# Machine Translation

CMSC 723 / LING 723 / INST 725

MARINE CARPUAT
marine@cs.umd.edu

# Today: an introduction to machine translation

- The **noisy channel model** decomposes machine translation into
  - Word alignment
  - Language modeling

- How can we automatically **align words** within sentence pairs? We'll rely on:
  - **probabilistic modeling**
    - IBM1 and variants [Brown et al. 1990]
  - **unsupervised learning**
    - Expectation Maximization algorithm

# MACHINE TRANSLATION AS A NOISY CHANNEL MODEL

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

| |
|---|
| The flowers bloom in the spring.<br>कलियाँ वसन्त में खिलती हैं । |
| Sita came yesterday.<br>सीता कल आयी थी । |
| The gymnast makes springing up to the bar look easy.<br>कसरतबाज डंडे के ऊपर से कूदने के कार्य को आसान बना देता है । |
| It rained yesterday.<br>कल बारिश हुई थी । |
| School will commence tomorrow.<br>विद्यालय कल से आरम्भ होगा । |
| With a spring the cat reached the branch.<br>वह बिल्ली एक टहनी पर कूद गयी । |
| I will come tomorrow.<br>मैं कल आऊँगा । |

# Rosetta Stone

Egyptian hieroglyphs

Demotic

Greek

# Warren Weaver (1947)

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

# Weaver's intuition formalized as a Noisy Channel Model

- Translating a French sentence **f** is finding the English sentence **e** that maximizes **P(elf)**
- The noisy channel model breaks down **P(elf)** into two components

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \overbrace{P(F|E)}^{\text{translation model}} \overbrace{P(E)}^{\text{language model}}$$

# Translation Model & Word Alignments

- How can we define the translation model p(f|e) between a French sentence f and an English sentence e?
- Problem: there are many possible sentences!
- Solution: break sentences into words
  - model mappings between word position to represent translation
  - Just like in the Centauri/Arcturian example

# PROBABILISTIC MODELS OF WORD ALIGNMENT

# Defining a probabilistic model for word alignment

Probability lets us

1) Formulate a model of pairs of sentences
2) Learn an instance of the model from data
3) Use it to infer alignments of new inputs

# Recall language modeling

Probability lets us

1) Formulate a model of a sentence

   e.g, bi-grams

2) Learn an instance of the model from data

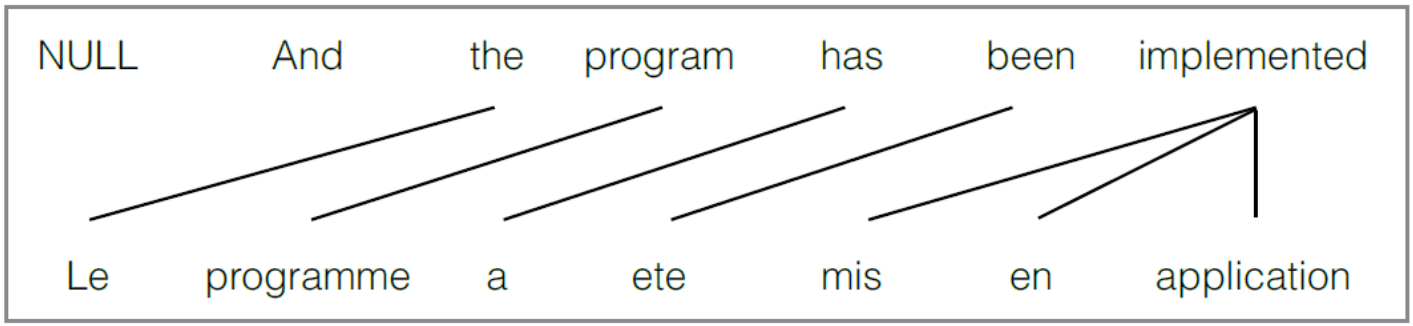
$$\hat{p}_{\text{MLE}}(\texttt{call} \mid \texttt{friends}) = \frac{\text{count}(\texttt{friends call})}{\text{count}(\texttt{friends})}$$

3) Use it to score new sentences

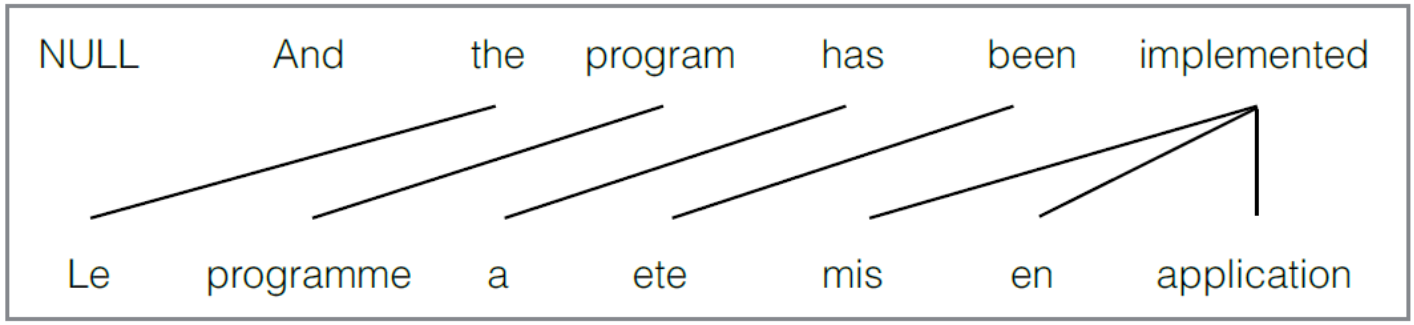
# How can we model p(f|e)?

- We'll describe the word alignment models introduced in early 90s at IBM

- Assumption: each French word f is aligned to exactly one English word e
  - Including NULL

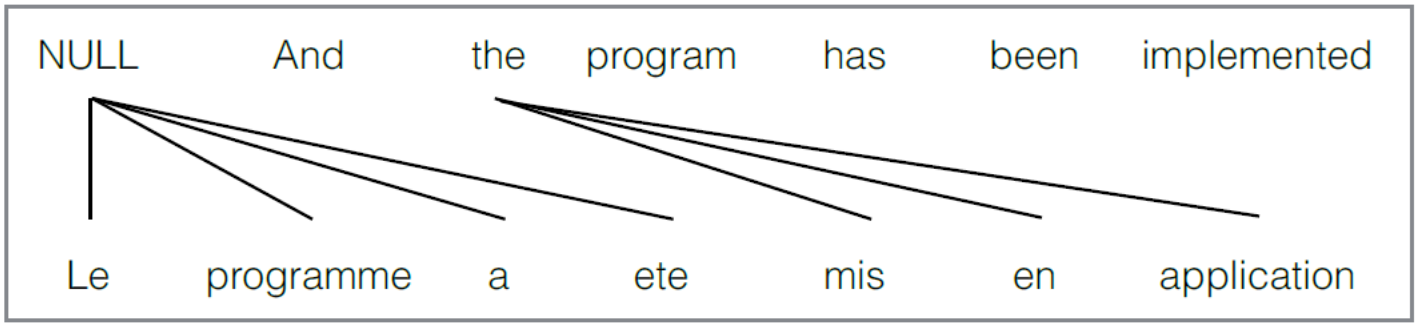


# Word Alignment Vector Representation




- Alignment vector a = [2,3,4,5,6,6,6]
  - length of a = length of sentence f
  - ai = j if French position i is aligned to English position j

# Word Alignment Vector Representation




- Alignment vector a = [0,0,0,0,2,2,2]

# How many possible alignments?

- How many possible alignments for (f,e) where
  - f is French sentence with m words
  - e is an English sentence with l words

- For each of m French words, we choose an alignment link among (l+1) English words

- Answer: $(l+1)^m$

# Formalizing the connection between word alignments & the translation model

$$p(f_1, f_2, \ldots, f_m \mid e_1, e_2, \ldots, e_l, m)$$
$$= \sum_{a \in A} p(f_1, \ldots, f_m, a_1, \ldots, a_m \mid e_1, \ldots, e_l, m)$$

- We define a conditional model
  - Projecting word translations
  - Through alignment links

# IBM Model 1: generative story

- Input
  - an English sentence of length l
  - a length m

- For each French position $i$ in 1..m
  - Pick an English source index j $q(j \mid i, l, m) = \frac{1}{l+1}$
  - Choose a translation $t(f_i \mid e_{a_i})$

# IBM Model 1: generative story

- Input
  - an English sentence of length l
  - a length m

Alignment is based on word positions, not word identities

Alignment probabilities are UNIFORM

- For each French position $i$ in 1..m
  - Pick an English source index j $q(j \mid i, l, m) = \frac{1}{l+1}$
  - Choose a translation $t(f_i \mid e_{a_i})$

