# Lecture 10: Machine Translation I

Alan Ritter

(many slides from Greg Durrett)

# This Lecture

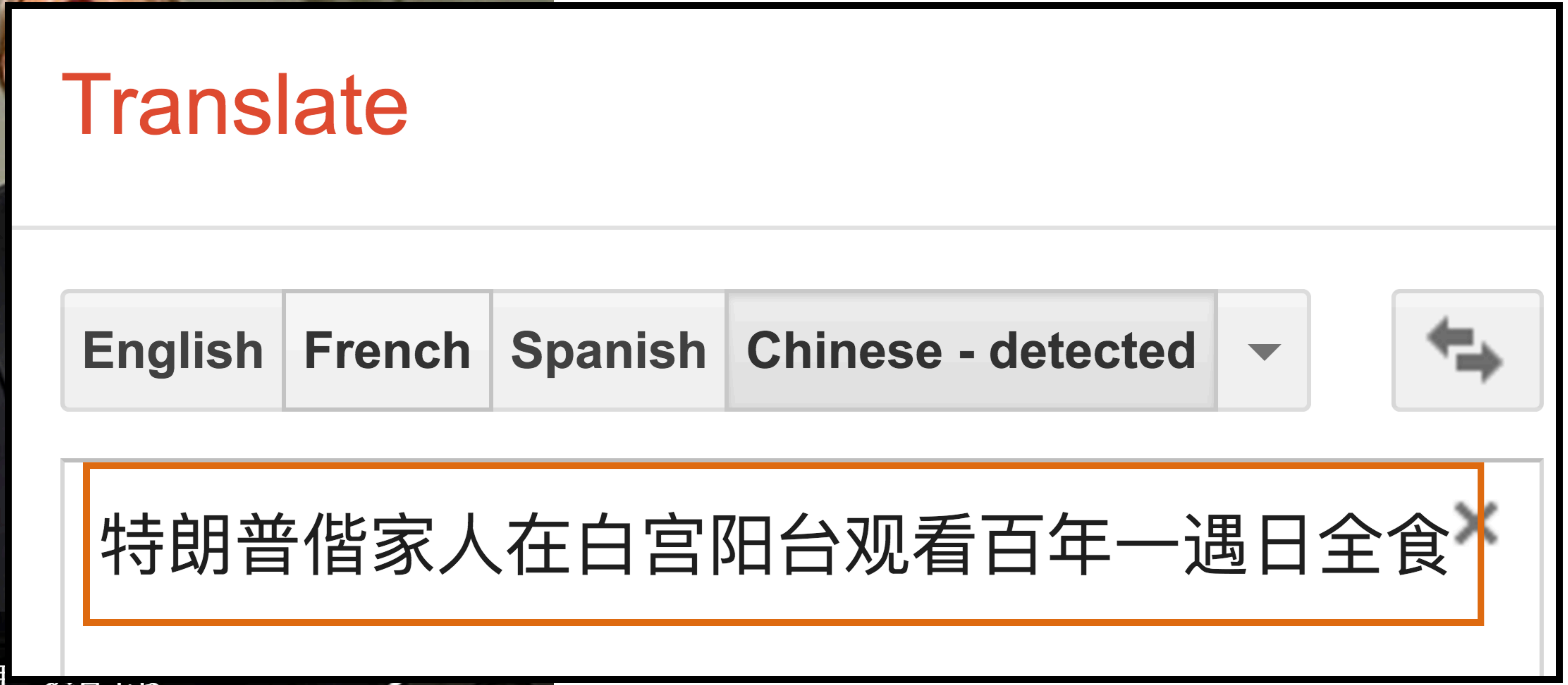
- MT and evaluation
- Word alignment
- Language models
- Phrase-based decoders
- Syntax-based decoders (probably next time)

# MT Basics

# MT Basics

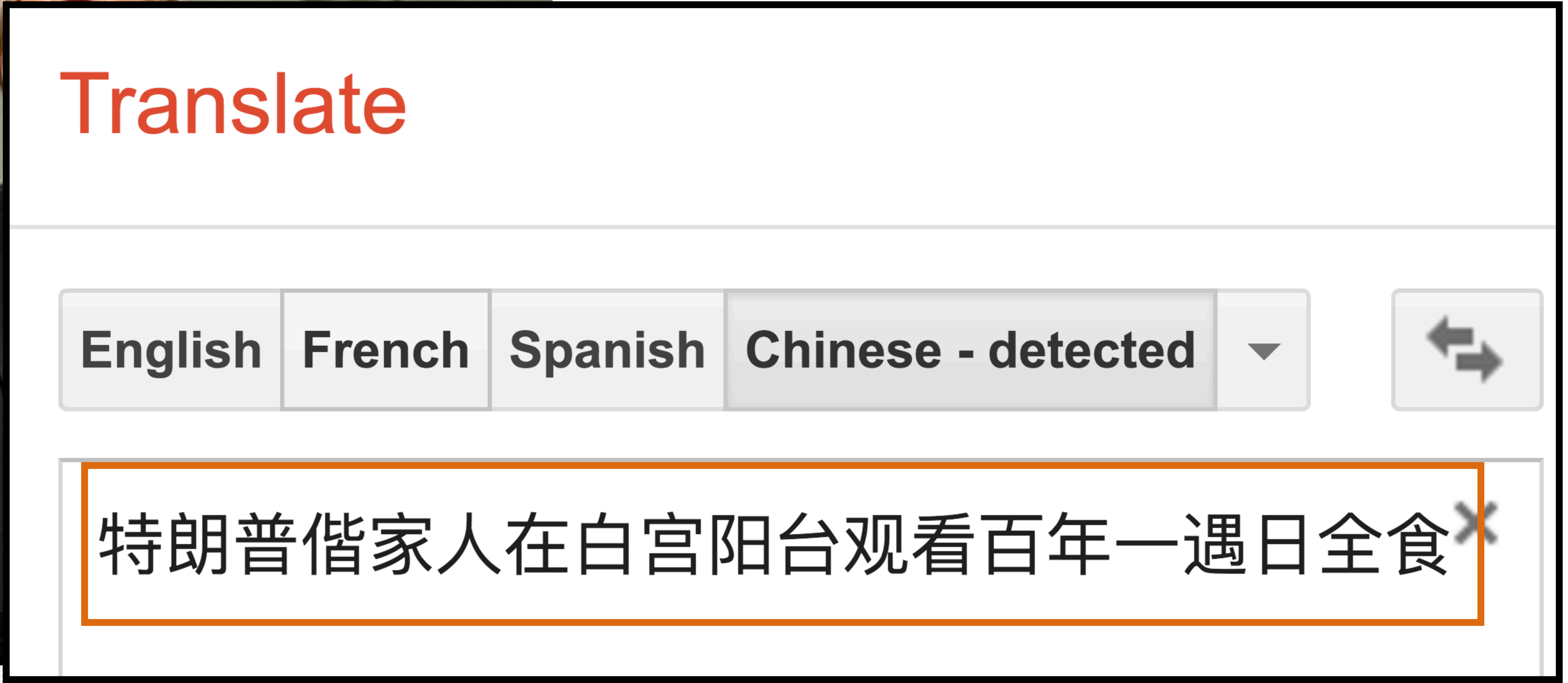

People’s Daily, August 30, 2017

# MT Basics


People's Daily, August 30, 2017

# MT Basics


People’s Daily, August 30, 2017

Trump Pope family watch a hundred years a year in the White House balcony

# MT Basics


People’s Daily, August 30, 2017

Trump Pope family watch a hundred years a year in the White House balcony

# MT Ideally

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)`

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J'ai un ami

# MT Ideally

- I have a friend => `∃x friend(x,self)` => J’ai un ami
  J’ai une amie

# MT Ideally

- I have a friend => `∃x friend(x,self)` => J’ai un ami
  J’ai une amie
  - May need information you didn’t think about in your representation

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J'ai un ami

  J'ai une amie

  - May need information you didn't think about in your representation
  - Hard for semantic representations to cover everything

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J’ai un ami

  J’ai une amie

  - May need information you didn’t think about in your representation
  - Hard for semantic representations to cover everything

- Everyone has a friend =>

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J'ai un ami

  J'ai une amie

  - May need information you didn't think about in your representation
  - Hard for semantic representations to cover everything

- Everyone has a friend => $\exists x \forall y$ `friend(x,y)`

  $\forall x \exists y$ `friend(x,y)`

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J'ai un ami

  J'ai une amie

  - May need information you didn't think about in your representation
  - Hard for semantic representations to cover everything
- Everyone has a friend => $\exists x \forall y$ `friend(x,y)` => Tous a un ami

  $\forall x \exists y$ `friend(x,y)`

# MT Ideally

- I have a friend => $\exists x$ `friend(x,self)` => J'ai un ami

  J'ai une amie

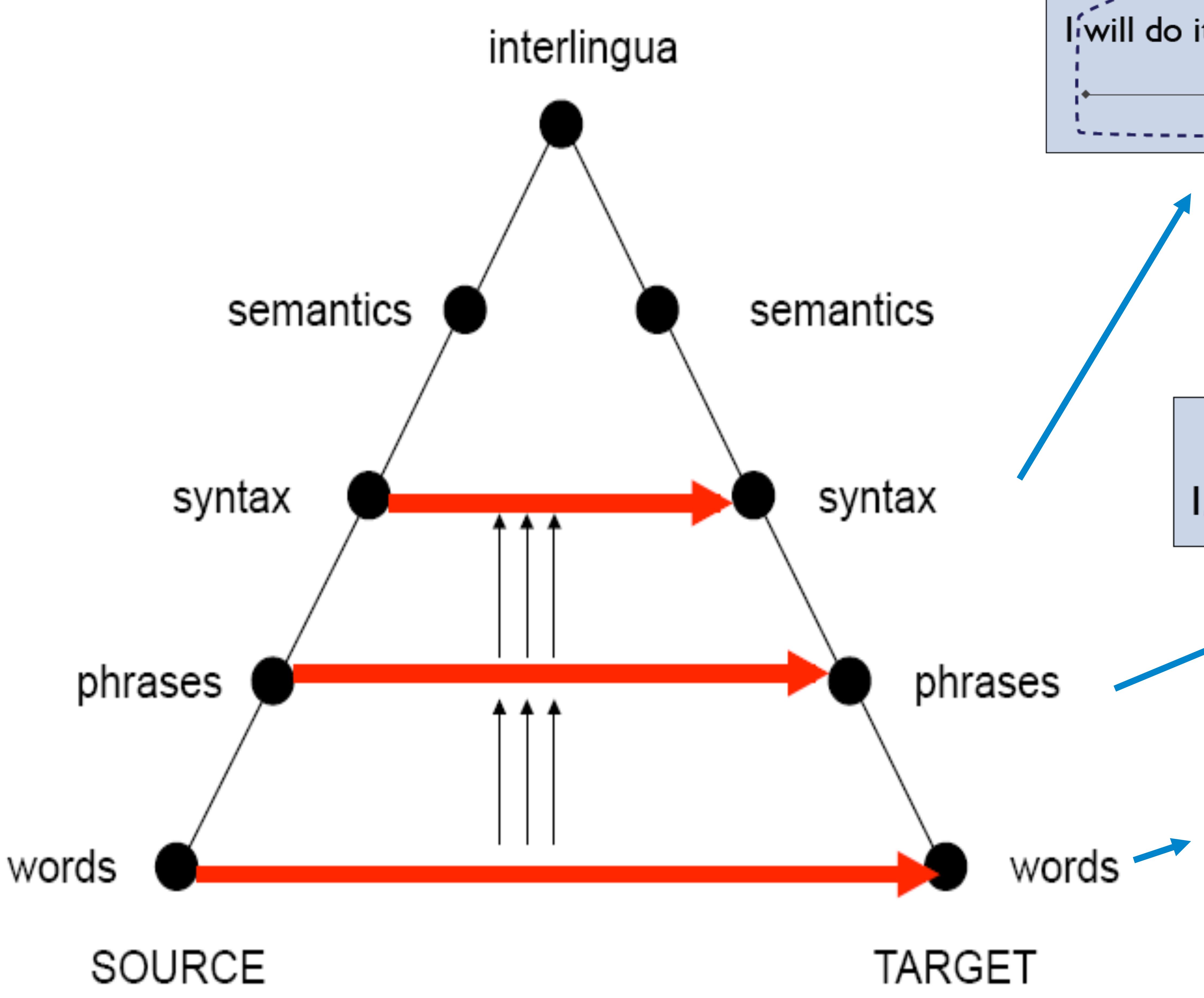
  - May need information you didn't think about in your representation
  - Hard for semantic representations to cover everything

