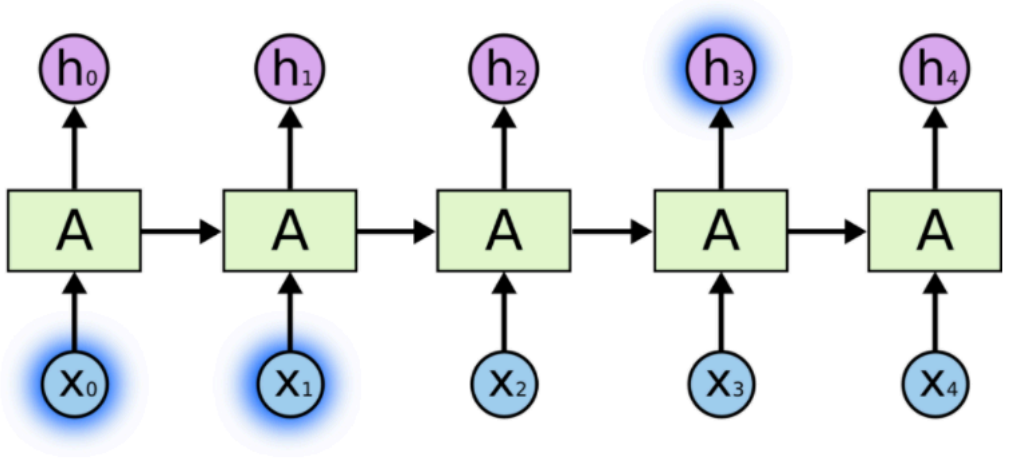
Testing



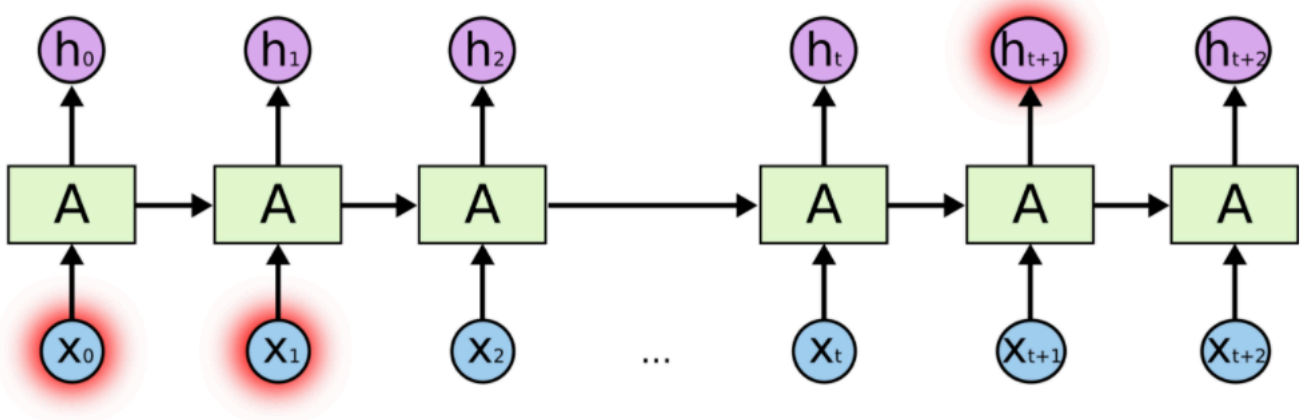
- Feed the most likely word

The problem of long term dependencies

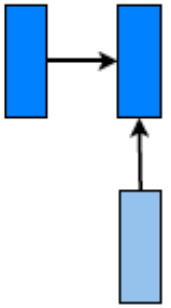
the clouds are in the sky

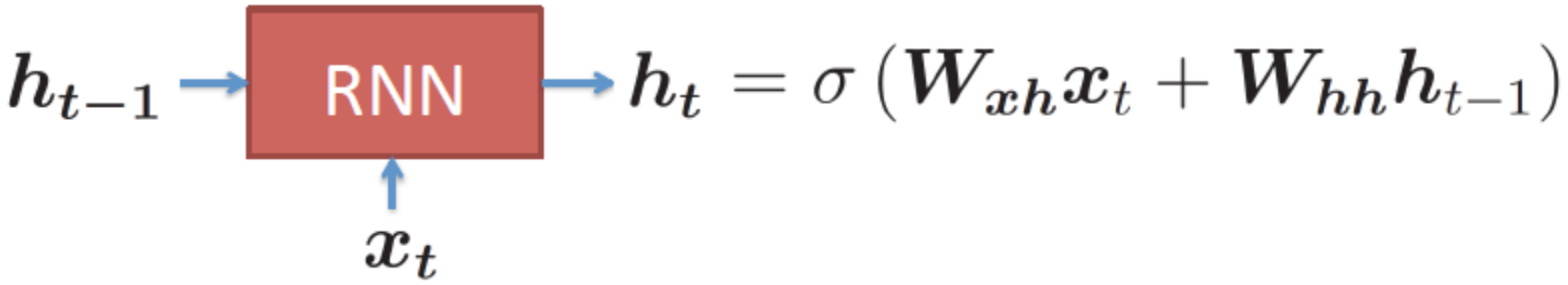


I grew up in France… I speak fluent French



Recurrent types – vanilla RNN





Vanishing gradient problem!