Words are translated independently

# IBM Model 1: Parameters

- t(f|e)
  - Word translation probability table
  - for all words in French & English vocab

| f | e | p(f \| e) |
|---|---|---|
| le | the | 0.42 |
| la | the | 0.4 |
| programme | the | 0.001 |
| a | has | 0.78 |
| … | … | … |

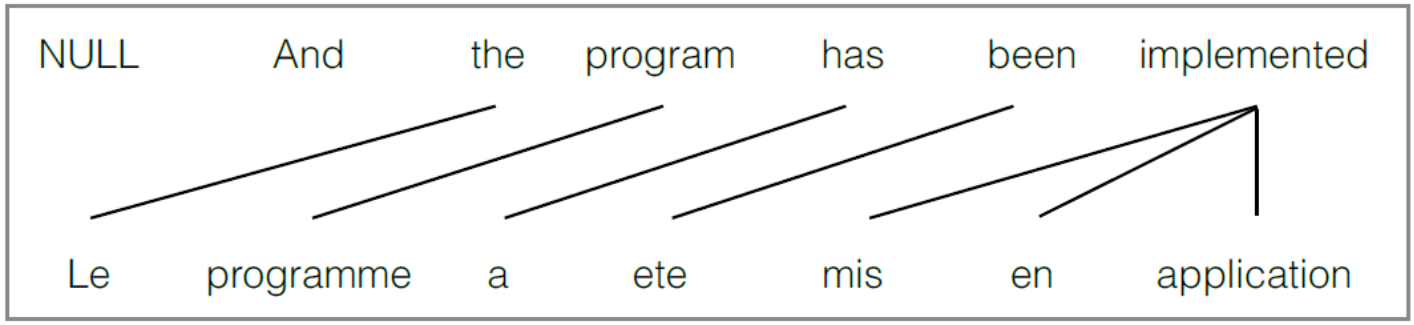
# IBM Model 1: generative story

- Input
  - an English sentence of length l
  - a length m

- For each French position $i$ in 1..m
  - Pick an English source index j $q(j \mid i, l, m) = \frac{1}{l+1}$
  - Choose a translation $t(f_i \mid e_{a_i})$

$$p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l, m) = \prod_{i=1}^{m} q(a_i | i, l, m) t(f_i | e_{a_i})$$

# IBM Model 1: Example




- Alignment vector a = [2,3,4,5,6,6,6]
- P(f,a|e)?

# Improving on IBM Model 1: IBM Model 2

- Input
  - an English sentence of length l
  - a length m

Remove assumption that q is uniform

- For each French position $i$ in 1..m
  - Pick an English source index j $q(j \mid i, l, m)$
  - Choose a translation $t(f_i \mid e_{a_i})$

# IBM Model 2: Parameters

- q(j|i,l,m)
  - now a table
  - not uniform as in IBM1

- How many parameters are there?

| j | q(j \| 1, 6, 7) |
|---|---|
| 1 | 0.27 |
| 2 | 0.14 |
| … | … |
| 48 | 1E-75 |

# Defining a probabilistic model for word alignment

Probability lets us

1) Formulate a model of pairs of sentences

   **=> IBM models 1 & 2**

2) Learn an instance of the model from data

3) Use it to infer alignments of new inputs

# 2 Remaining Tasks

**Inference**

- Given
  - a sentence pair (e,f)
  - an alignment model with parameters t(e|f) and q(j|i,l,m)
- What is the most probable alignment a?

**Parameter Estimation**

- Given
  - training data (lots of sentence pairs)
  - a model definition
- how do we learn the parameters t(e|f) and q(j|i,l,m)?

# Inference

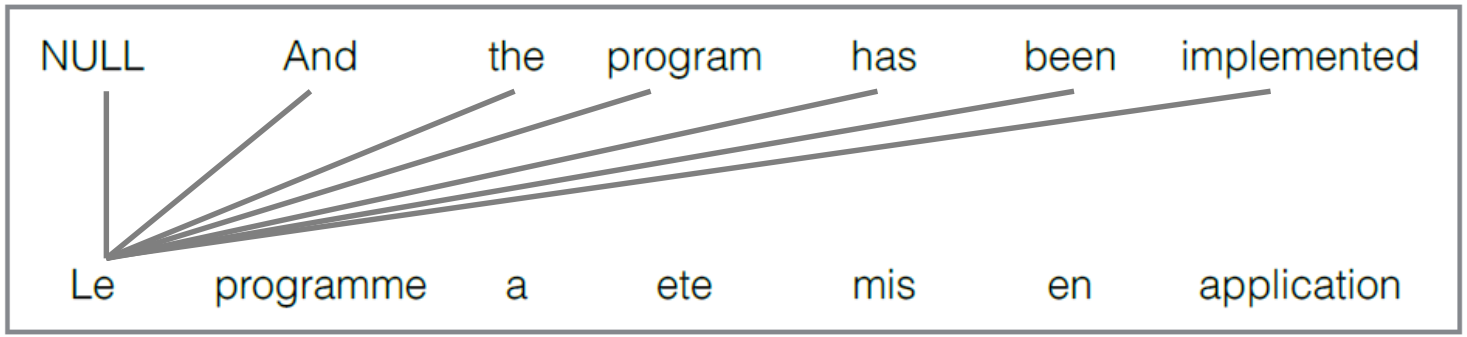
- Inputs
  - Model parameter tables for t and q
  - A sentence pair

| NULL | And | the | program | has | been | implemented |
|---|---|---|---|---|---|---|
| Le | programme | a | ete | mis | en | application |

- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Inference

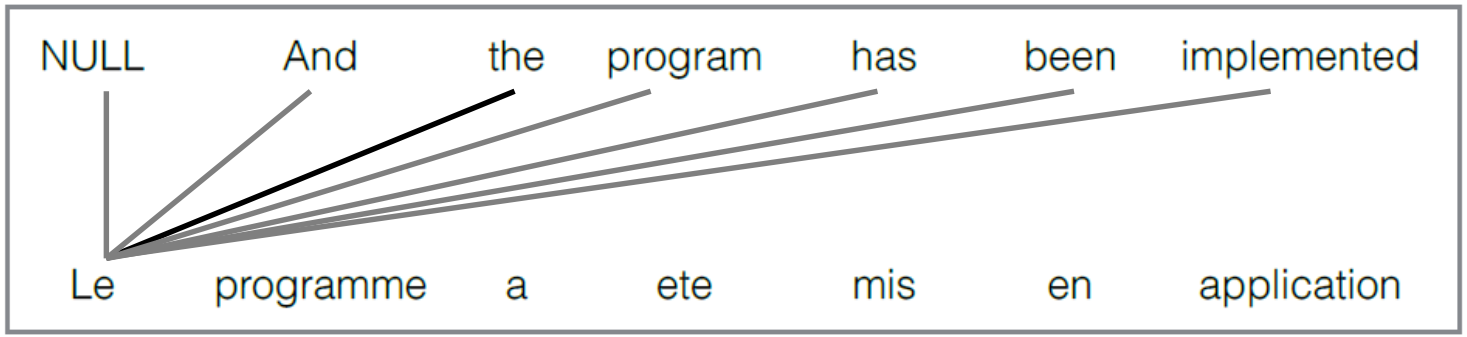
- Inputs
  - Model parameter tables for t and q
  - A sentence pair




- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Inference

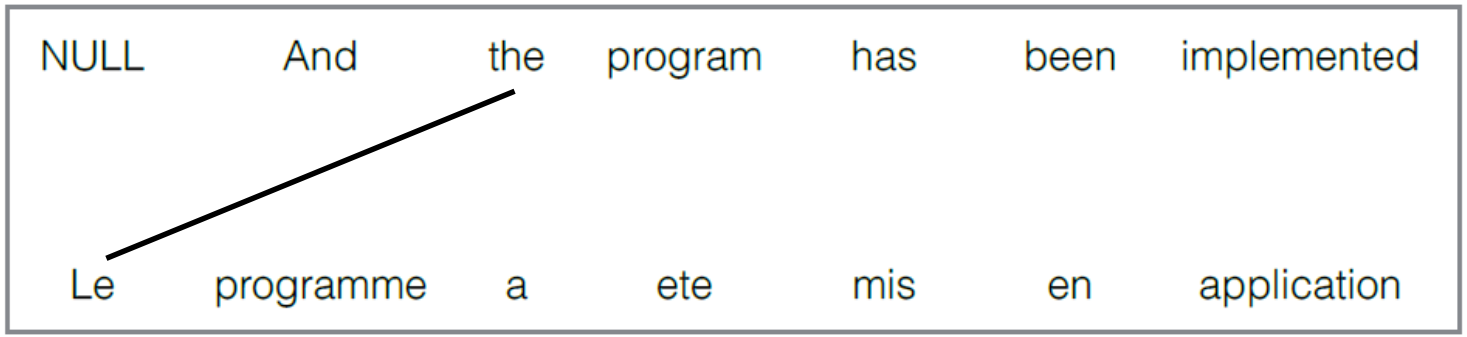
- Inputs
  - Model parameter tables for t and q
  - A sentence pair




- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Inference

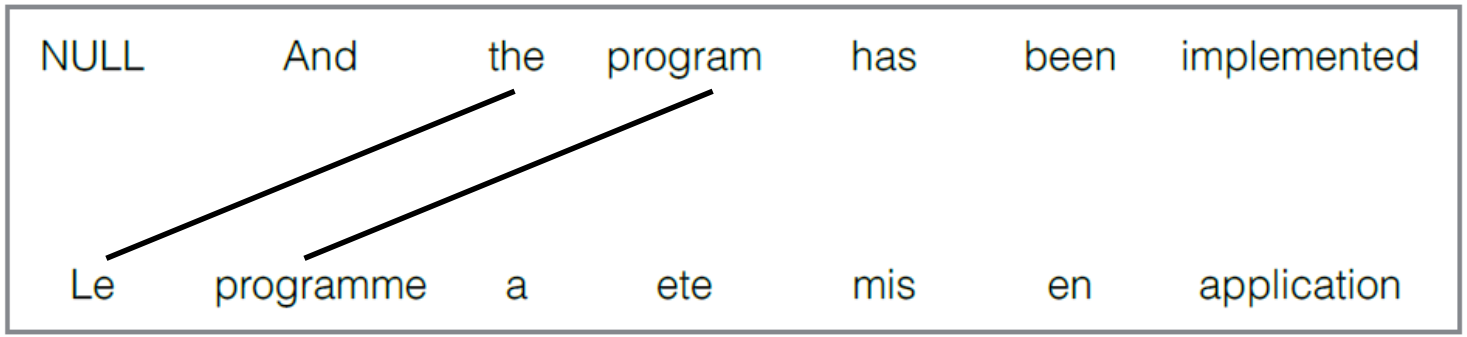
- Inputs
  - Model parameter tables for t and q
  - A sentence pair




- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Inference

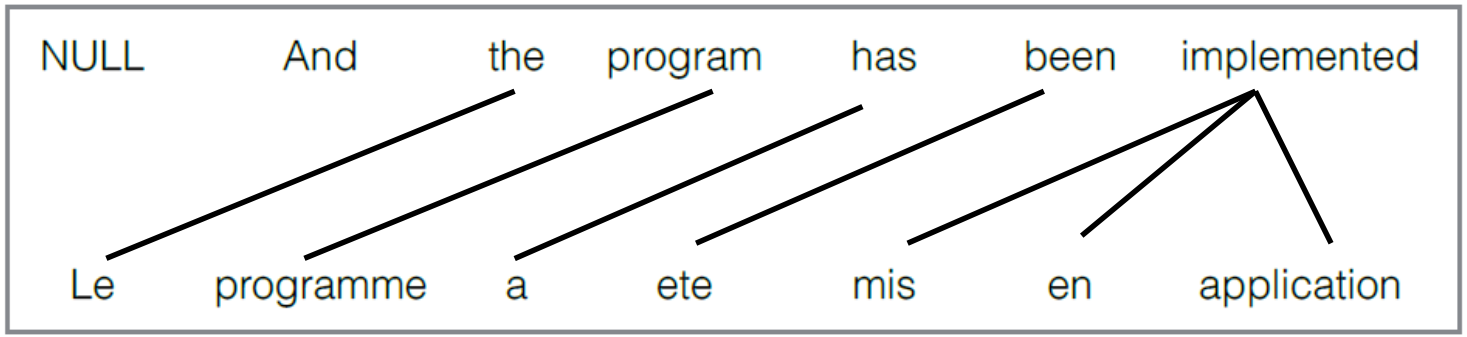
- Inputs
  - Model parameter tables for t and q
  - A sentence pair




- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Inference

- Inputs
  - Model parameter tables for t and q
  - A sentence pair

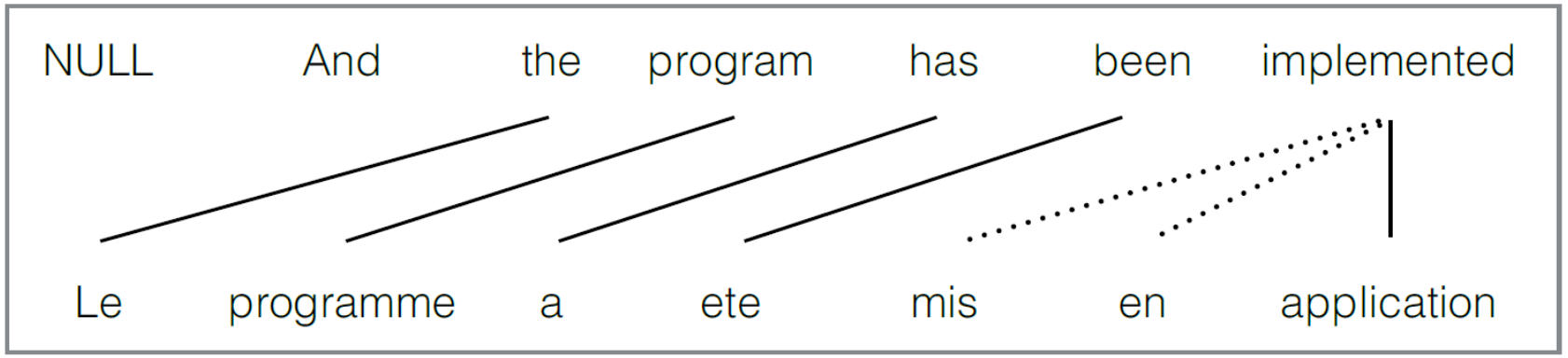



- How do we find the alignment a that maximizes P(e,a|f)?
  - Hint: recall independence assumptions!

# Alignment Error Rates: How good is the prediction?

- Given: predicted alignments A, sure links S, and possible links P
- Precision: $\frac{|A \cap P|}{|A|}$ Recall: $\frac{|A \cap S|}{|S|}$
- AER(A|S,P) $= 1 - \frac{|A \cap P| + |A \cap S|}{|A| + |S|}$

**Reference alignments**, with **P**ossible links and **S**ure links



# 1 Remaining Task

**Inference**

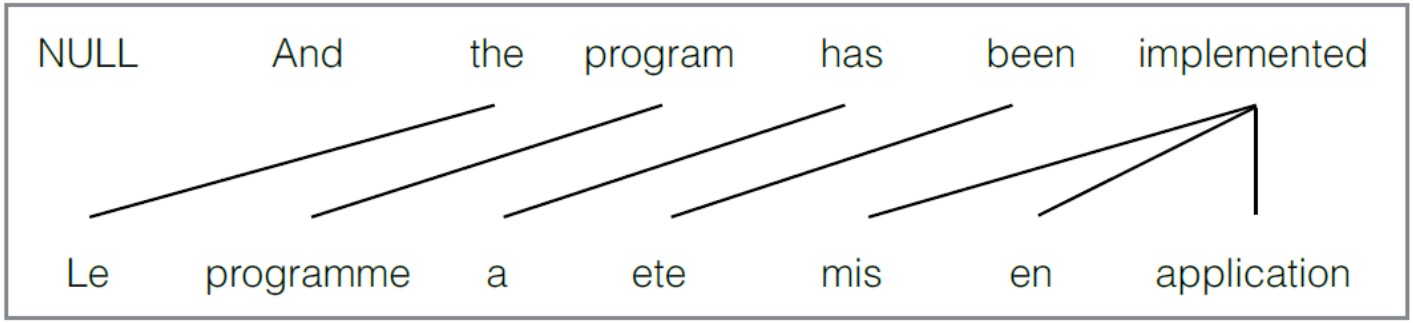
- Given a sentence pair (e,f), what is the most probable alignment a?

**Parameter Estimation**

- How do we learn the parameters $t(e|f)$ and $q(j|i,l,m)$ from data?

# Parameter Estimation (warm-up)

- Inputs
  - Model definition ( t and q )
  - A corpus of sentence pairs, with word alignment




- How do we build tables for t and q?
  - Use counts, just like for n-gram models!