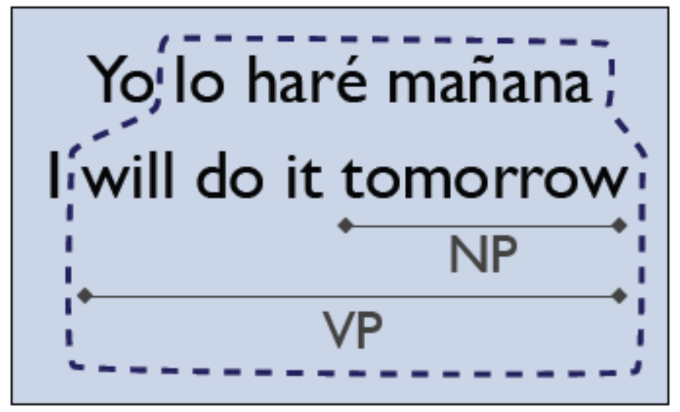
- Everyone has a friend => $\exists x \forall y$ `friend(x,y)` => Tous a un ami

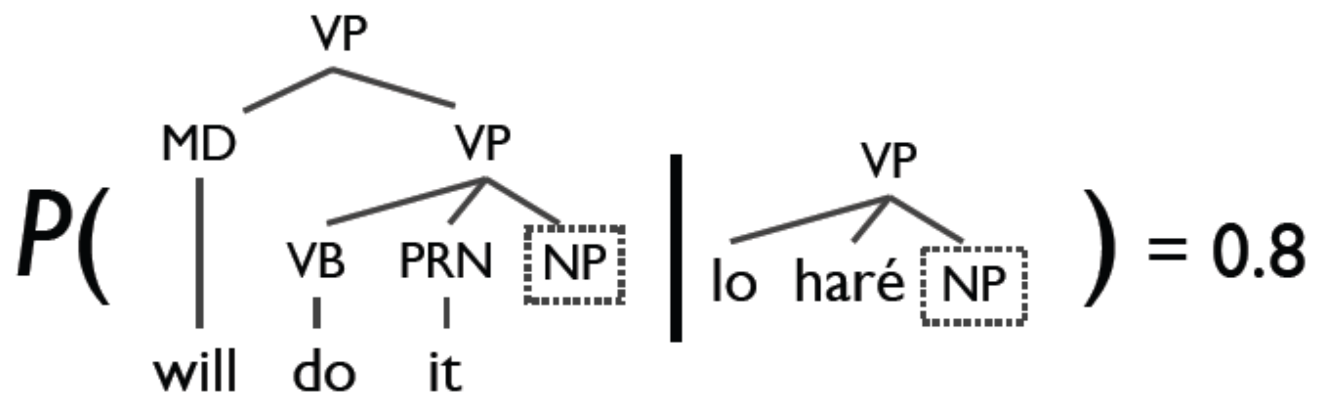
  $\forall x \exists y$ `friend(x,y)`

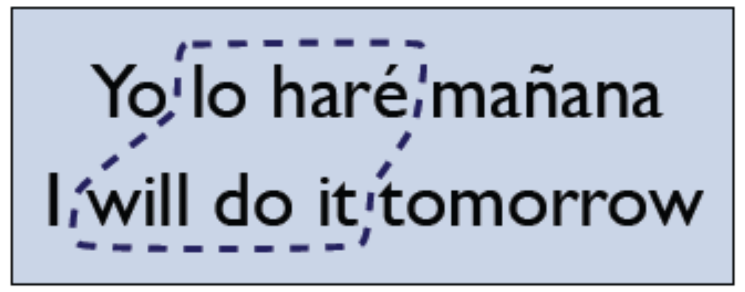
  - Can often get away without doing all disambiguation — same ambiguities may exist in both languages

# Levels of Transfer: Vauquois Triangle

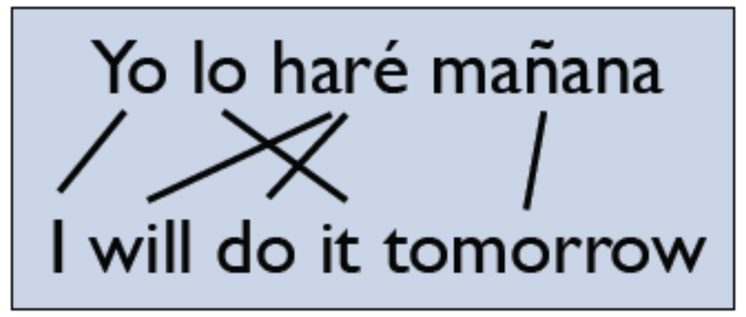



| *English* (E) | *P*( E \| lo haré ) |
|---|---|
| will do it | 0.8 |
| will do so | 0.2 |




| *English* (E) | *P*( E \| mañana ) |
|---|---|
| tomorrow | 0.7 |
| morning | 0.3 |

Slide credit: Dan Klein

# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use

# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate

# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext

# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language

# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language
  - Decoder takes phrases and a language model and searches over possible translations

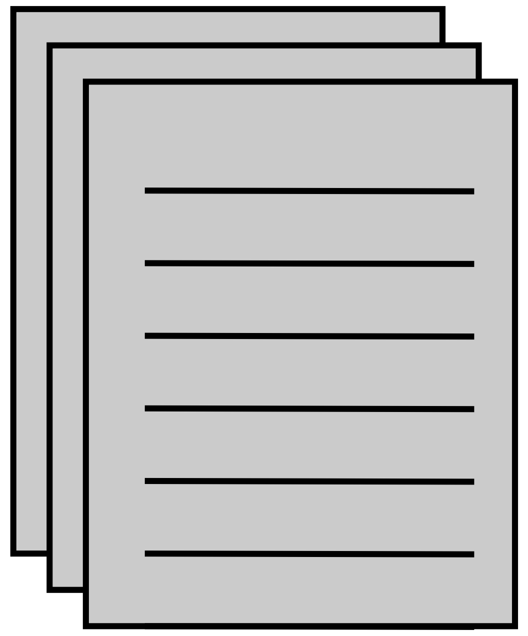
# Phrase-Based MT

- Key idea: translation works better the bigger chunks you use
- Remember phrases from training data, translate piece-by-piece and stitch those pieces together to translate
  - How to identify phrases? Word alignment over source-target bitext
  - How to stitch together? Language model over target language
  - Decoder takes phrases and a language model and searches over possible translations
- NOT like standard discriminative models (take a bunch of translation pairs, learn a ton of parameters in an end-to-end way)

# Phrase-Based MT

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Unlabeled English data

Language model P(e)

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

"Translate faithfully but make fluent English"

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

| | | 1-gram | 2-gram | 3-gram |
|---|---|---|---|---|
| hypothesis 1 | I am exhausted | 3/3 | 1/2 | 0/1 |
| hypothesis 2 | Tired is I | 1/3 | 0/2 | 0/1 |
| hypothesis 3 | I I I | 1/3 | 0/2 | 0/1 |
| reference 1 | I am tired | | | |
| reference 2 | I am ready to sleep now and so exhausted | | | |

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{B}\textsc{leu} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \cdot$$

| | | 1-gram | 2-gram | 3-gram |
|---|---|---|---|---|
| hypothesis 1 | I am exhausted | 3/3 | 1/2 | 0/1 |
| hypothesis 2 | Tired is I | 1/3 | 0/2 | 0/1 |
| hypothesis 3 | I I I | 1/3 | 0/2 | 0/1 |
| reference 1 | I am tired | | | |
| reference 2 | I am ready to sleep now and so exhausted | | | |

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

- r = length of reference
  c = length of prediction

# Evaluating MT

- Fluency: does it sound good in the target language?
- Fidelity/adequacy: does it capture the meaning of the original?
- BLEU score: geometric mean of 1-, 2-, 3-, and 4-gram precision vs. a reference, multiplied by brevity penalty

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

- Typically $n = 4$, $w_i = 1/4$

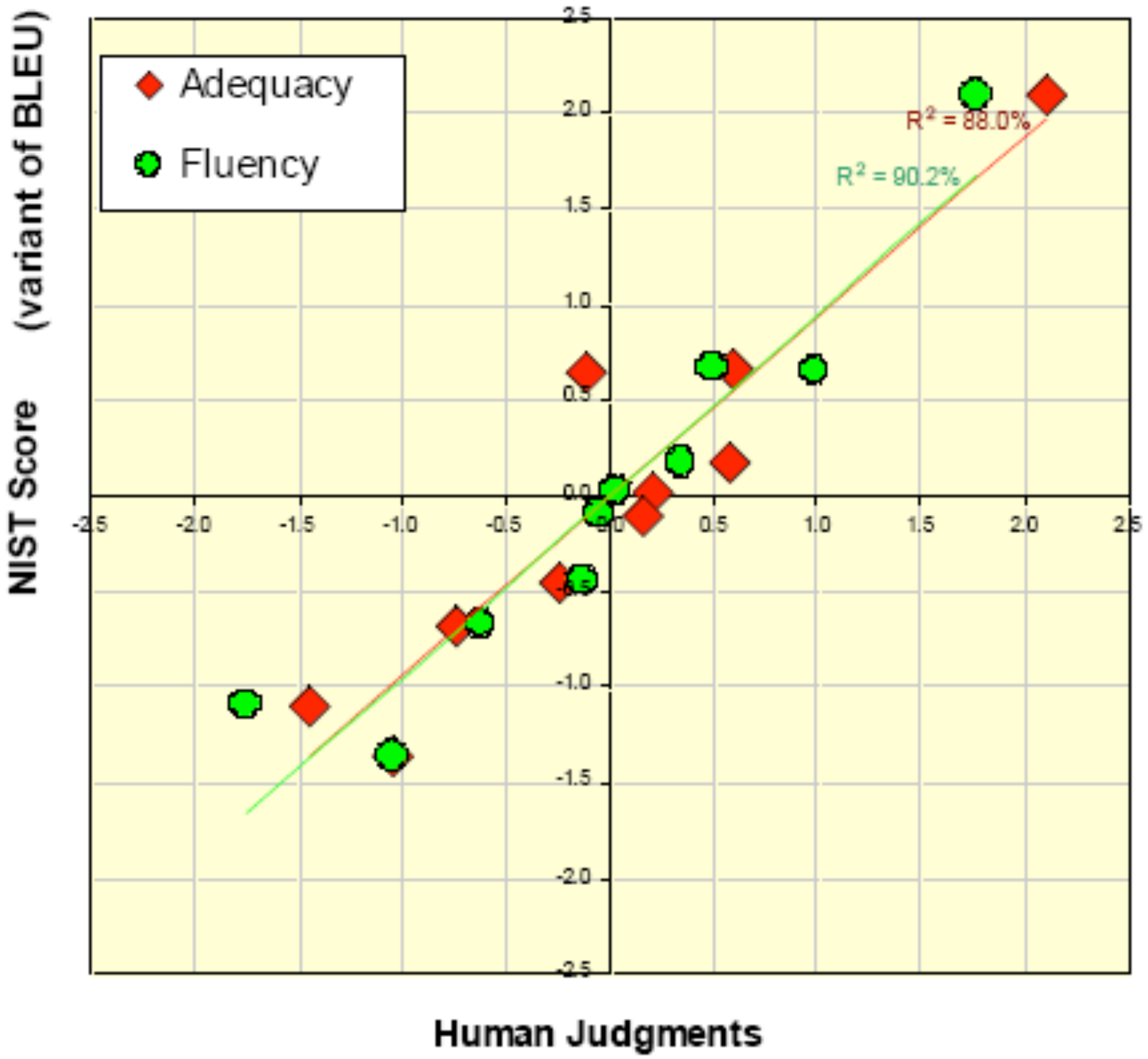
$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

- r = length of reference
  c = length of prediction

- Does this capture fluency and adequacy?