Vanishing gradients

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{W}_{\boldsymbol{xh}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hh}}\boldsymbol{h}_{t-1}\right)$$

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = \operatorname{diag}\left(\sigma'(\ldots)\right) \boldsymbol{W}_{\boldsymbol{hh}}{}^{\top}$$

Chain Rule

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \right\| \leq \gamma\lambda_1$$

Bound Rules

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-k}} \right\| \leq (\gamma\lambda_1)^k$$

Chain Rule

(Pascanu et al., 2013)

Vanishing gradients

$$\boldsymbol{h}_t = \sigma\left(\boldsymbol{W}_{\boldsymbol{xh}}\boldsymbol{x}_t + \boldsymbol{W}_{\boldsymbol{hh}}\boldsymbol{h}_{t-1}\right)$$

$$\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} = \operatorname{diag}\left(\sigma'(\ldots)\right) \boldsymbol{W}_{\boldsymbol{hh}}{}^{\top}$$

Chain Rule

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \right\| \leq \gamma\lambda_1$$

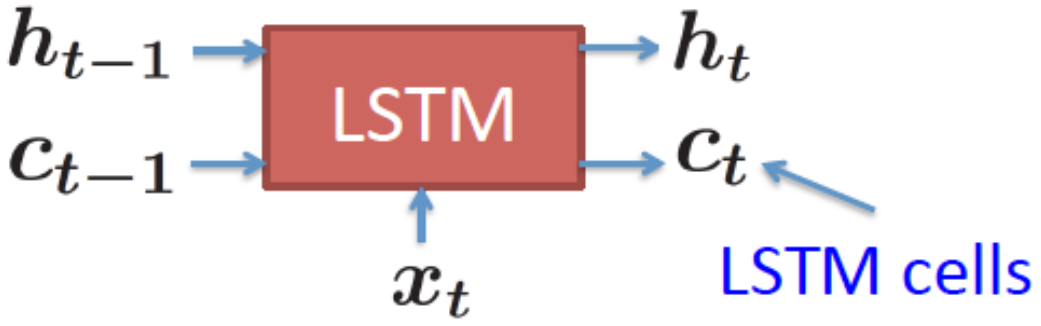
Bound Rules

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-k}} \right\| \leq (\gamma\lambda_1)^k \to 0 \text{ if } \lambda_1 < \frac{1}{\gamma}$$

Sufficient Cond

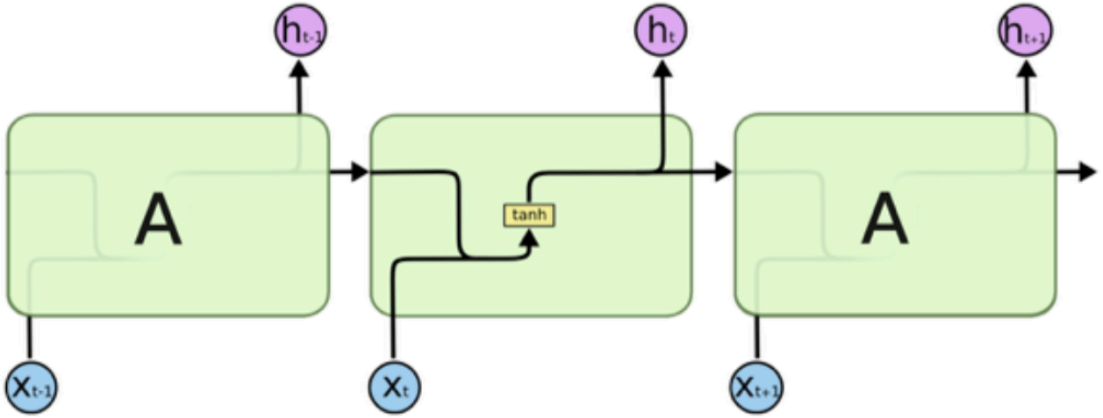
(Pascanu et al., 2013)

Recurrent types – LSTM

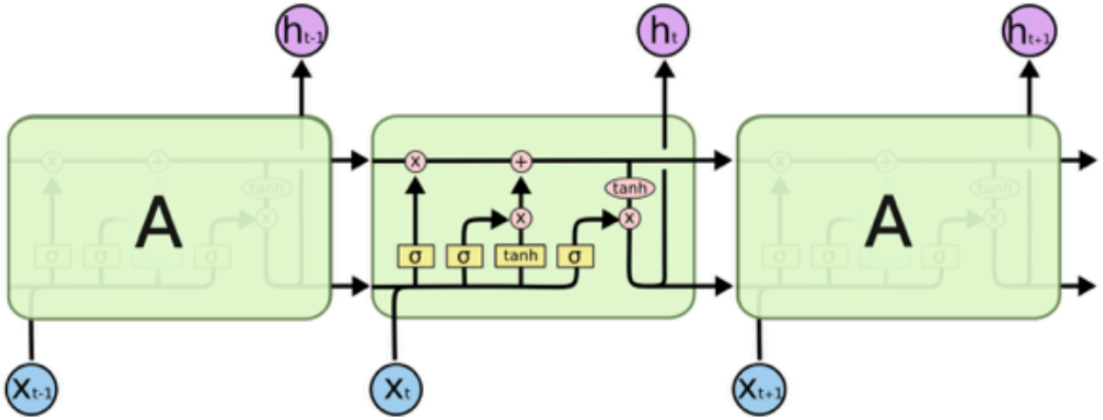


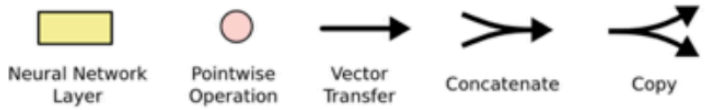
- Long-Short Term Memory (LSTM)
  - (Hochreiter & Schmidhuber, 1997)
- LSTM cells are additively updated
  - Make backprop through time easier.

RNN:

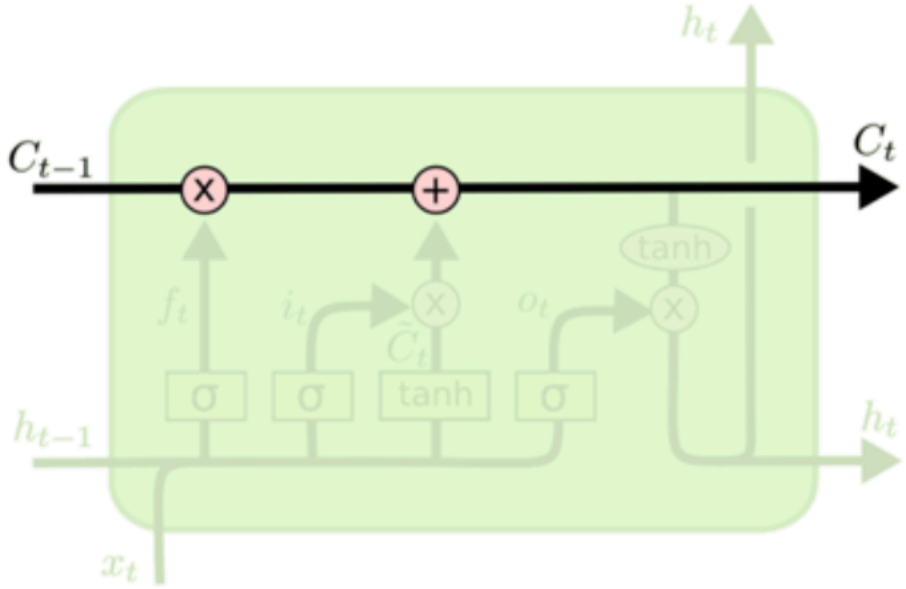

h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}

LSTM:

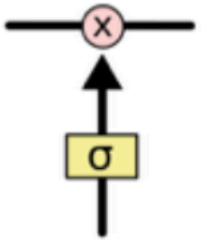

h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
X_{t-1}
X_t
X_{t+1}


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

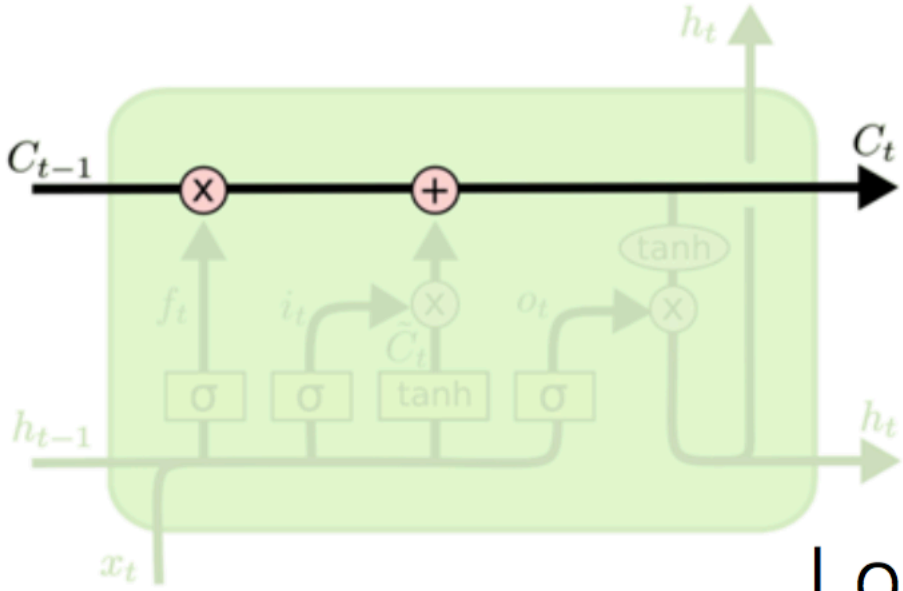
“The cell”



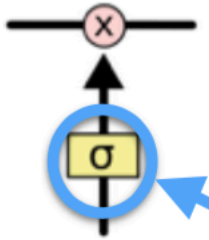
A Gate:



“The cell”

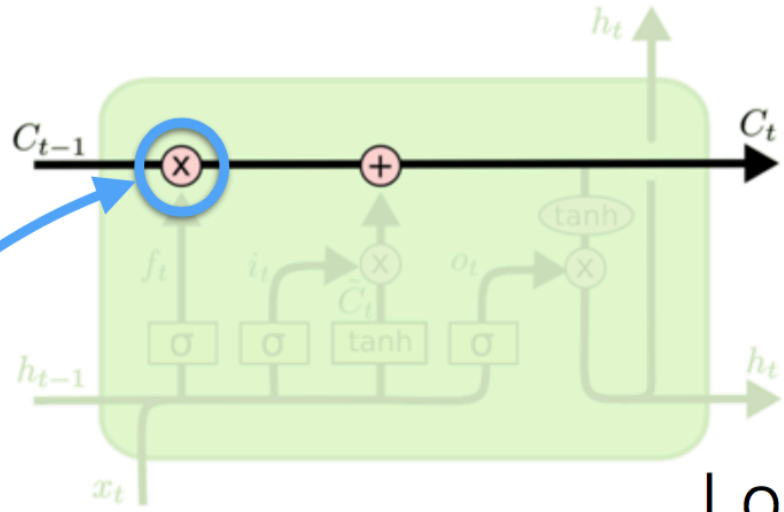


A Gate:

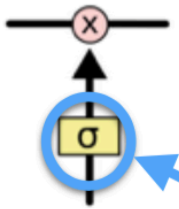


Logistic sigmoid (returns values between 0 and 1)

“The cell”



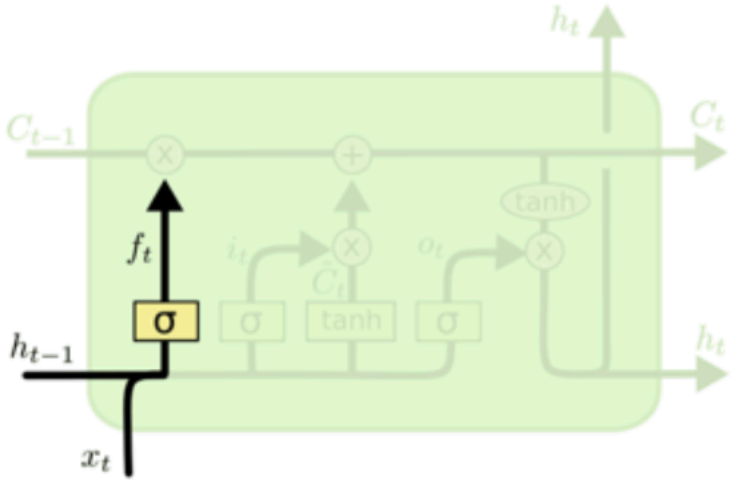
A Gate:



How much of the old state should we now forget?