# Parameter Estimation (for real)

- Problem
  - Parallel corpus gives us **(e,f)** pairs only, **a is hidden**
- We know how to
  - estimate **t** and **q**, given **(e,a,f)**
  - compute **p(e,a|f)**, given t and q
- Solution: Expectation-Maximization algorithm (EM)
  - E-step: given hidden variable, estimate parameters
  - M-step: given parameters, update hidden variable

# Parameter Estimation: hard EM

Algorithm 1 (hard EM)

initialize parameters $t$ and $q$ to something
repeat until convergence
    for every sentence
        for every target position $j$
            for every source position $i$
                if aligned($i$, $j$)
                    count($f_j$ | $e_i$) += 1
                    count($e_i$) += 1
                    count($j$, $i$, $l$, $m$) += 1
                    count($i$, $l$, $m$) += 1
    t(f | e) = count(f, e) / count(e)
    q(j | i, l, m) = count(j, i, l, m) / count(i, l, m)

# Parameter Estimation: soft EM

Algorithm 1 (soft EM)

initialize parameters $t$ and $q$ to something
repeat until convergence
    for every sentence
        for every target position $j$
            for every source position $i$
                $\text{count}(f_j, e_i) \mathrel{+}= P(a_i = j \mid e_i, f_j)$
                $\text{count}(e_i) \mathrel{+}= P(a_i = j \mid e_i, f_j)$
                $\text{count}(j, i, l, m) \mathrel{+}= P(a_i = j \mid e_i, f_j)$
                $\text{count}(i, l, m) \mathrel{+}= P(a_i = j \mid e_i, f_j)$
    $t(f \mid e) = \text{count}(f, e) / \text{count}(e)$
    $q(j \mid i, l, m) = \text{count}(j, i, l, m) / \text{count}(i, l, m)$

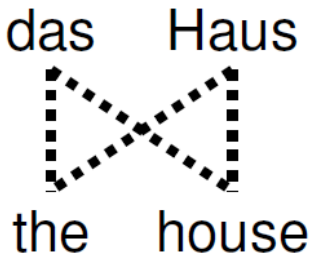
Use “Soft” values instead of binary counts

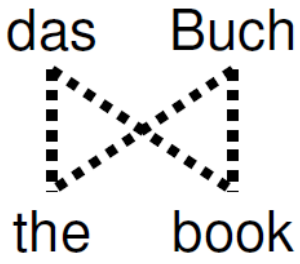
# Parameter Estimation: soft EM

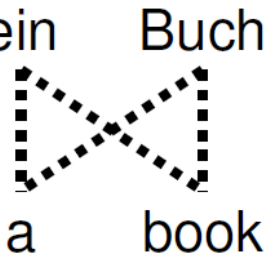
- Soft EM considers all possible alignment links
- Each alignment link now has a weight

$$P(a_i = j \mid e_i, f_j) = \frac{q(j \mid i, l, m) \cdot t(f_i \mid e_j)}{\sum_{j'=1}^{l} q(j' \mid i, l, m) \cdot t(f_i \mid e_{j'})}$$

# Example: learning t table using EM for IBM1




| $e$ | $f$ | 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 |

# We have now fully specified our probabilistic alignment model!

Probability lets us

1) Formulate a model of pairs of sentences

   **=> IBM models 1 & 2**

2) Learn an instance of the model from data

   **=> using EM**

3) Use it to infer alignments of new inputs

   **=> based on independent translation decisions**

# Summary: Noisy Channel Model for Machine Translation

- The **noisy channel model** decomposes machine translation into two independent subproblems
  - Word alignment
  - Language modeling

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \overbrace{P(F|E)}^{\text{translation model}} \overbrace{P(E)}^{\text{language model}}$$

# Summary: Word Alignment with IBM Models 1, 2

- Probabilistic models with **strong independence assumptions**
  - Results in linguistically naïve models
    - asymmetric, 1-to-many alignments
  - But allows efficient parameter estimation and inference
- Alignments are hidden variables
  - unlike words which are observed
  - require **unsupervised learning** (EM algorithm)

# Today

- Walk through an example of EM
- Phrase-based Models
  - A slightly more recent translation model
- Decoding

# EM FOR IBM1

# IBM Model 1: generative story

- Input
  - an English sentence of length l
  - a length m

- For each French position $i$ in 1..m
  - Pick an English source index j $q(j \mid i, l, m) = \frac{1}{l+1}$
  - Choose a translation $t(f_i \mid e_{a_i})$

$$p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l, m) = \prod_{i=1}^{m} q(a_i | i, l, m) t(f_i | e_{a_i})$$

# EM for IBM Model 1

- Expectation (E)-step:
  - Compute expected counts for parameters (t) based on summing over hidden variable
- Maximization (M)-step:
  - Compute the maximum likelihood estimate of t from the expected counts

# EM example: initialization

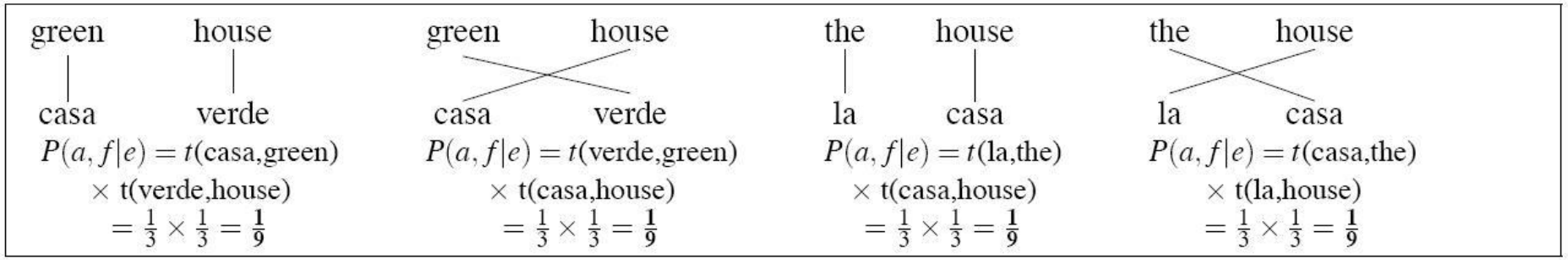
| | |
|---|---|
| `green house` | `the house` |
| `casa verde` | `la casa` |

| | | |
|---|---|---|
| $t(\text{casa}\mid\text{green}) = \frac{1}{3}$ | $t(\text{verde}\mid\text{green}) = \frac{1}{3}$ | $t(\text{la}\mid\text{green}) = \frac{1}{3}$ |
| $t(\text{casa}\mid\text{house}) = \frac{1}{3}$ | $t(\text{verde}\mid\text{house}) = \frac{1}{3}$ | $t(\text{la}\mid\text{house}) = \frac{1}{3}$ |
| $t(\text{casa}\mid\text{the}) = \frac{1}{3}$ | $t(\text{verde}\mid\text{the}) = \frac{1}{3}$ | $t(\text{la}\mid\text{the}) = \frac{1}{3}$ |

For the rest of this talk, French = Spanish

# EM example: E-step

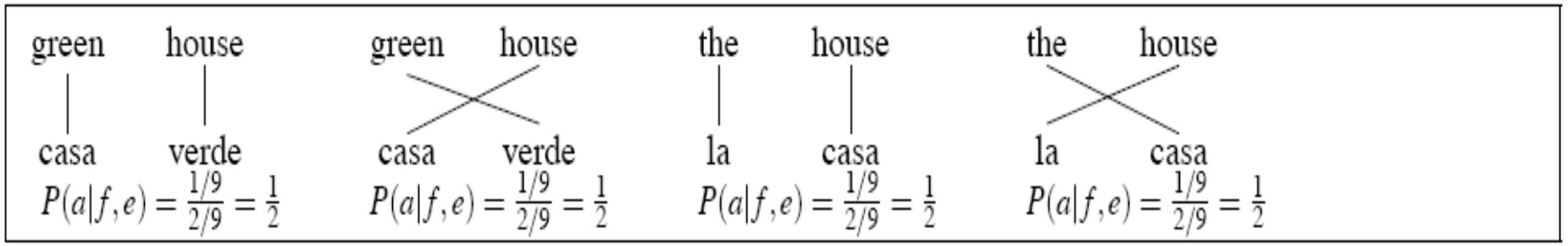
## (a) compute probability of each alignment p(a|f,e)




Note: we're making many simplification assumptions in this example!!