# BLEU Score

- Better methods with human-in-the-loop
- HTER: human-assisted translation error rate
- If you're building real MT systems, you do user studies. In academia, you mostly use BLEU




slide from G. Doddington (NIST)

# Word Alignment

# Word Alignment

- Input: a bitext, pairs of translated sentences

  nous acceptons votre opinion . ||| we accept your view

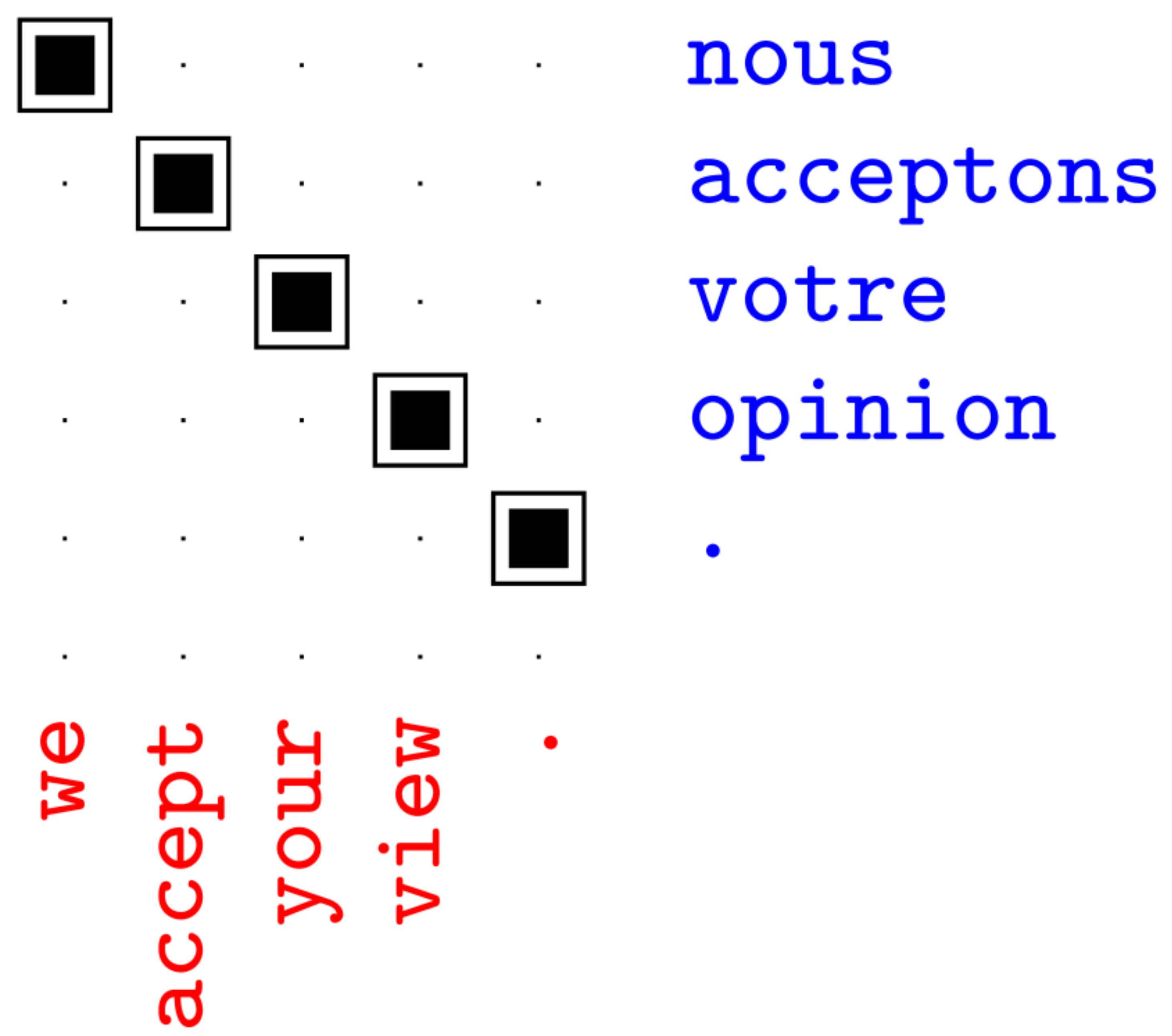
  nous allons changer d’avis ||| we are going to change our minds