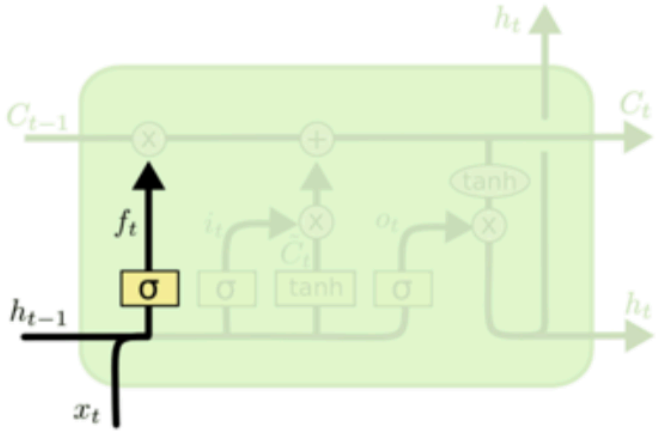
Logistic sigmoid (returns values between 0 and 1)

“Forget gate”



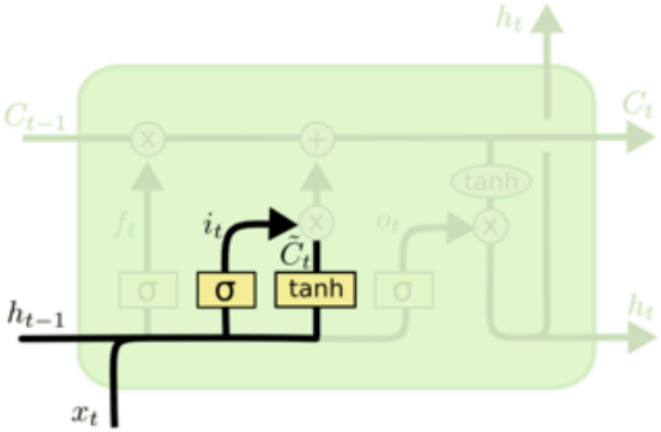
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

“Forget gate”



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

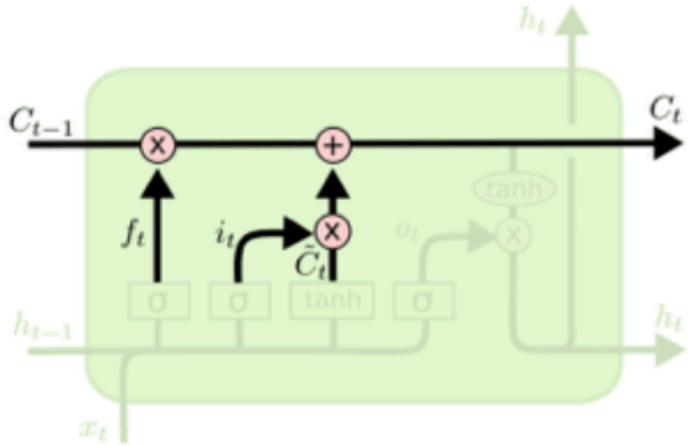
“Input gate”



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

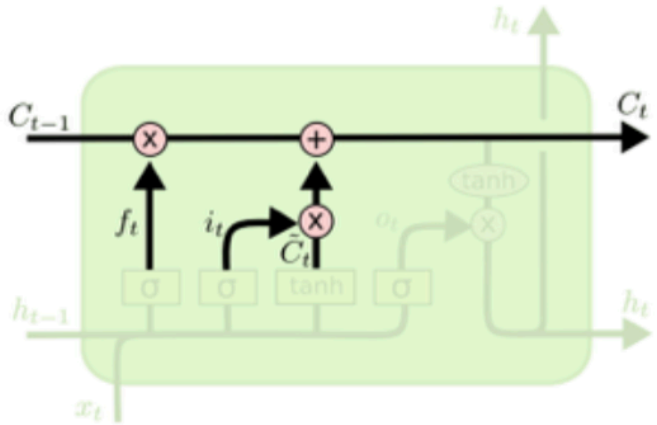
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cell update



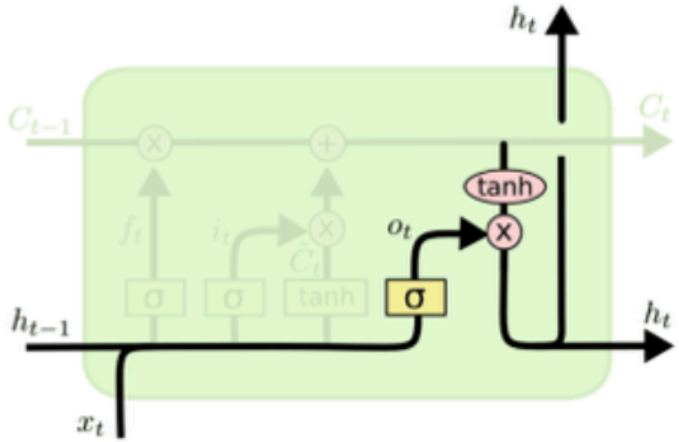
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell update