- No NULL word
- We only consider alignments were each French and English word is aligned to something
- We ignore q

# EM example: E-step
## (b) normalize to get p(a|f,e)



# EM example: E-step

## (c) compute expected counts (weighting each count by p(a|e,f)

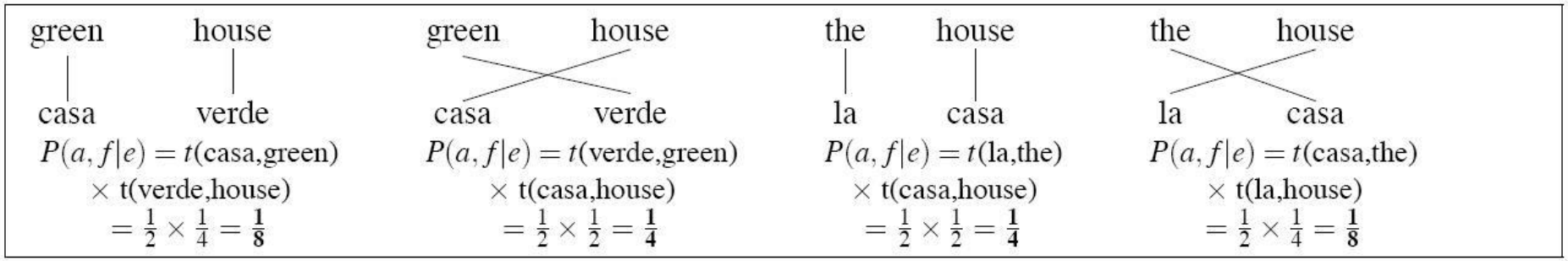
| | | |
|---|---|---|
| $\text{tcount}(\text{casa}\mid\text{green}) = \frac{1}{2}$ | $\text{tcount}(\text{verde}\mid\text{green}) = \frac{1}{2}$ | $\text{tcount}(\text{la}\mid\text{green}) = 0$ $\quad \text{total}(\text{green}) = 1$ |
| $\text{tcount}(\text{casa}\mid\text{house}) = \frac{1}{2} + \frac{1}{2}$ | $\text{tcount}(\text{verde}\mid\text{house}) = \frac{1}{2}$ | $\text{tcount}(\text{la}\mid\text{house}) = \frac{1}{2}$ $\quad \text{total}(\text{house}) = 2$ |
| $\text{tcount}(\text{casa}\mid\text{the}) = \frac{1}{2}$ | $\text{tcount}(\text{verde}\mid\text{the}) = 0$ | $\text{tcount}(\text{la}\mid\text{the}) = \frac{1}{2}$ $\quad \text{total}(\text{the}) = 1$ |

# EM example: M-step

## Compute probability estimate by normalizing expected counts

| | | |
|---|---|---|
| $t(\text{casa}\mid\text{green}) = \frac{1/2}{1} = \frac{1}{2}$ | $t(\text{verde}\mid\text{green}) = \frac{1/2}{1} = \frac{1}{2}$ | $t(\text{la}\mid\text{green}) = \frac{0}{1} = 0$ |
| $t(\text{casa}\mid\text{house}) = \frac{1}{2} = \frac{1}{2}$ | $t(\text{verde}\mid\text{house}) = \frac{1/2}{2} = \frac{1}{4}$ | $t(\text{la}\mid\text{house}) = \frac{1/2}{2} = \frac{1}{4}$ |
| $t(\text{casa}\mid\text{the}) = \frac{1/2}{1} = \frac{1}{2}$ | $t(\text{verde}\mid\text{the}) = \frac{0}{1} = 0$ | $t(\text{la}\mid\text{the}) = \frac{1/2}{1} = \frac{1}{2}$ |

# EM example: next iteration



# EM for IBM 1 in practice

- The previous example aims to illustrate the intuition of EM algorithm

- But it is a little naïve
  - we had to enumerate all possible alignments
  - very inefficient!!
  - In practice, we don't need to sum overall all possible alignments explicitly for IBM1

  http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/ibm12.pdf

# PHRASE-BASED MODELS

# Phrase-based models

- Most common way to model P(F|E) nowadays (instead of IBM models)

$$P(F|E) = \prod_{i=1}^{I} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1})$$

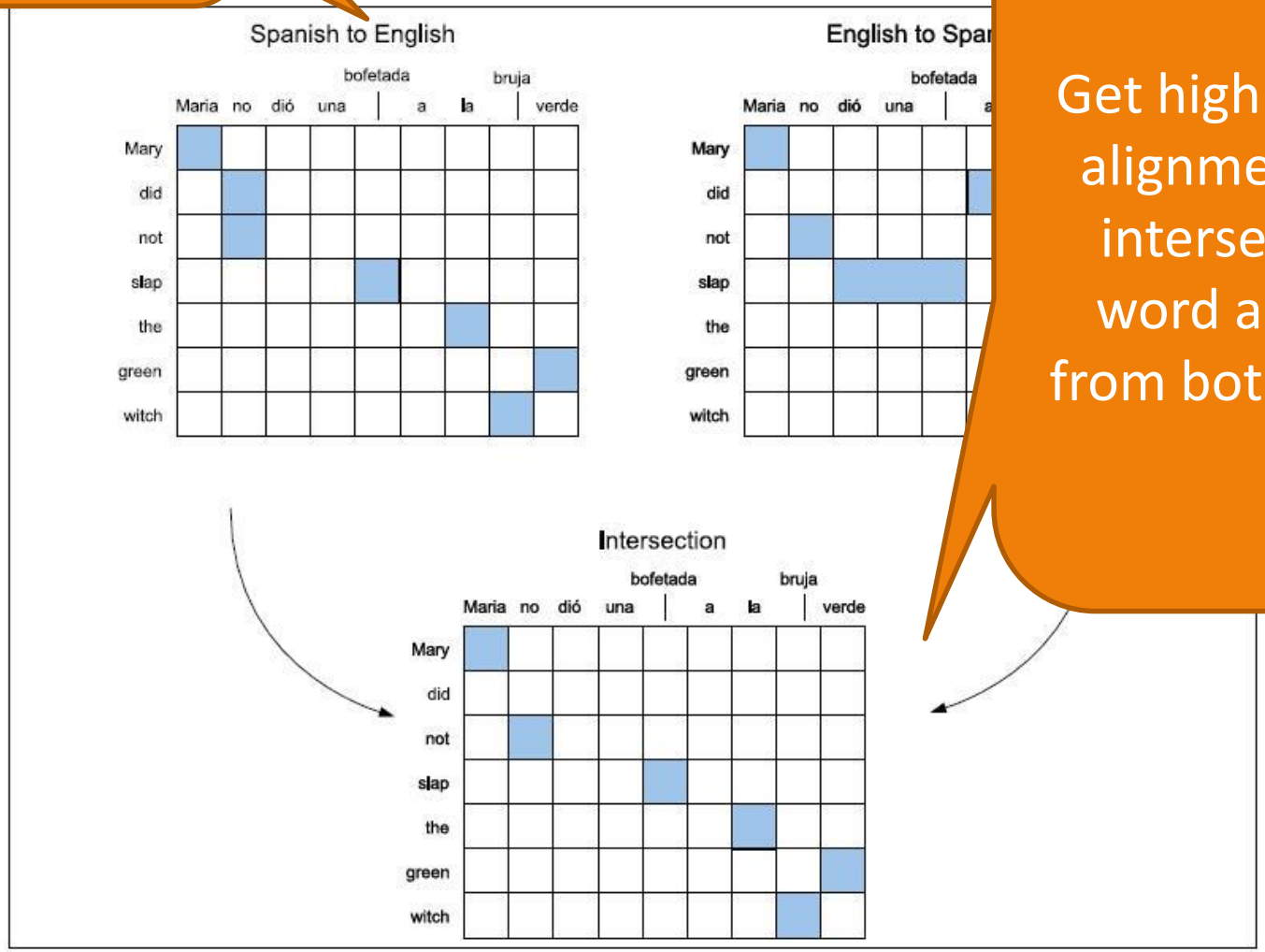
Start position of f_i

End position of f_(i-1)

Probability of two consecutive English phrases being separated by a particular span in French

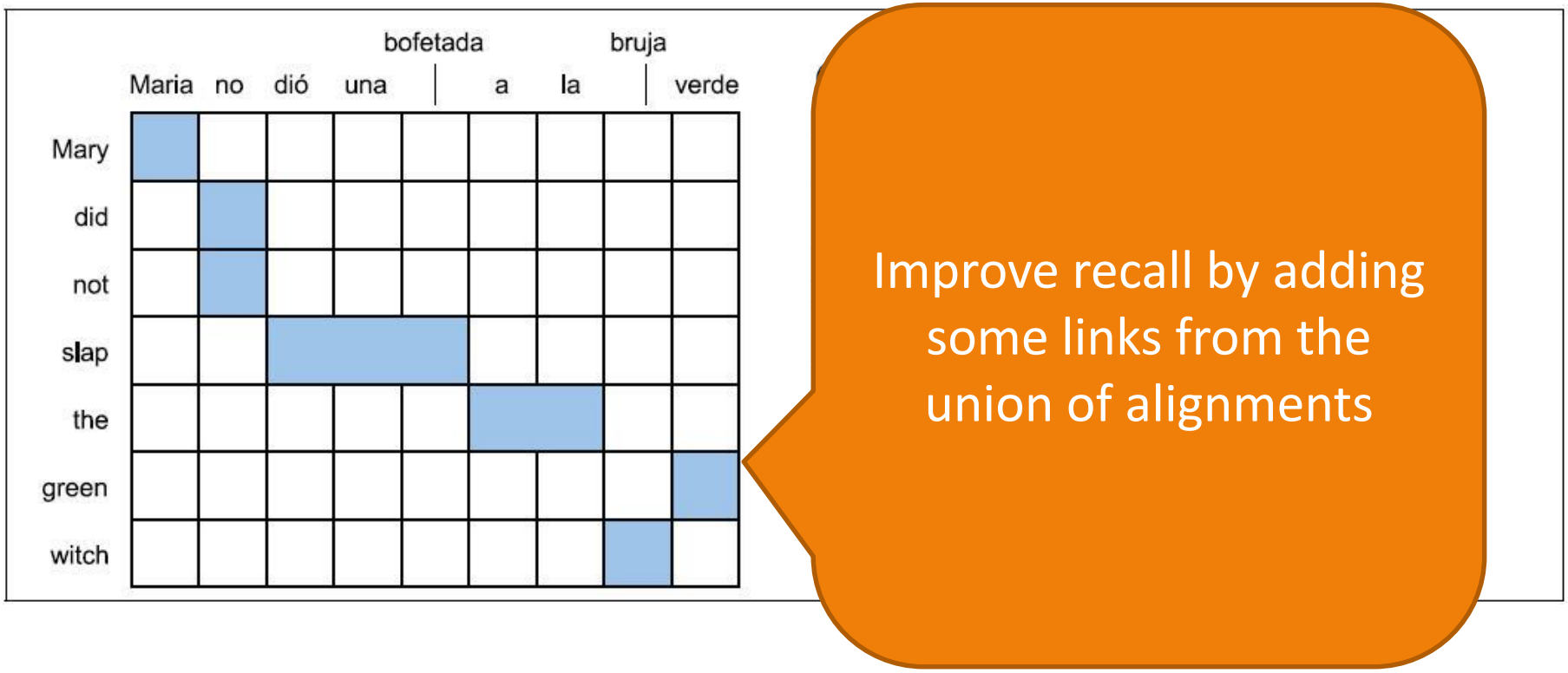
# Phrase alignments are derived from word alignments