# Word Alignment

- Input: a bitext, pairs of translated sentences

  nous acceptons votre opinion . ||| we accept your view

  nous allons changer d'avis ||| we are going to change our minds



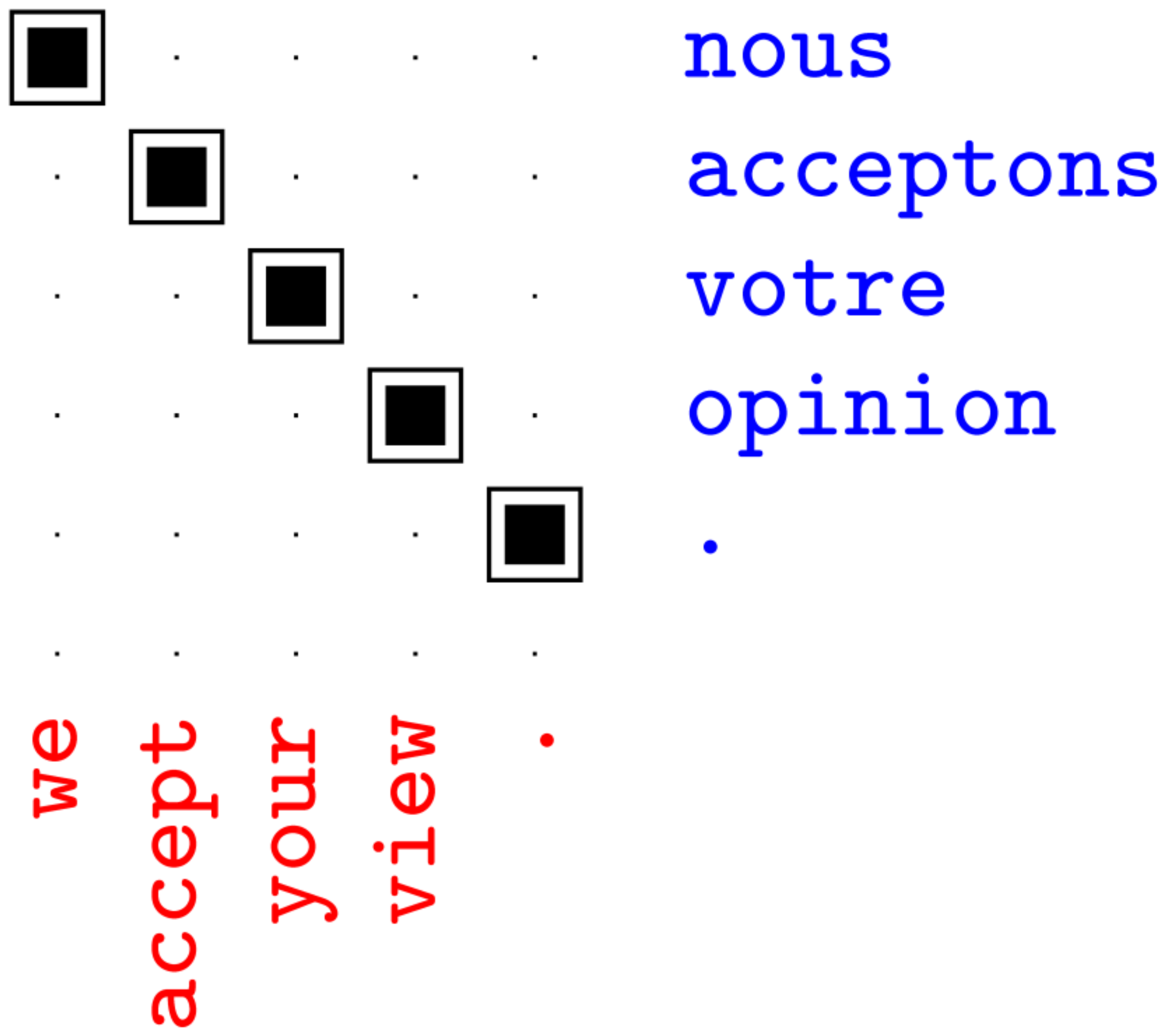
# Word Alignment

- Input: a bitext, pairs of translated sentences

  nous acceptons votre opinion . ||| we accept your view

  nous allons changer d’avis ||| we are going to change our minds

- Output: alignments between words in each sentence



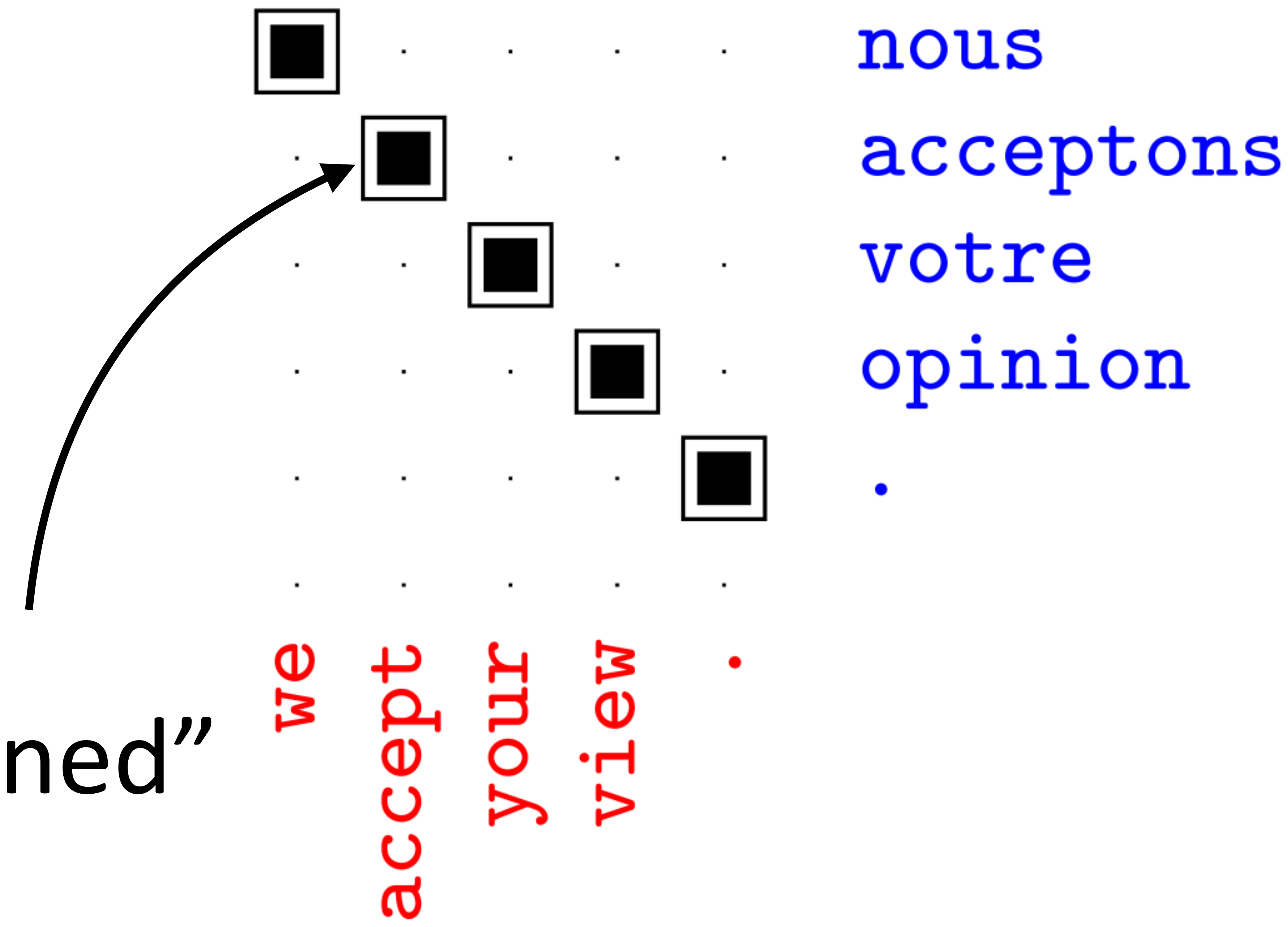
# Word Alignment

- Input: a bitext, pairs of translated sentences

  nous acceptons votre opinion . ||| we accept your view

  nous allons changer d'avis ||| we are going to change our minds

- Output: alignments between words in each sentence




"accept and acceptons are aligned"

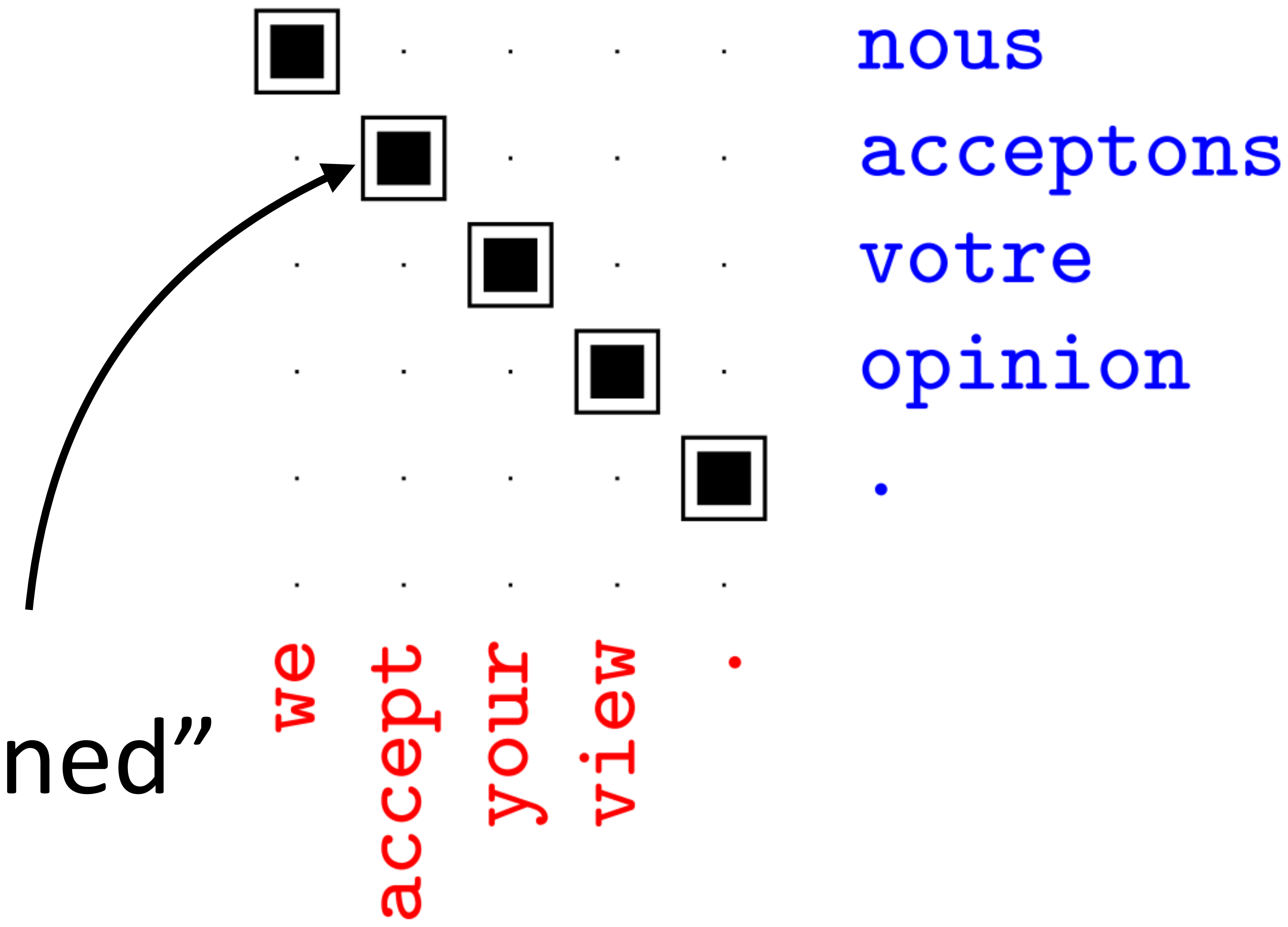
# Word Alignment

- Input: a bitext, pairs of translated sentences

  nous acceptons votre opinion . ||| we accept your view

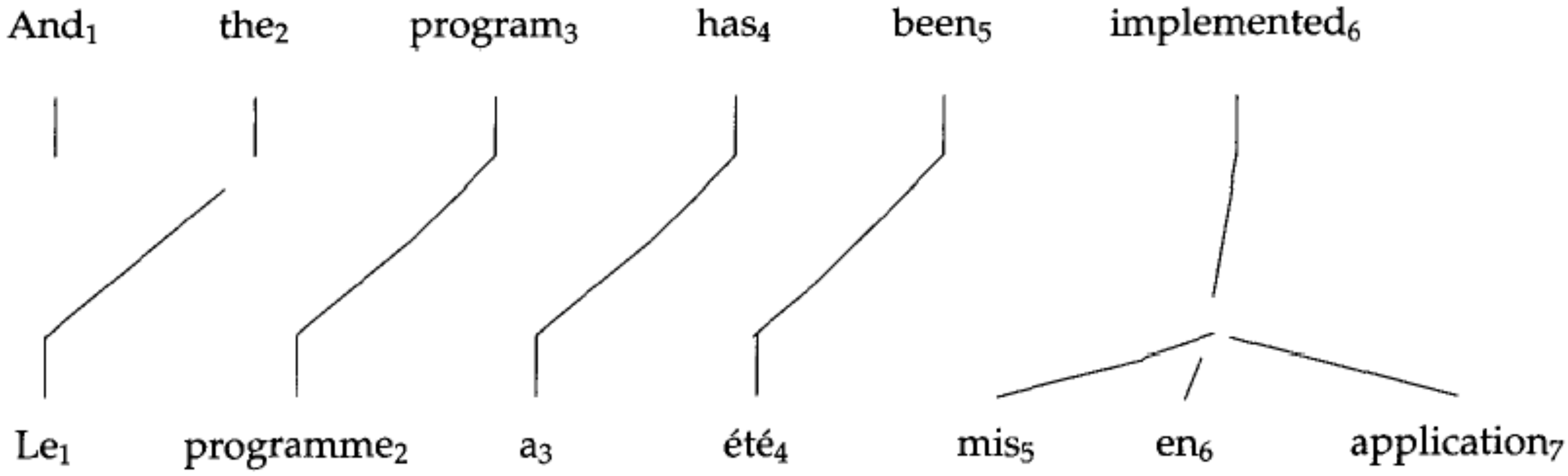
  nous allons changer d’avis ||| we are going to change our minds

- Output: alignments between words in each sentence
  - We will see how to turn these into phrases




“accept and acceptons are aligned”

# 1-to-Many Alignments



# Word Alignment

- Models P(**f**|**e**): probability of "French" sentence being generated from "English" sentence according to a model

# Word Alignment

- Models P(**f**|**e**): probability of "French" sentence being generated from "English" sentence according to a model

- Latent variable model: $P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}|\mathbf{a}, \mathbf{e}) P(\mathbf{a})$

# Word Alignment

- Models P(**f**|**e**): probability of "French" sentence being generated from "English" sentence according to a model
- Latent variable model: $P(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \sum_{\mathbf{a}} P(\mathbf{f}|\mathbf{a}, \mathbf{e}) P(\mathbf{a})$
- Correct alignments should lead to higher-likelihood generations, so by optimizing this objective we will learn correct alignments

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i)$$

Brown et al. (1993)

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i)$$

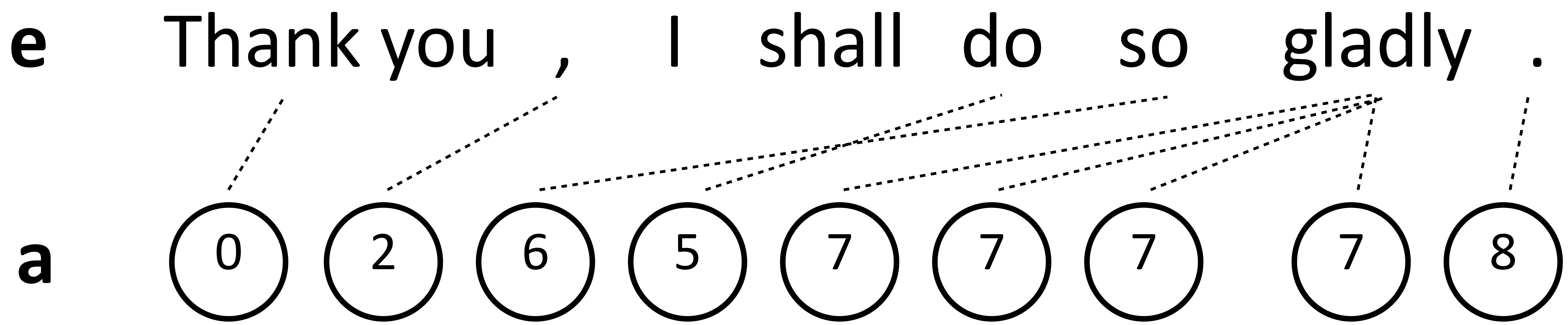
**e** Thank you , I shall do so gladly .