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

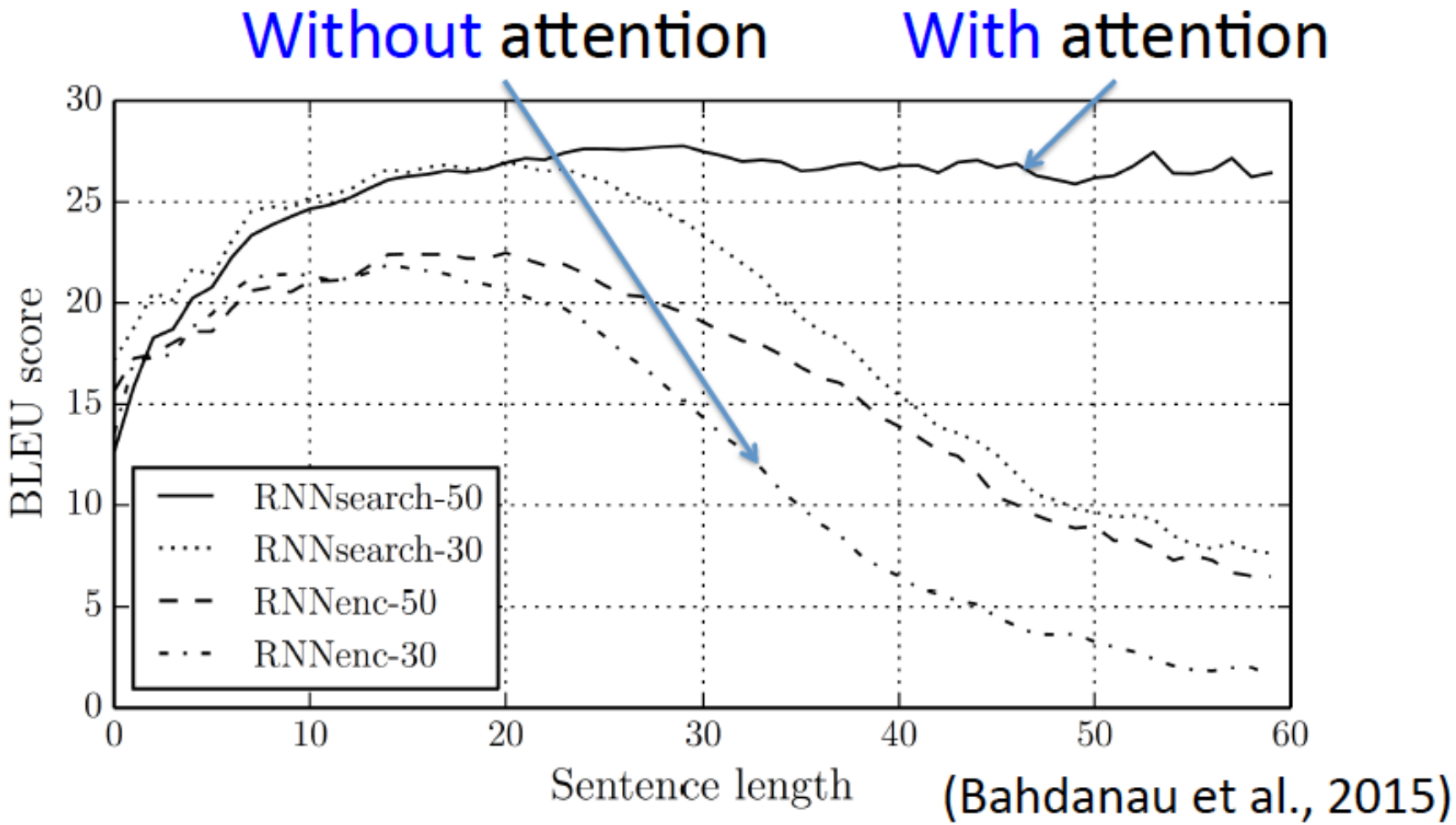
"Output gate" and output



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

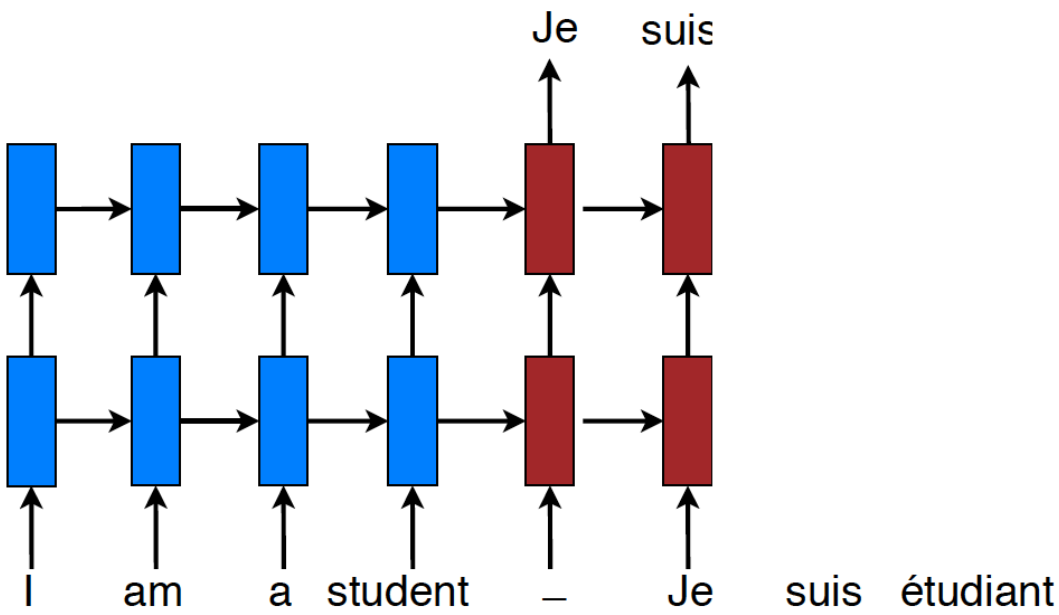
$$h_t = o_t * \tanh\left(C_t\right)$$

Sentence Length Problem



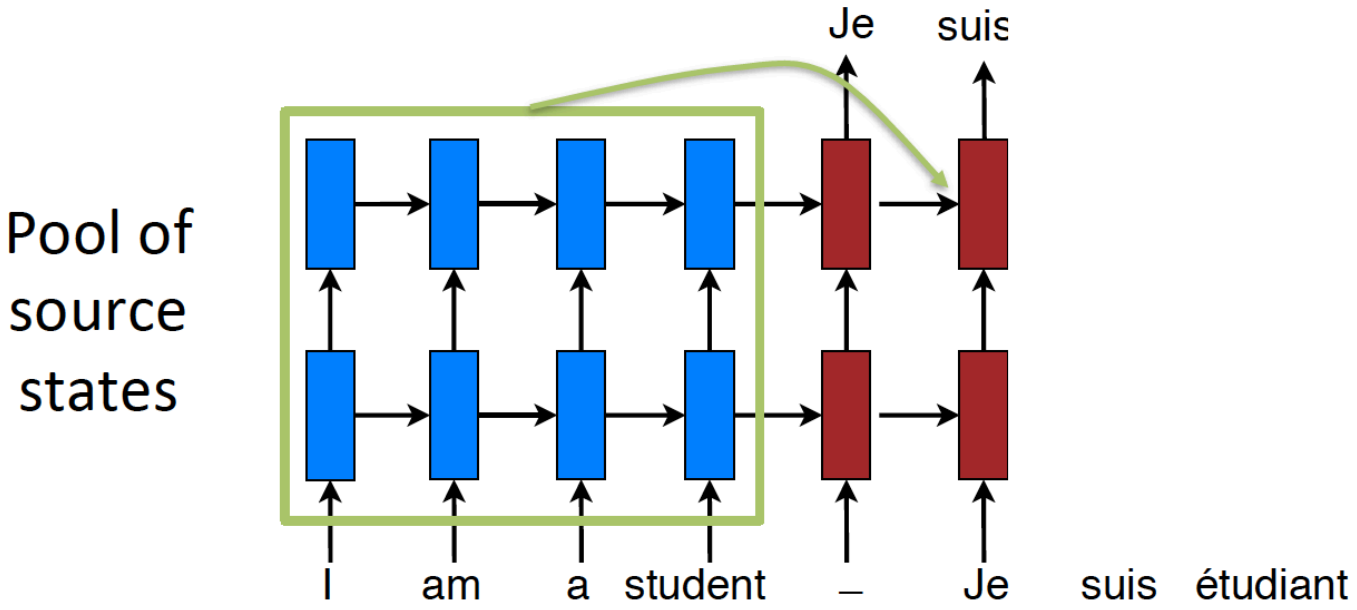
(Bahdanau et al., 2015)