This means that the IBM model represents P(Spanish|English)

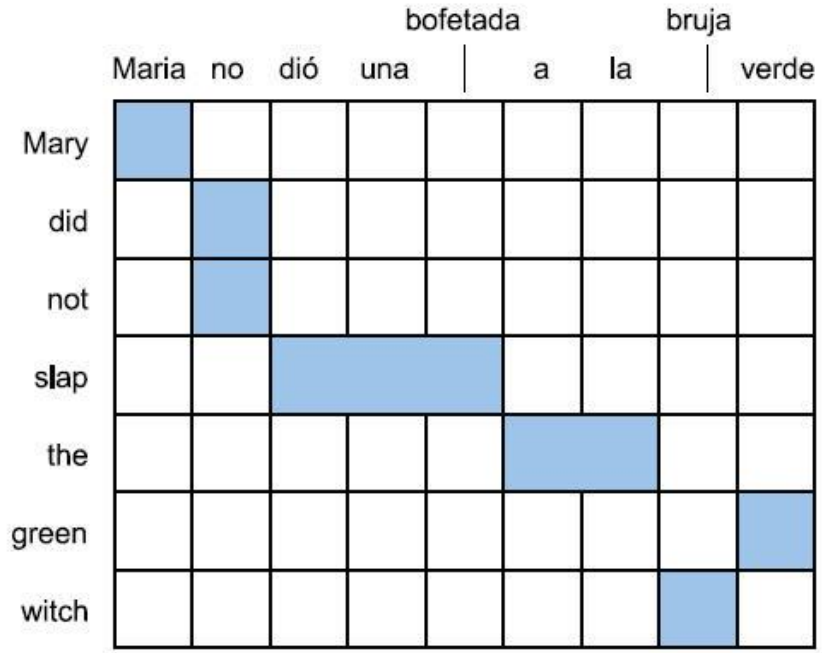



Get high confidence alignment links by intersecting IBM word alignments from both directions

# Phrase alignments are derived from word alignments



# Phrase alignments are derived from word alignments




(Maria, Mary), (no, did not),
(slap, dió una bofetada), (verde, green),
(a la, the), (bruja, witch),
(Maria no, Mary did not),
(no dió una bofetada, did not slap),
(dió una bofetada a la, slap the),
(bruja verde, green witch),
(a la bruja verde, the green witch),. . .

Extract phrases that are **consistent** with word alignment

# Phrase Translation Probabilities

- Given such phrases we can get the required statistics for the model from

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

# Phrase-based Machine Translation

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \overbrace{P(F|E)}^{\text{translation model}} \overbrace{P(E)}^{\text{language model}}$$

$$\prod_{i \in S} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1})$$

# DECODING

# Decoding for phrase-based MT

- Basic idea
  - search the space of possible English translations in an efficient manner.
  - According to our model

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \overbrace{P(F|E)}^{\text{translation model}} \overbrace{P(E)}^{\text{language model}}$$

$$\text{cost}(E,F) = \prod_{i \in S} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1}) P(E)$$

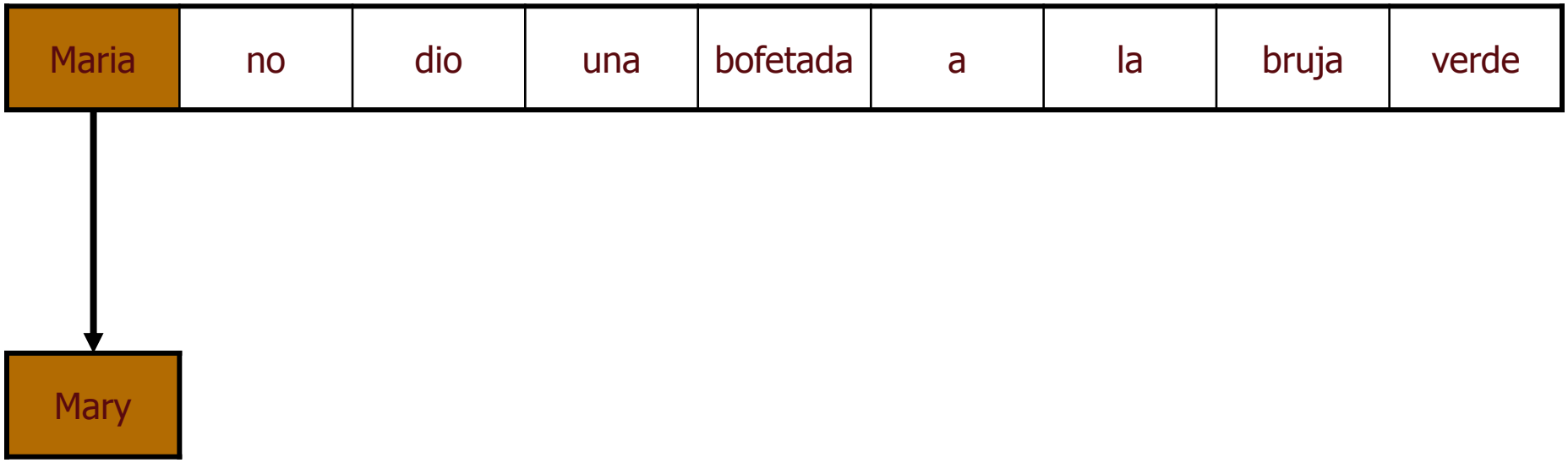
# Decoding as Search

- Starting point: null state.  No French content covered, no English included.
- We'll drive the search by
  - Choosing French word/phrases to "cover",
  - Choosing a way to cover them
- Subsequent choices are pasted left-to-right to previous choices.
- Stop: when all input words are covered.