Brown et al. (1993)

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a} | \mathbf{e}) = \prod_{i=1}^{n} P(f_i | e_{a_i}) P(a_i)$$

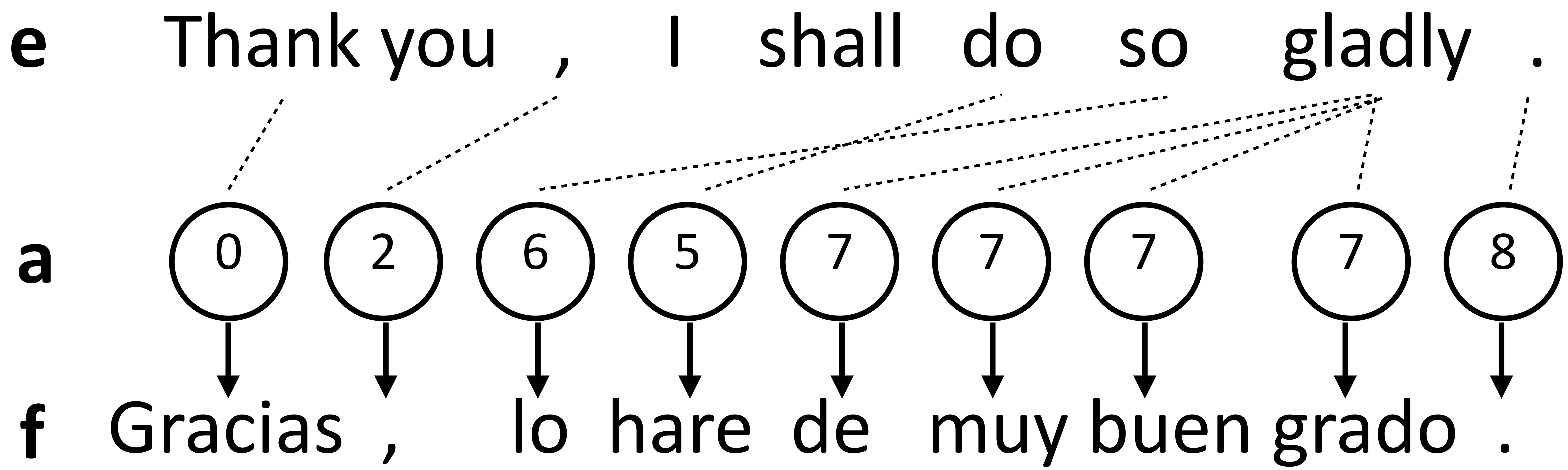



Brown et al. (1993)

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a} | \mathbf{e}) = \prod_{i=1}^{n} P(f_i | e_{a_i}) P(a_i)$$

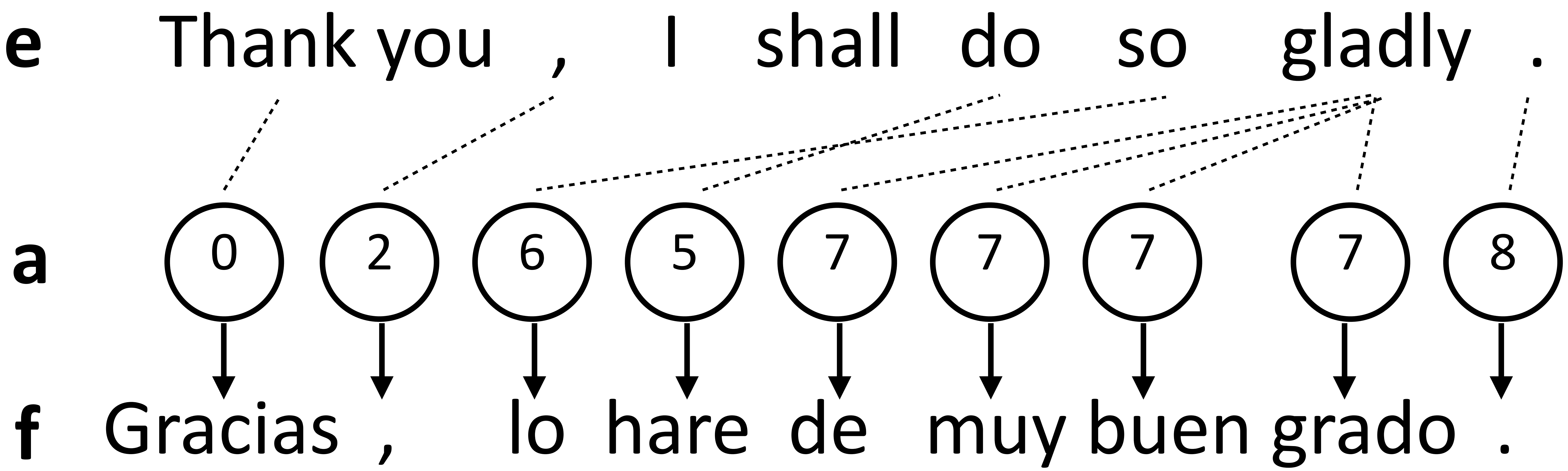



Brown et al. (1993)

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i)$$



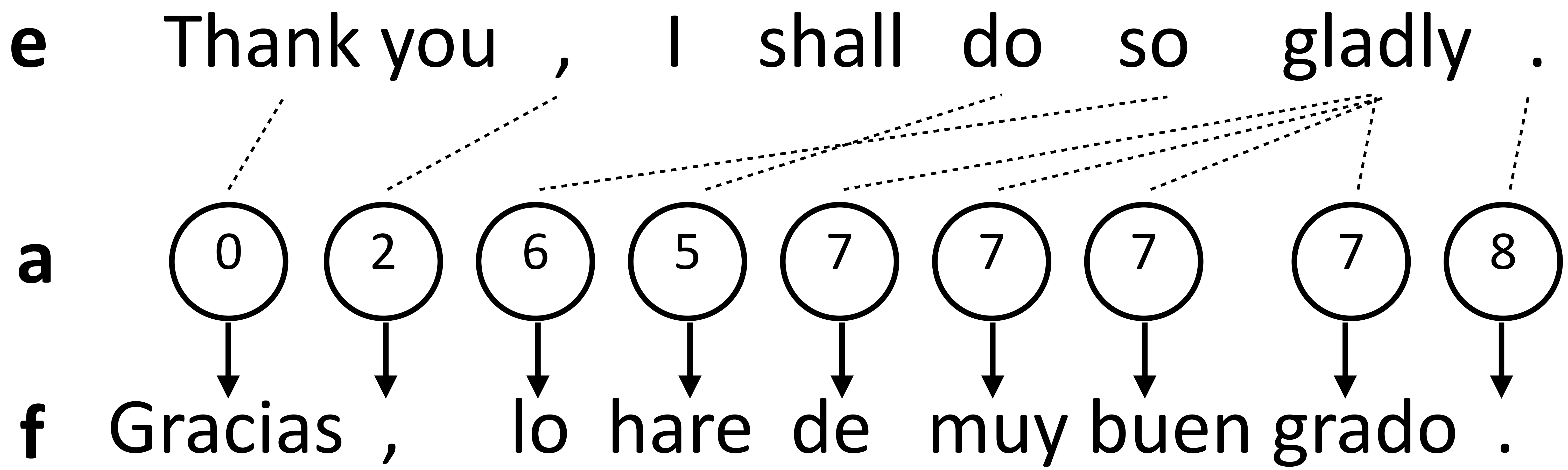

- Set P(a) uniformly (no prior over good alignments)

Brown et al. (1993)

# IBM Model 1

- Each French word is aligned to *at most* one English word

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i)$$



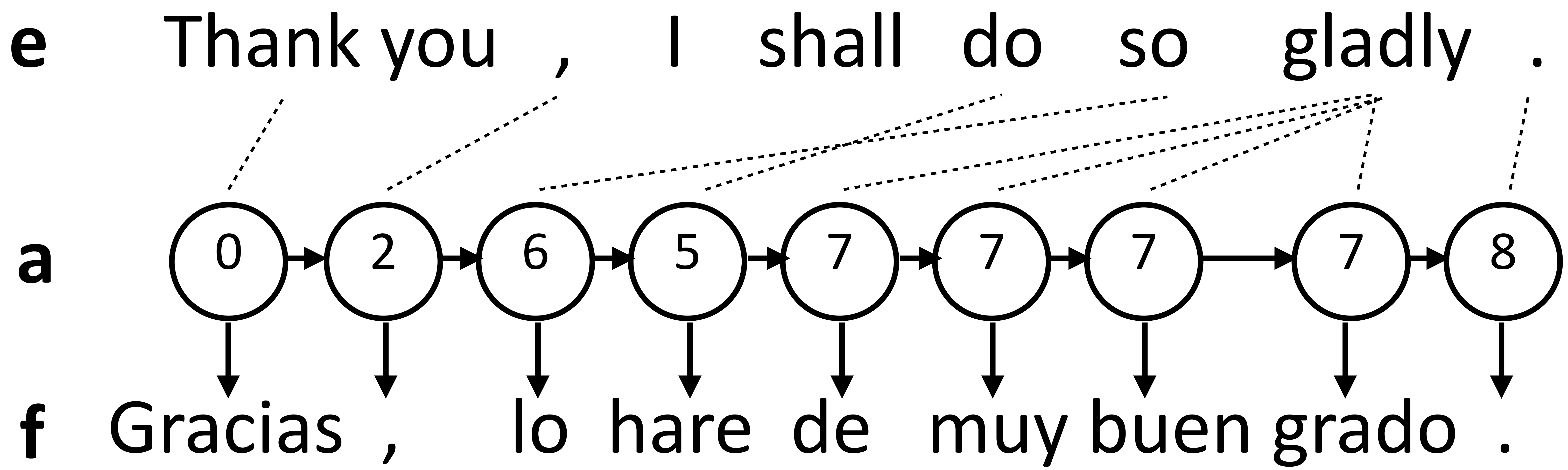

- Set P(a) uniformly (no prior over good alignments)
- $P(f_i|e_{a_i})$: word translation probability table

Brown et al. (1993)

# HMM for Alignment

- Sequential dependence between a's to capture monotonicity

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i|a_{i-1})$$

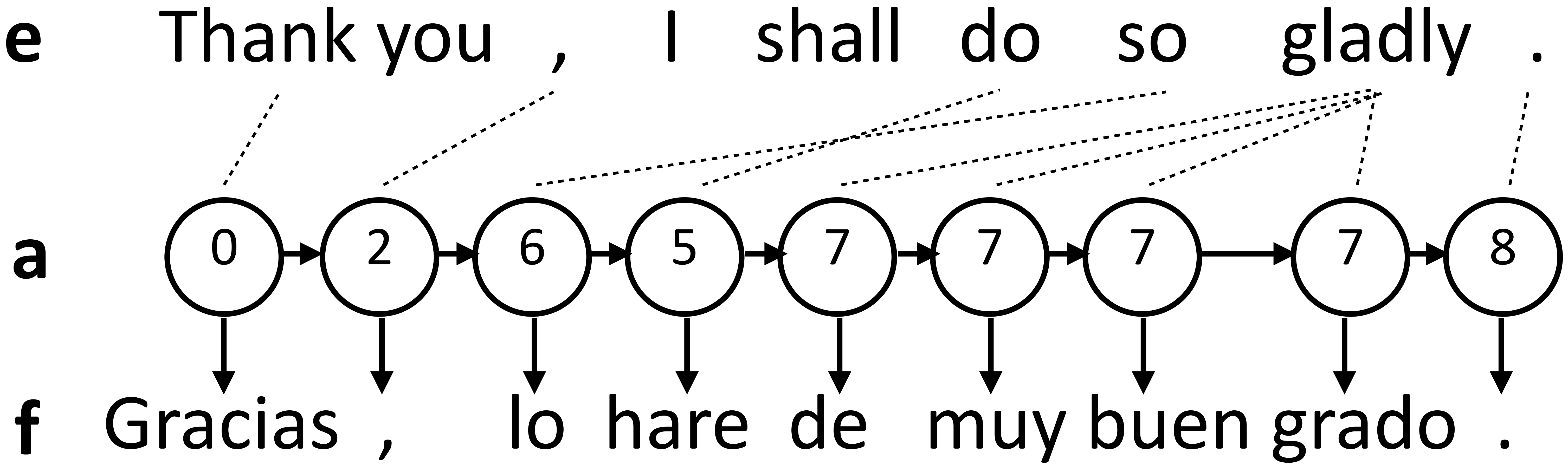



Brown et al. (1993)