Why?



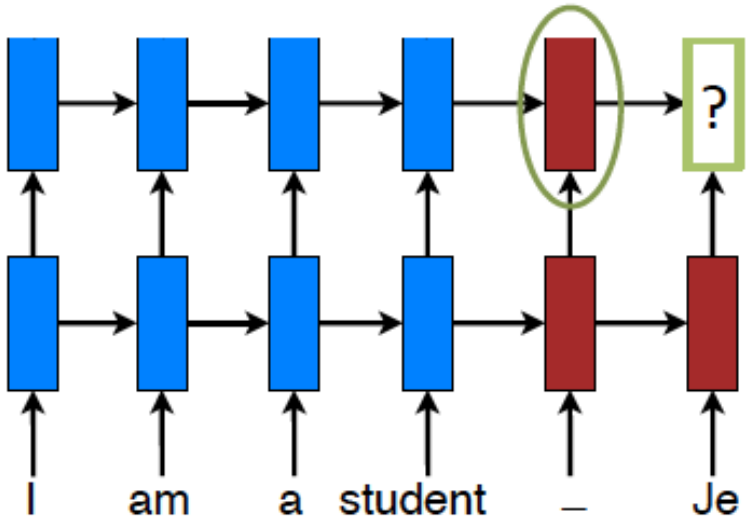
- A fixed-dimensional source vector.

Attention Mechanism



- Solution: random access memory
  - Retrieve as needed.

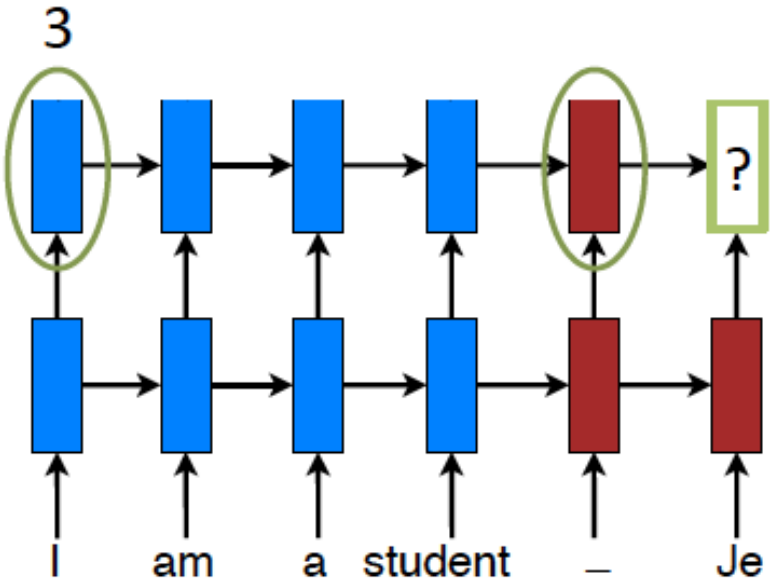
Attention Mechanism – *Scoring*



- Compare target and source hidden states.

Attention Mechanism – *Scoring*

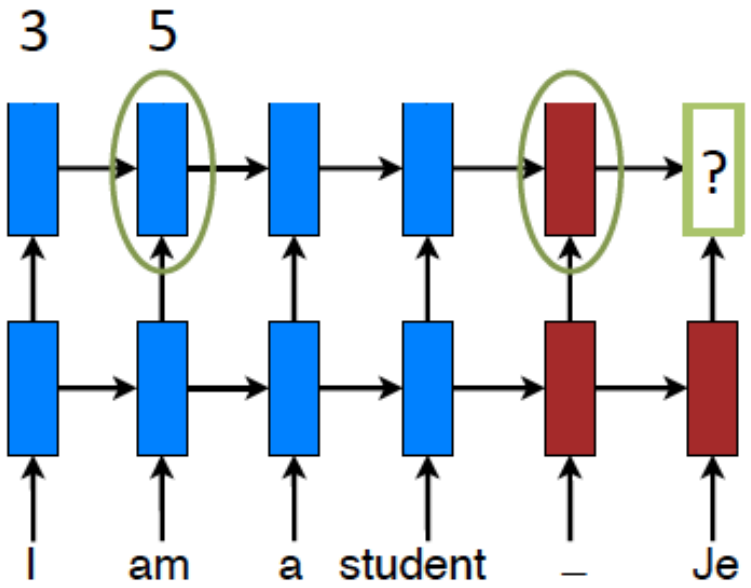
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



- Compare target and source hidden states.