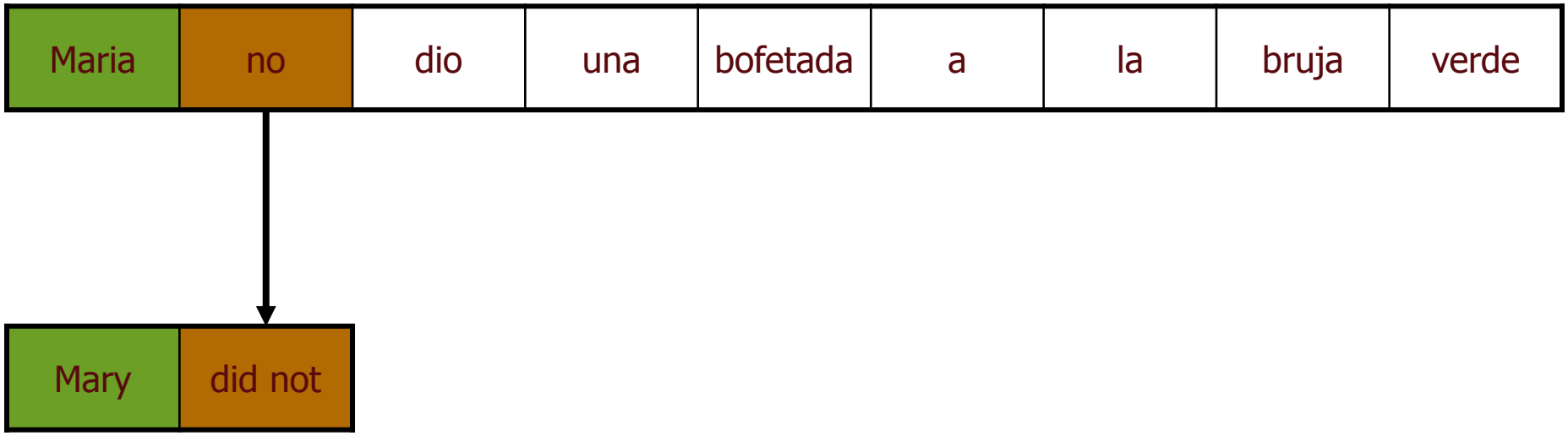
# Decoding

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

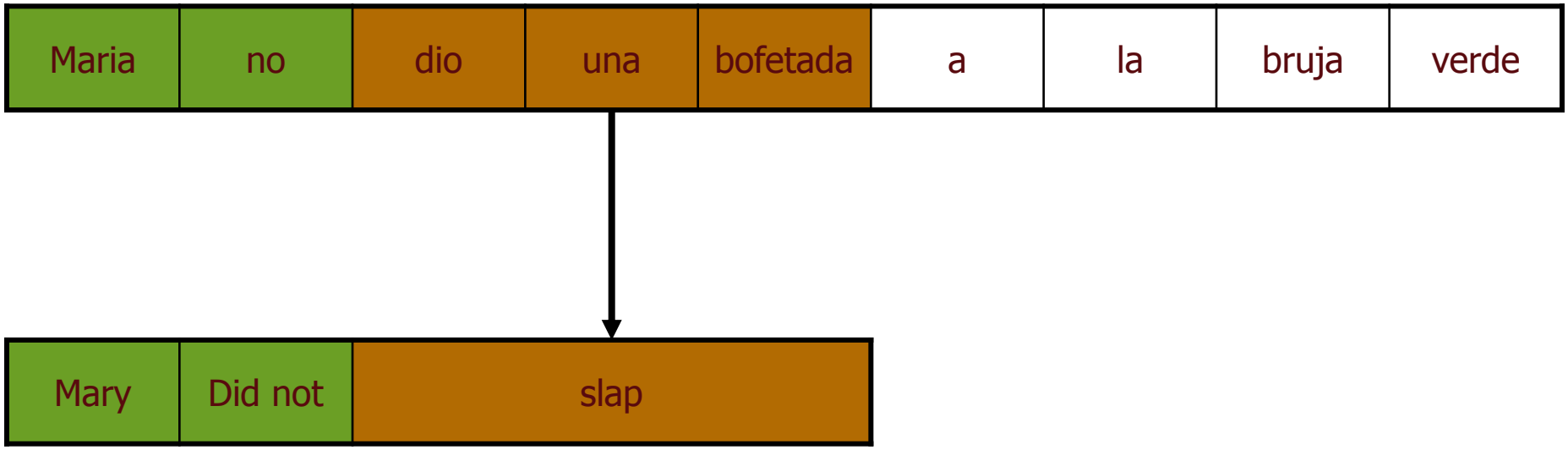
# Decoding



# Decoding



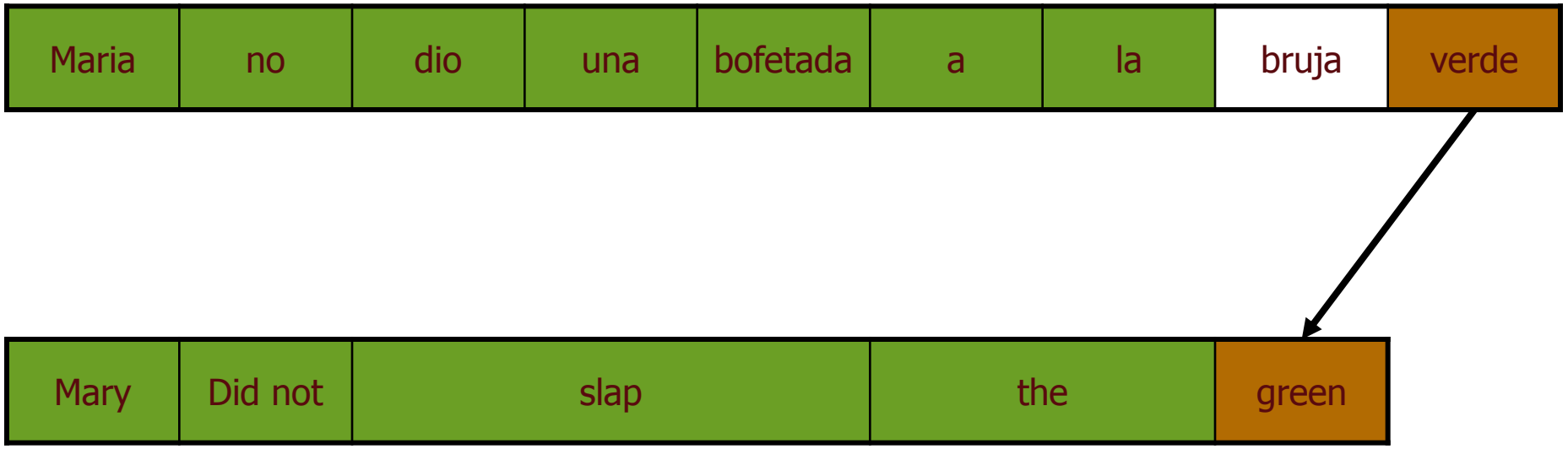
# Decoding



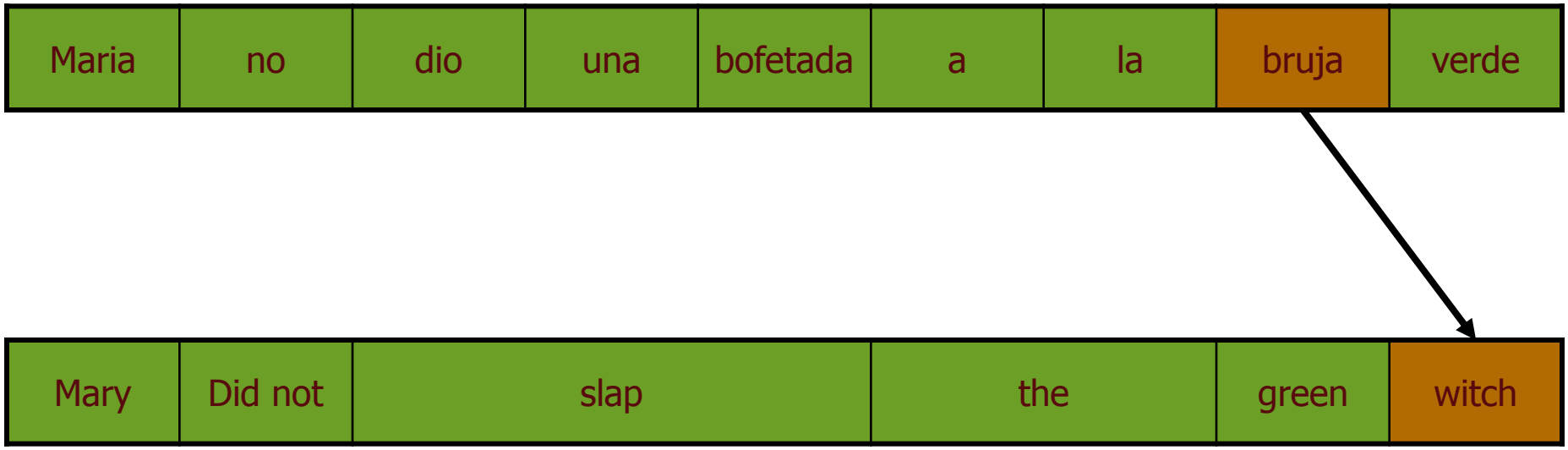
# Decoding


Maria
no
dio
una
bofetada
a
la
bruja
verde
Mary
Did not
slap
the

# Decoding


Maria
no
dio
una
bofetada
a
la
bruja
verde
Mary
Did not
slap
the
green

# Decoding


Maria
no
dio
una
bofetada
a
la
bruja
verde
Mary
Did not
slap
the
green
witch

# Decoding

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

| Mary | did not | slap | the | green | witch |
|---|---|---|---|---|---|

# Decoding

- In practice: we need to incrementally pursue a large number of paths.