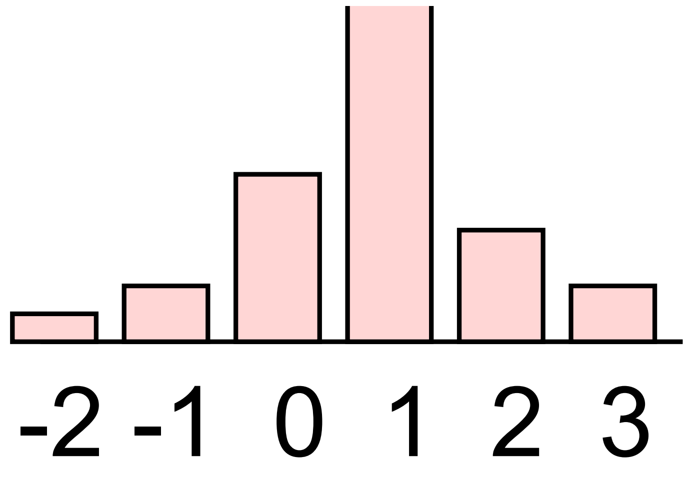
# HMM for Alignment

- Sequential dependence between a's to capture monotonicity

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i|a_{i-1})$$




- Alignment dist parameterized by jump size: $P(a_j - a_{j-1})$

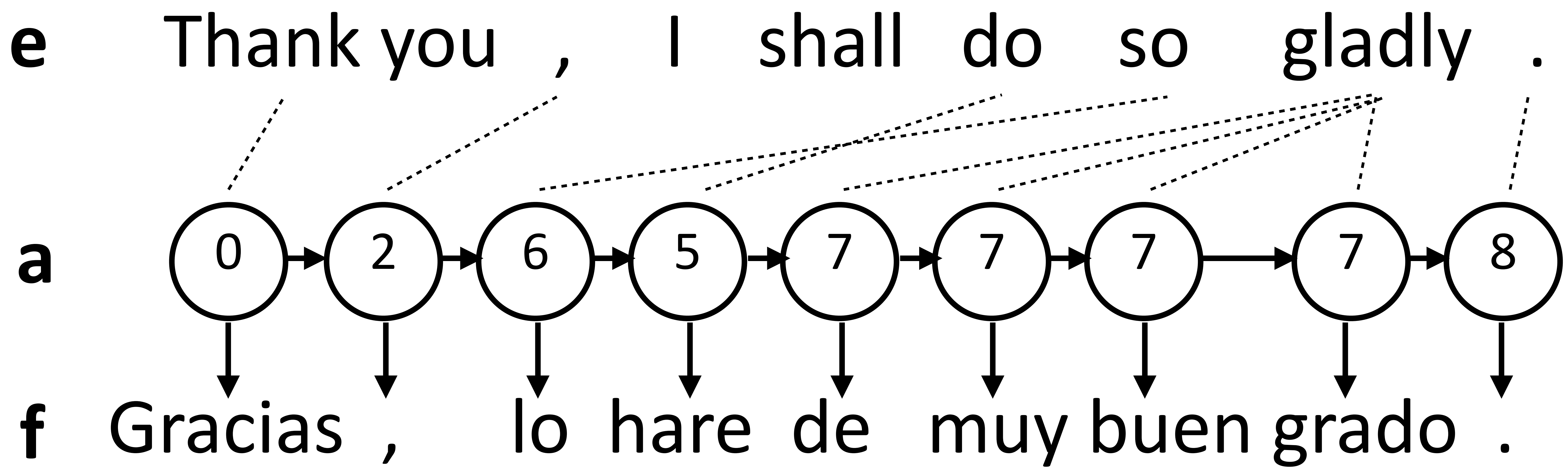



Brown et al. (1993)

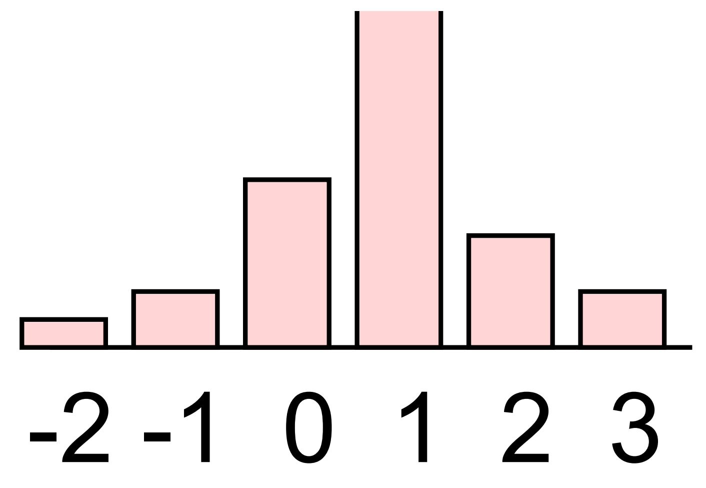
# HMM for Alignment

- Sequential dependence between a's to capture monotonicity

$$P(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \prod_{i=1}^{n} P(f_i|e_{a_i})P(a_i|a_{i-1})$$




- Alignment dist parameterized by jump size: $P(a_j - a_{j-1})$

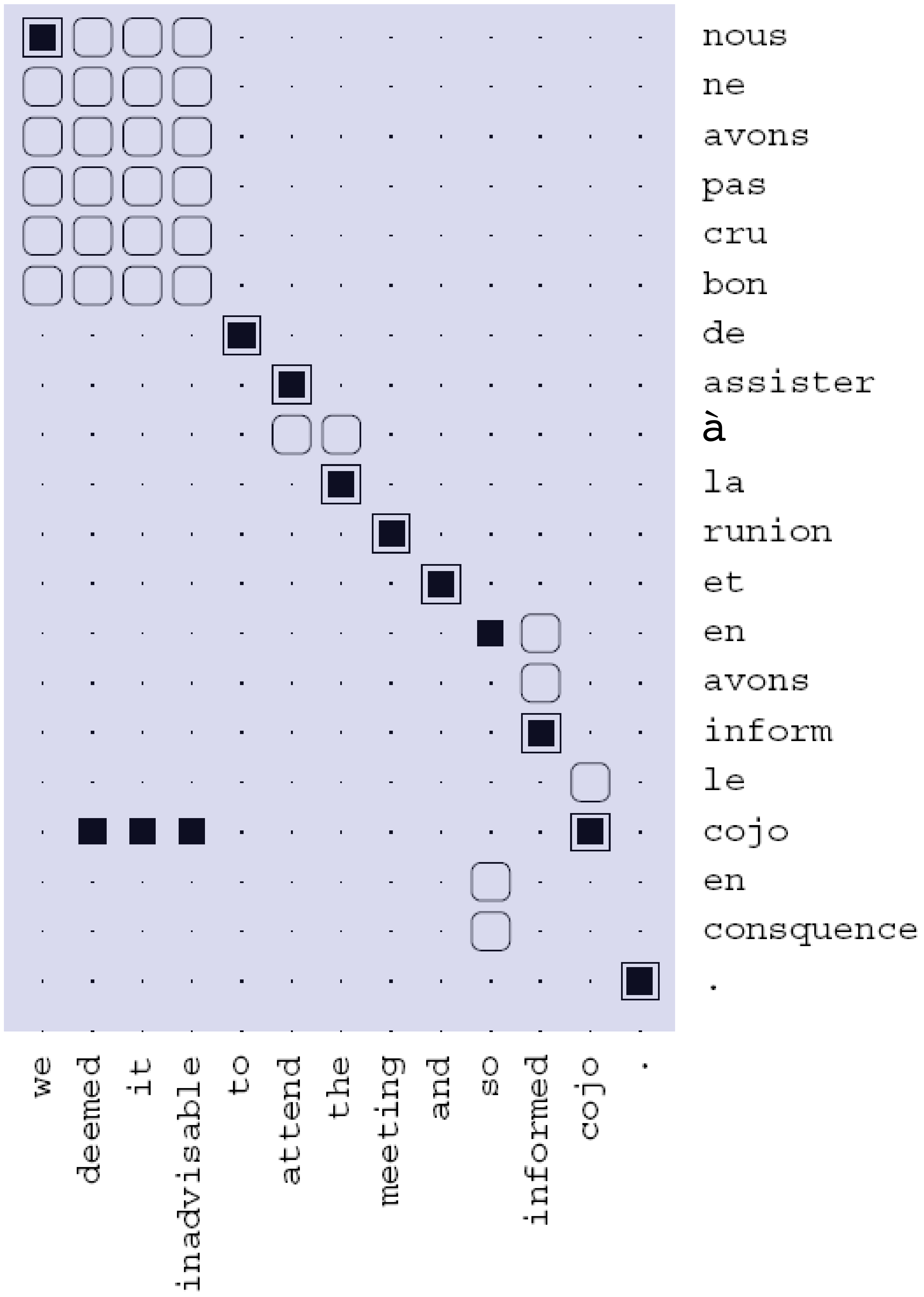



- $P(f_i|e_{a_i})$: same as before

Brown et al. (1993)