Attention Mechanism – *Scoring*

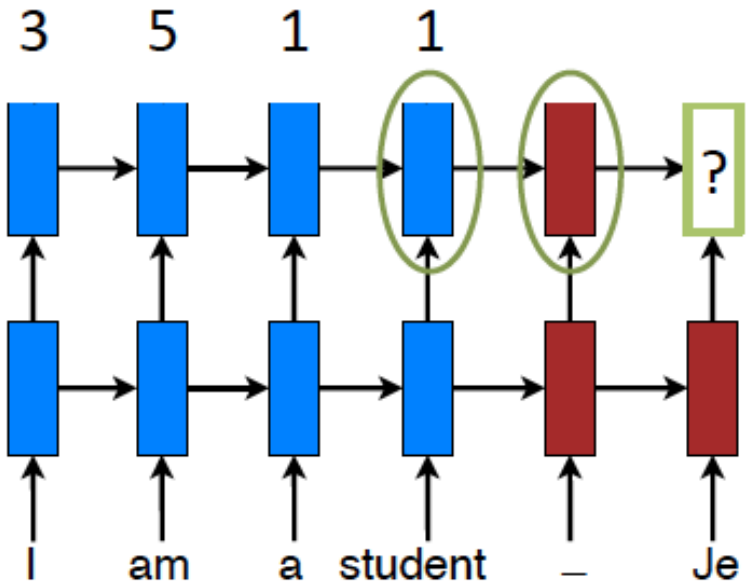
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



- Compare target and source hidden states.

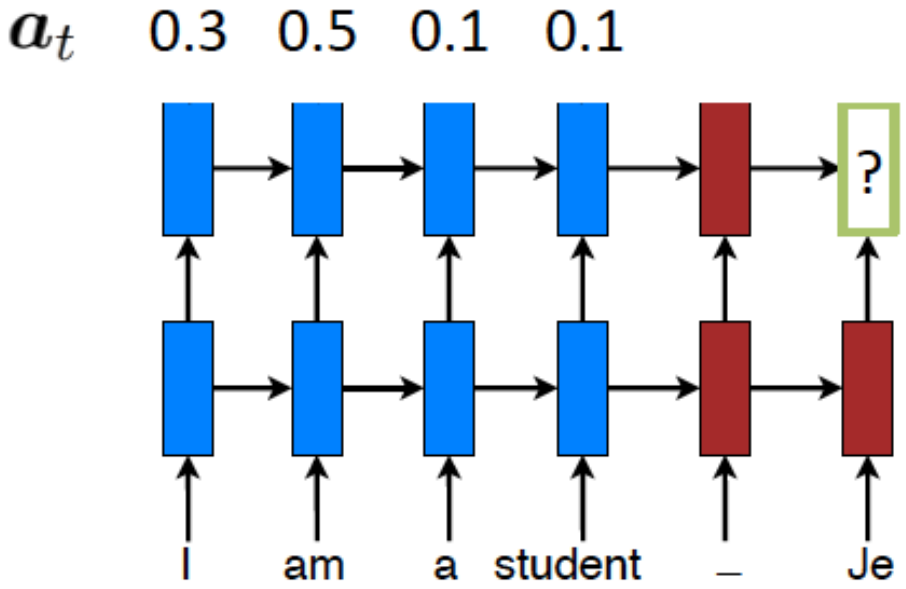
Attention Mechanism – *Scoring*

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



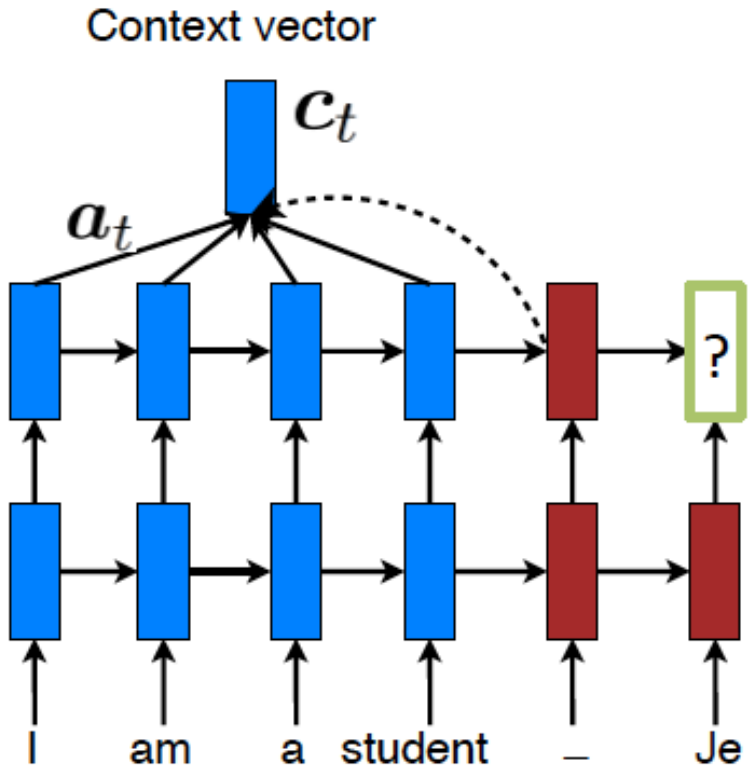
- Compare target and source hidden states.

Attention Mechanism – *Normalization*



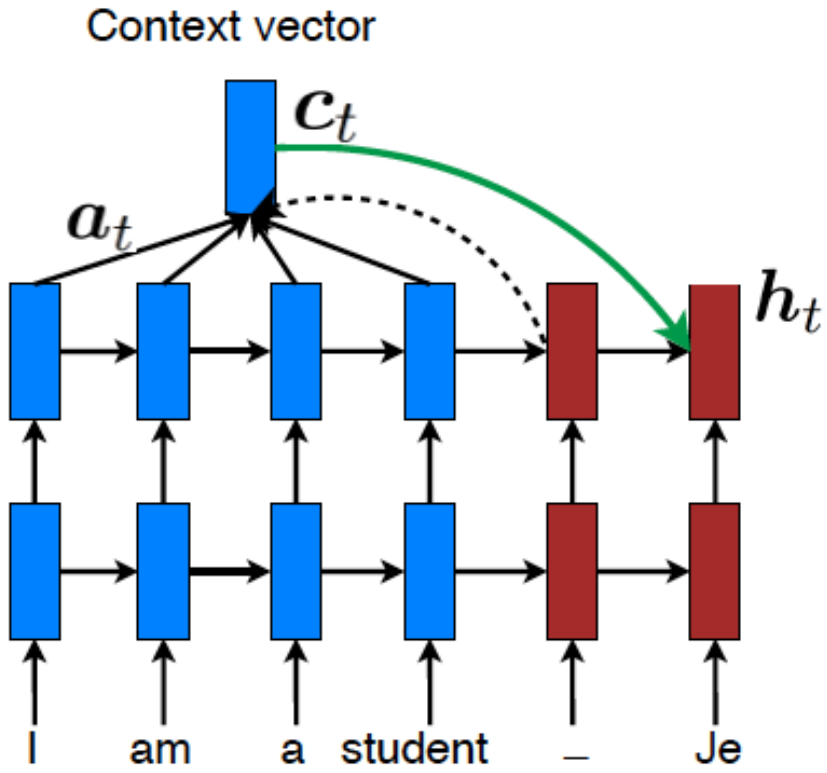
- Convert into alignment weights.