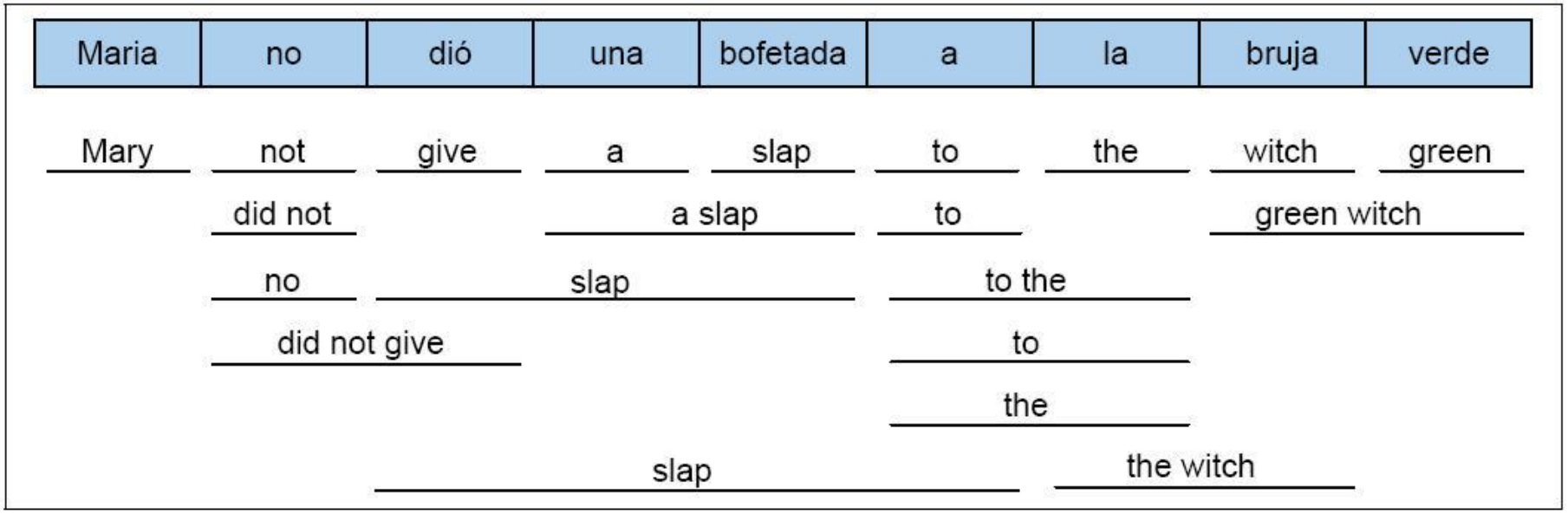
- Solution: heuristic search algorithm called "multi-stack beam search"

# Stack decoding: a simplified view

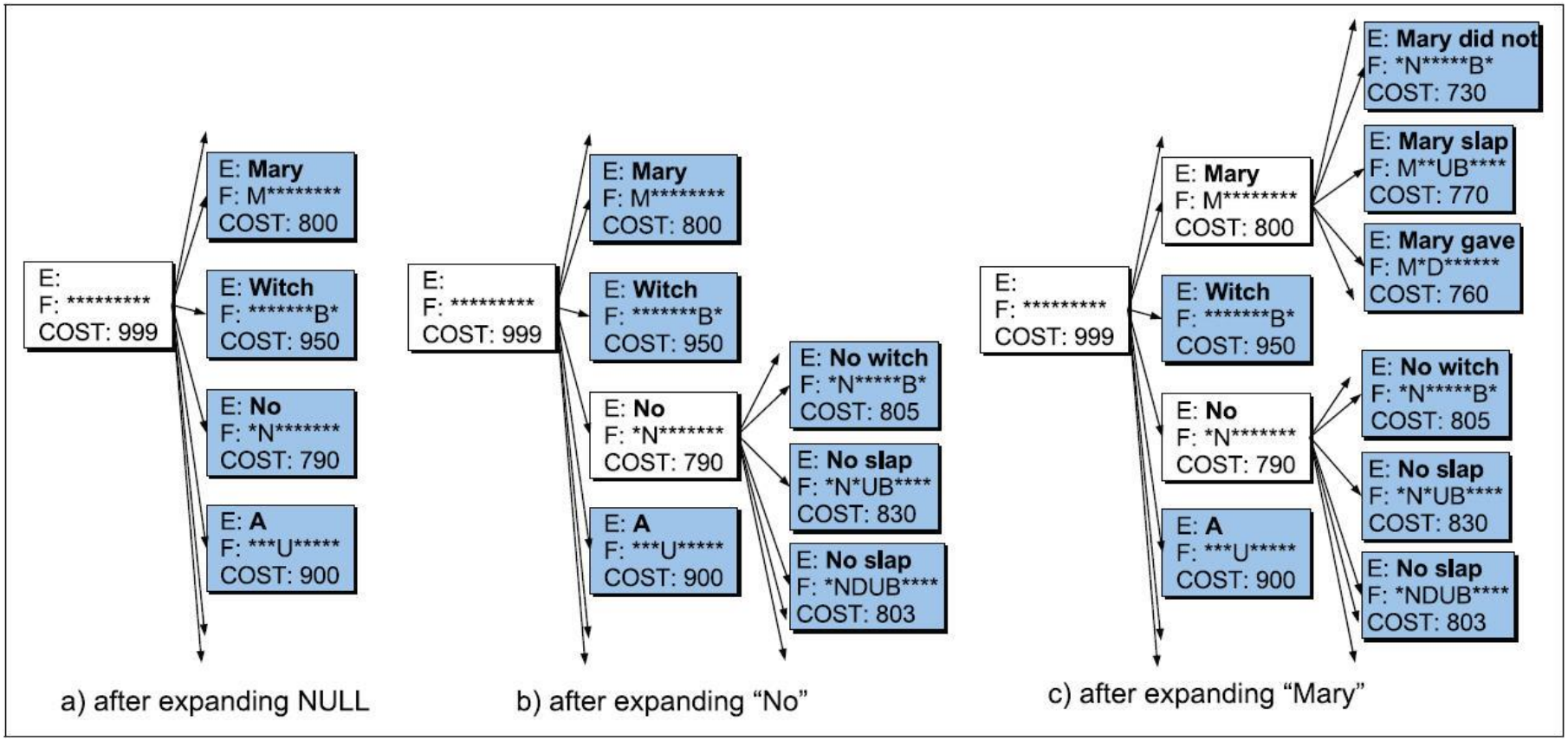
**function** STACK DECODING(source sentence) **returns** target sentence

initialize stack with a null hypothesis
**loop do**
  pop best hypothesis $h$ off of stack
  **if** $h$ is a complete sentence, **return** $h$
  **for each** possible expansion $h'$ of $h$
    assign a score to $h'$
    push $h'$ onto stack

# Space of possible English translations given phrase-based model



# Three stages of stack decoding



# "multi-stack beam search"

```
function BEAM SEARCH STACK DECODER(source sentence) returns target sentence

initialize hypothesisStack[0..nf]
push initial null hypothesis on hypothesisStack[0]
for i ← 0 to nf-1
   for each hyp in hypothesisStack[i]
      for each new_hyp that can be derived from hyp
         nf_new_hyp ← number of foreign words covered by new_hyp
         add new_hyp to hypothesisStack[nf_new_hyp]
         prune hypothesisStack[nf_new_hyp]
find best hypothesis best_hyp in hypothesisStack[nf]
return best path that leads to best_hyp via backtrace
```

# "multi-stack beam search"

```
function BEAM SEARCH STACK DE

initialize hypothesisStack[0..nf]
push initial null hypothesis on hyp
for i ← 0 to nf-1
   for each hyp in hypothesisStack[i]
      for each new_hyp that can be derived from hyp
         nf_new_hyp ← number of foreign words covered by new_hyp
         add new_hyp to hypothesisStack[nf_new_hyp]
         prune hypothesisStack[nf_new_hyp]
find best hypothesis best_hyp in hypothesisStack[nf]
return best path that leads to best_hyp via backtrace
```

One stack per number of French words covered: so that we make apples-to-apples comparisons **when pruning**

Beam-search pruning **for each stack**: prune high cost states (those "outside the beam")

# Cost = current cost + future cost

- Future cost = cost of translating remaining words in the French sentence
- Exact future cost = minimum probability of all remaining translations
  - Too expensive to compute!
- Approximation
  - Find sequence of English phrases that has the minimum product of language model and translation model costs

# Complexity Analysis

- Time complexity of decoding as described so far

  **O(max stack size x sentence length^2)**

  - O( max stack size x number of ways to expand hyps. x sentence length)
  - Number of hyp expansions is linear in sentence length, because we only consider the top k translation candidates in the phrase-table

- In practice: **O(max stack size x sentence length)**
  - because we limit reordering distance, so that only a constant number of hypothesis expansions are considered

# RECAP

# Phrase-based Machine Translation: the full picture

$$\hat{E} = \underset{E \in \text{English}}{\text{argmax}} \overbrace{P(F|E)}^{\text{translation model}} \overbrace{P(E)}^{\text{language model}}$$

$$\prod_{i \in S} \phi(\bar{f}_i, \bar{e}_i) d(a_i - b_{i-1})$$

# Phrase-based MT: discussion

- What is the advantage of splitting the problem in 2?
- What are the strengths and weaknesses of this approach?