# HMM Model

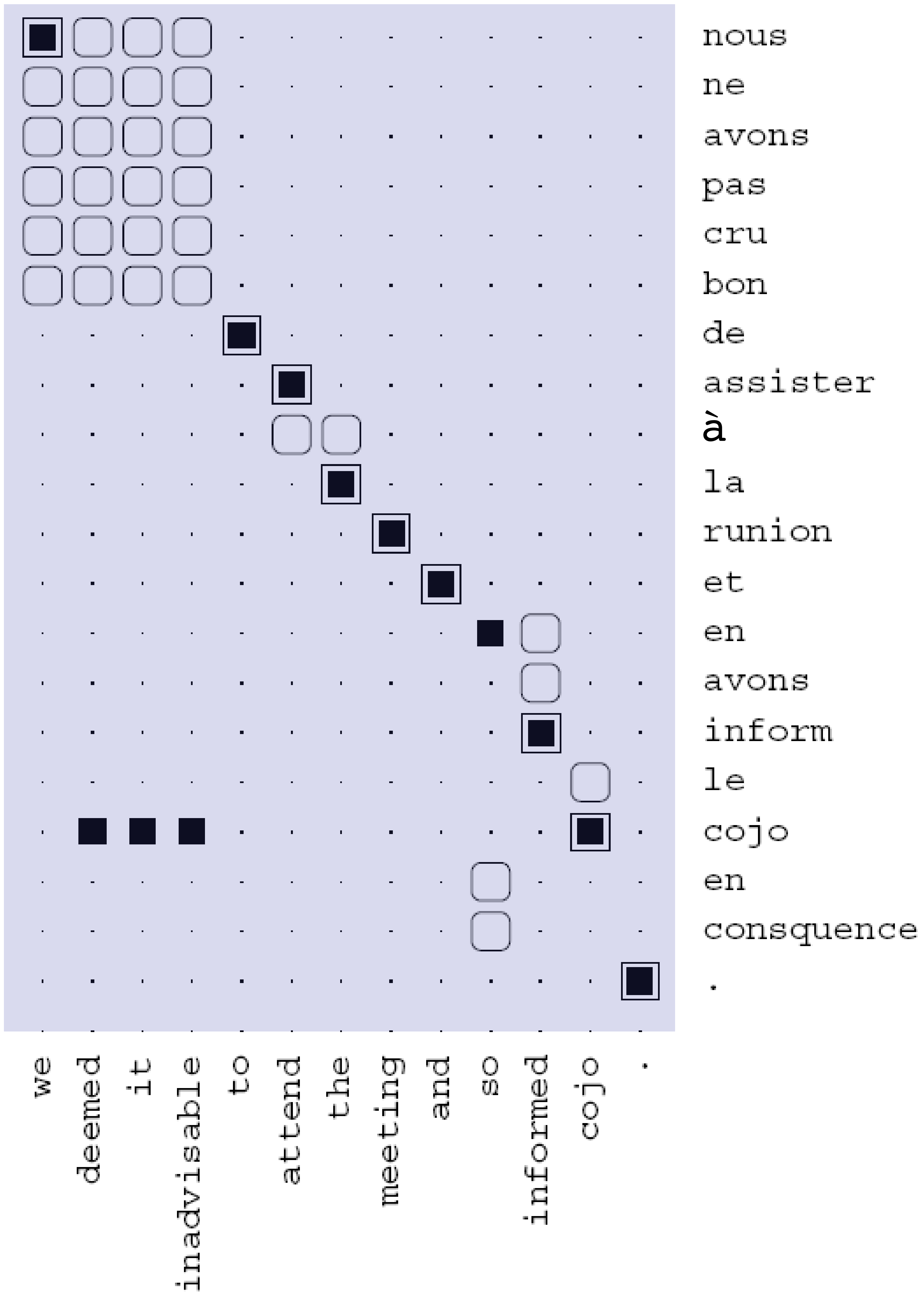
- Which direction is this?



# HMM Model

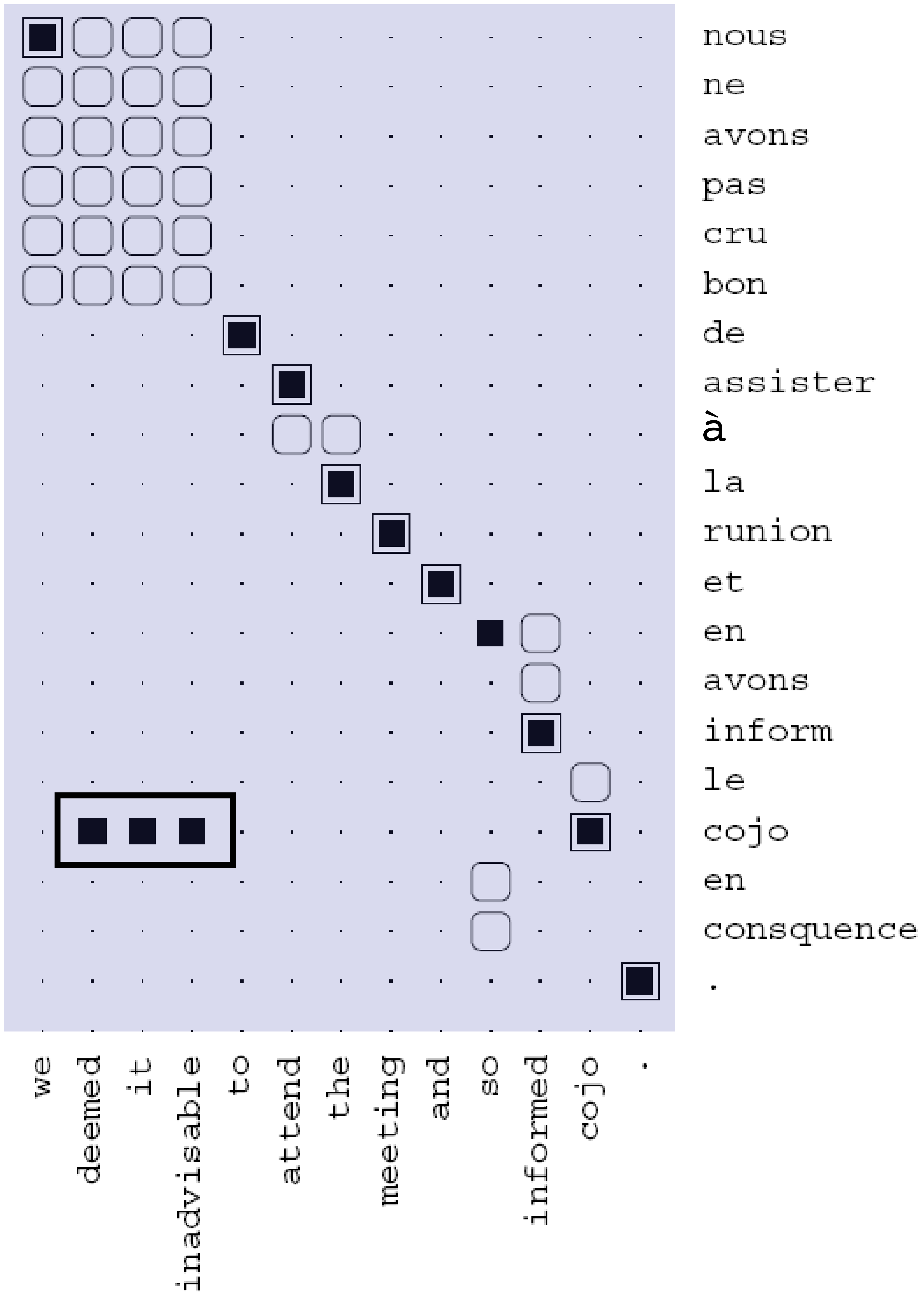
- Which direction is this?

- Alignments are generally monotonic (along diagonal)



# HMM Model

- Which direction is this?
- Alignments are generally monotonic (along diagonal)
- Some mistakes, especially when you have rare words (*garbage collection*)



# Evaluating Word Alignment

- “Alignment error rate”: use labeled alignments on small corpus

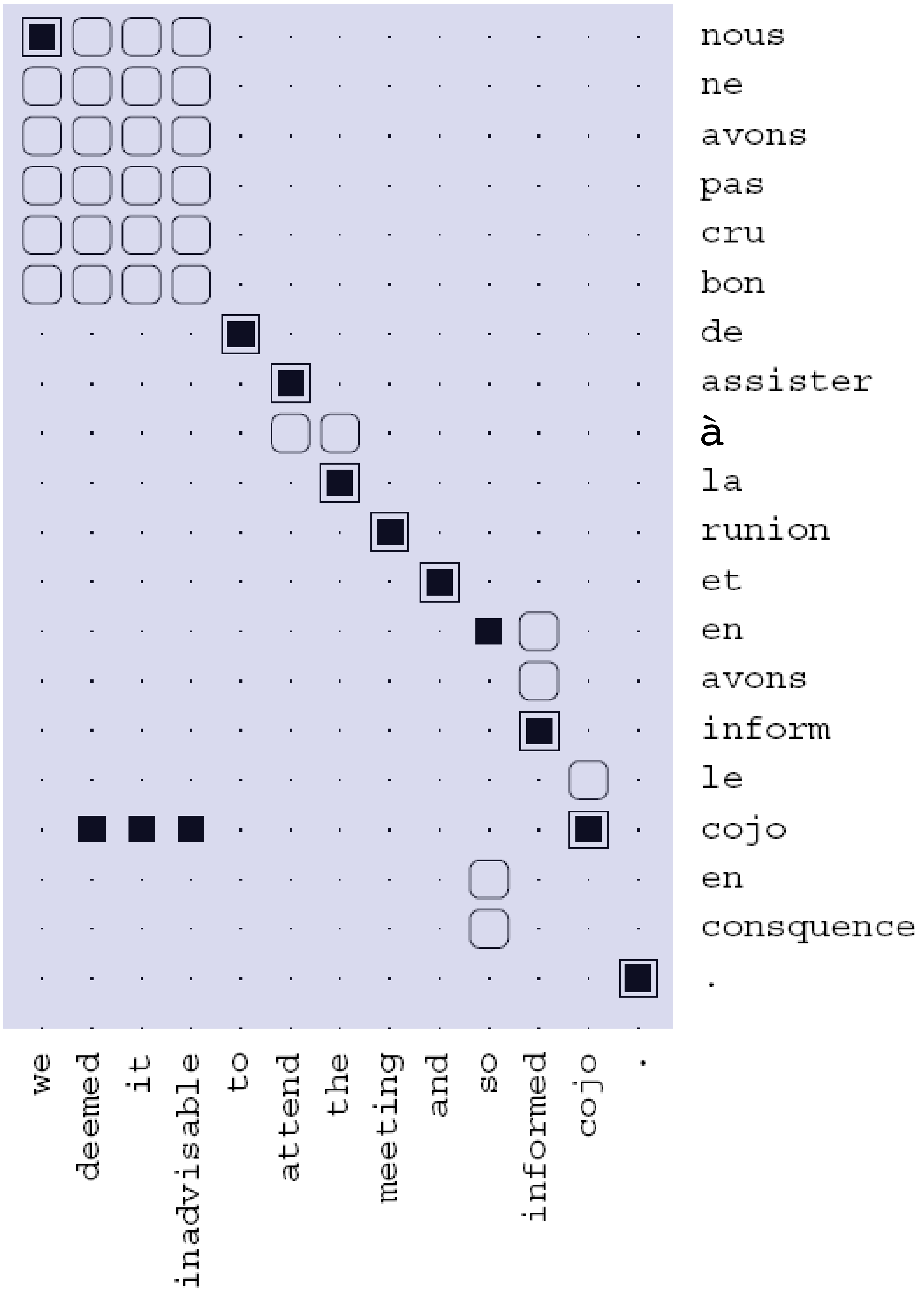
| Model | AER |
|---|---|
| Model 1 INT | 19.5 |
| HMM E→F | 11.4 |
| HMM F→E | 10.8 |
| HMM AND | 7.1 |
| HMM INT | 4.7 |
| GIZA M4 AND | 6.9 |

- Run Model 1 in both directions and intersect “intelligently”

- Run HMM model in both directions and intersect “intelligently”