Attention Mechanism – *Context vector*



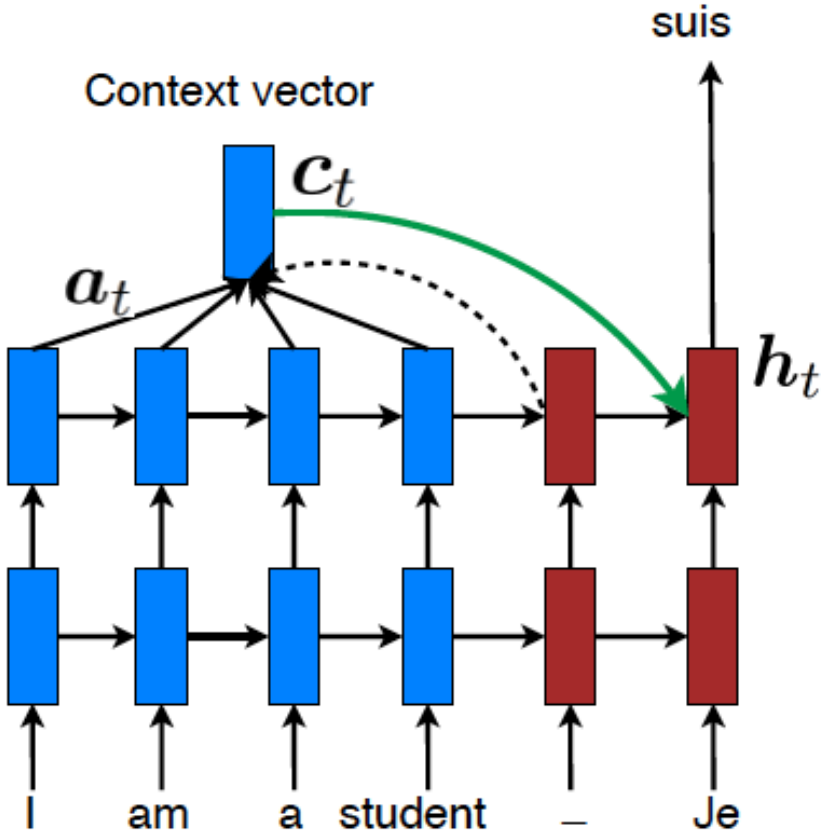
- Build context vector: weighted average.

Attention Mechanism – *Hidden state*



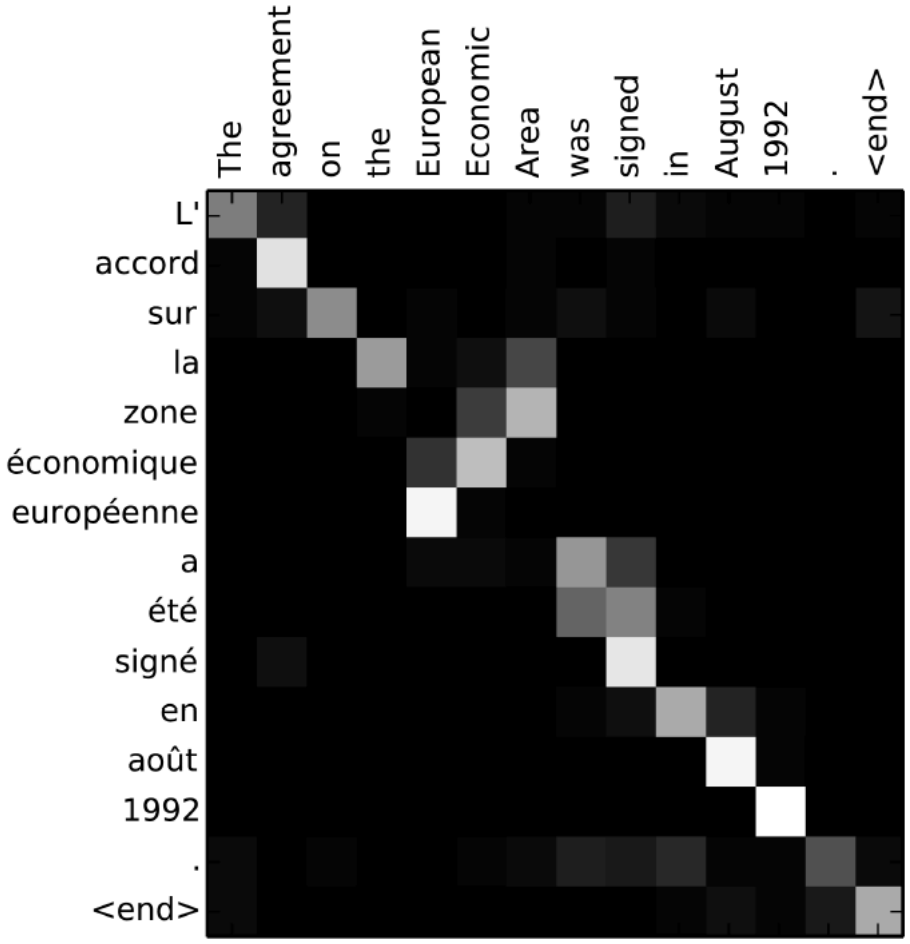
- Compute the next hidden state.

Attention Mechanism – *Predict*



- Predict the next word.

Alignments as a by-product



(Bahdanau et al., 2015)

Acknowledgment

- Chris Dyer (CMU)
- Chris Callison-Burch (JHU)
- Adam Lopez (Edinburgh)
- Philip Koehn
- Thang Luong (Stanford)
- … many others