# Phrase Extraction

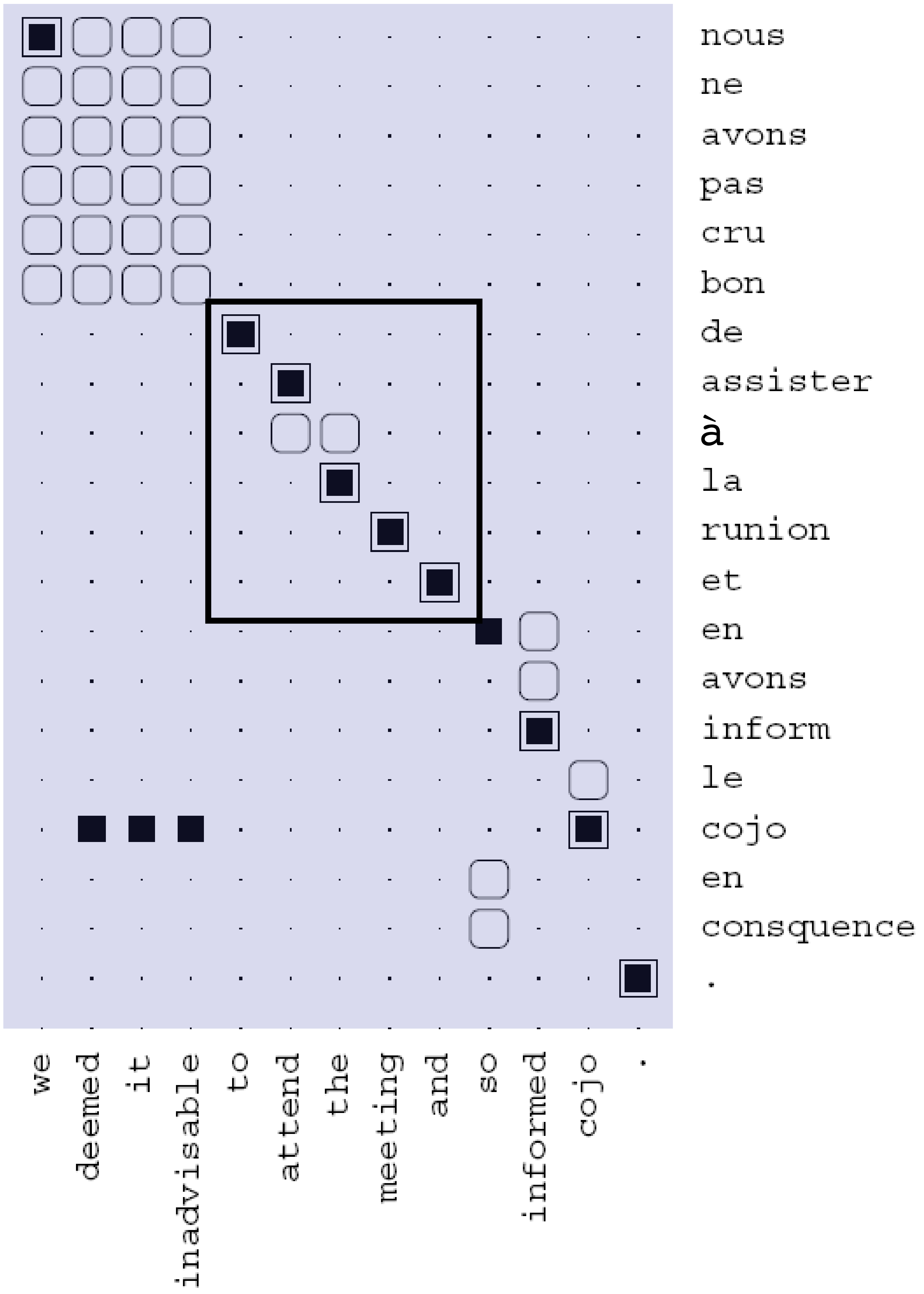
- Find contiguous sets of aligned words in the two languages that don't have alignments to other words



# Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

d'assister à la reunion et ||| to attend the meeting and

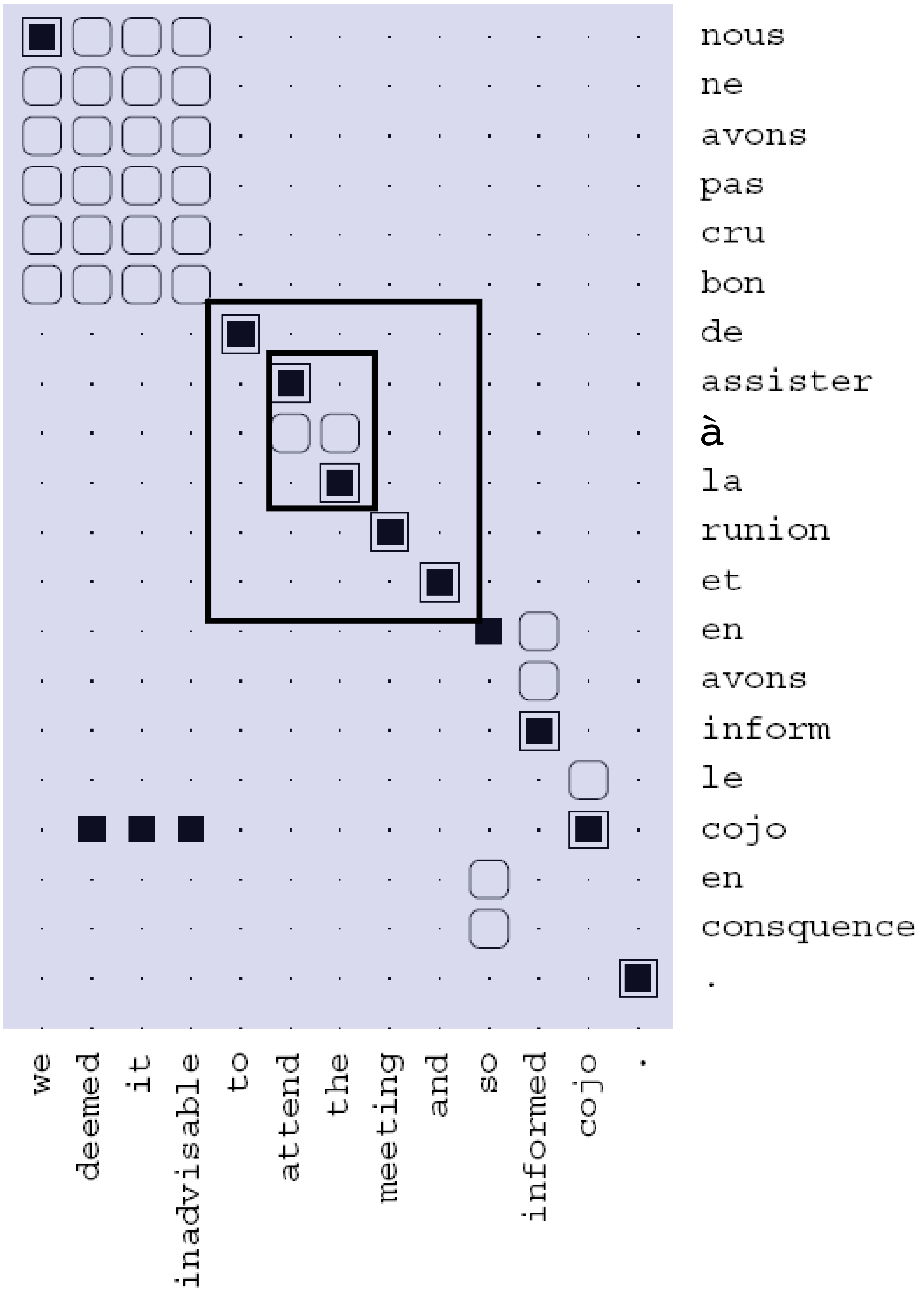


# Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

d'assister à la reunion et ||| to attend the meeting and

assister à la reunion ||| attend the meeting



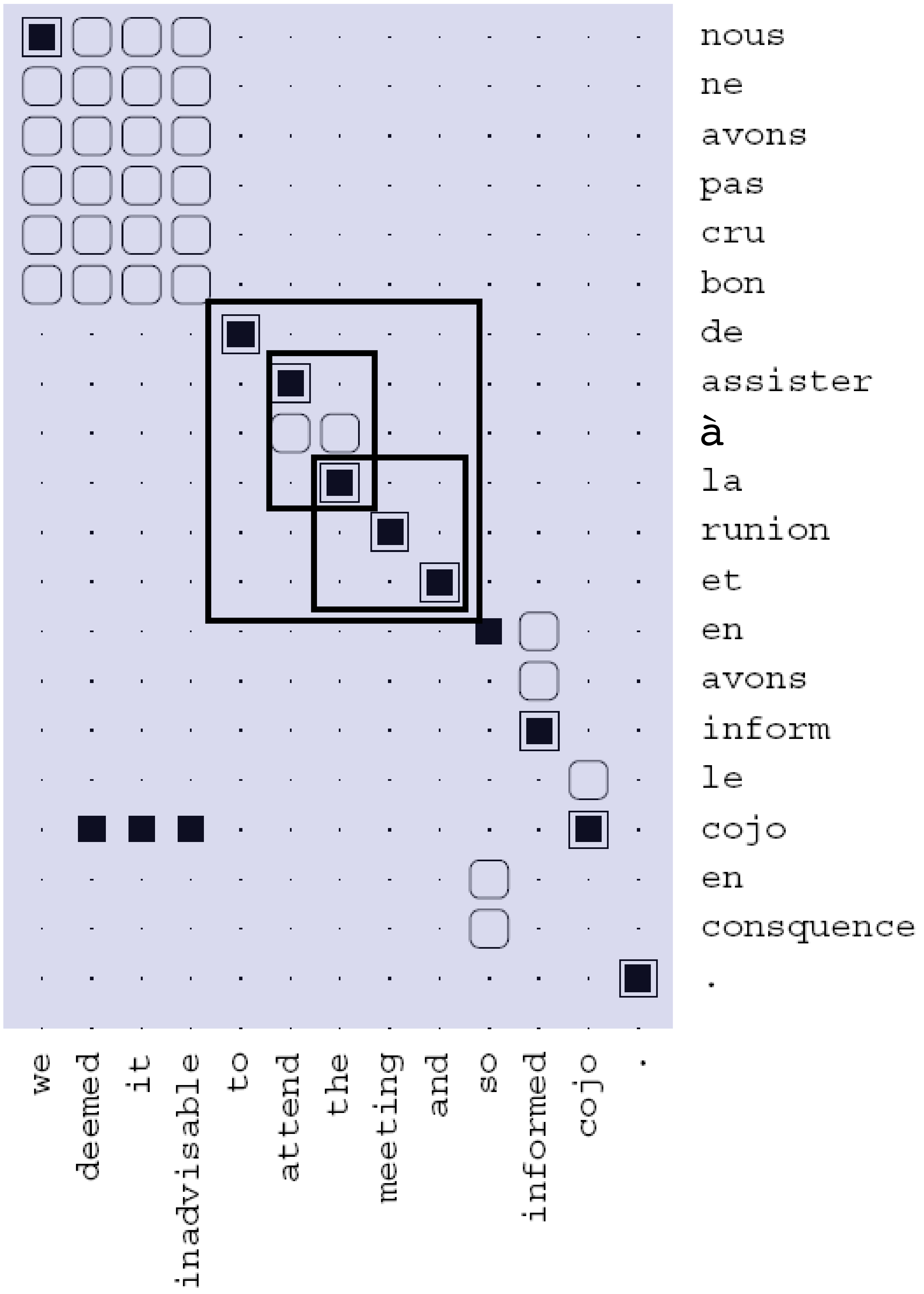
# Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

d'assister à la reunion et ||| to attend the meeting and

assister à la reunion ||| attend the meeting

la reunion and ||| the meeting and



# Phrase Extraction

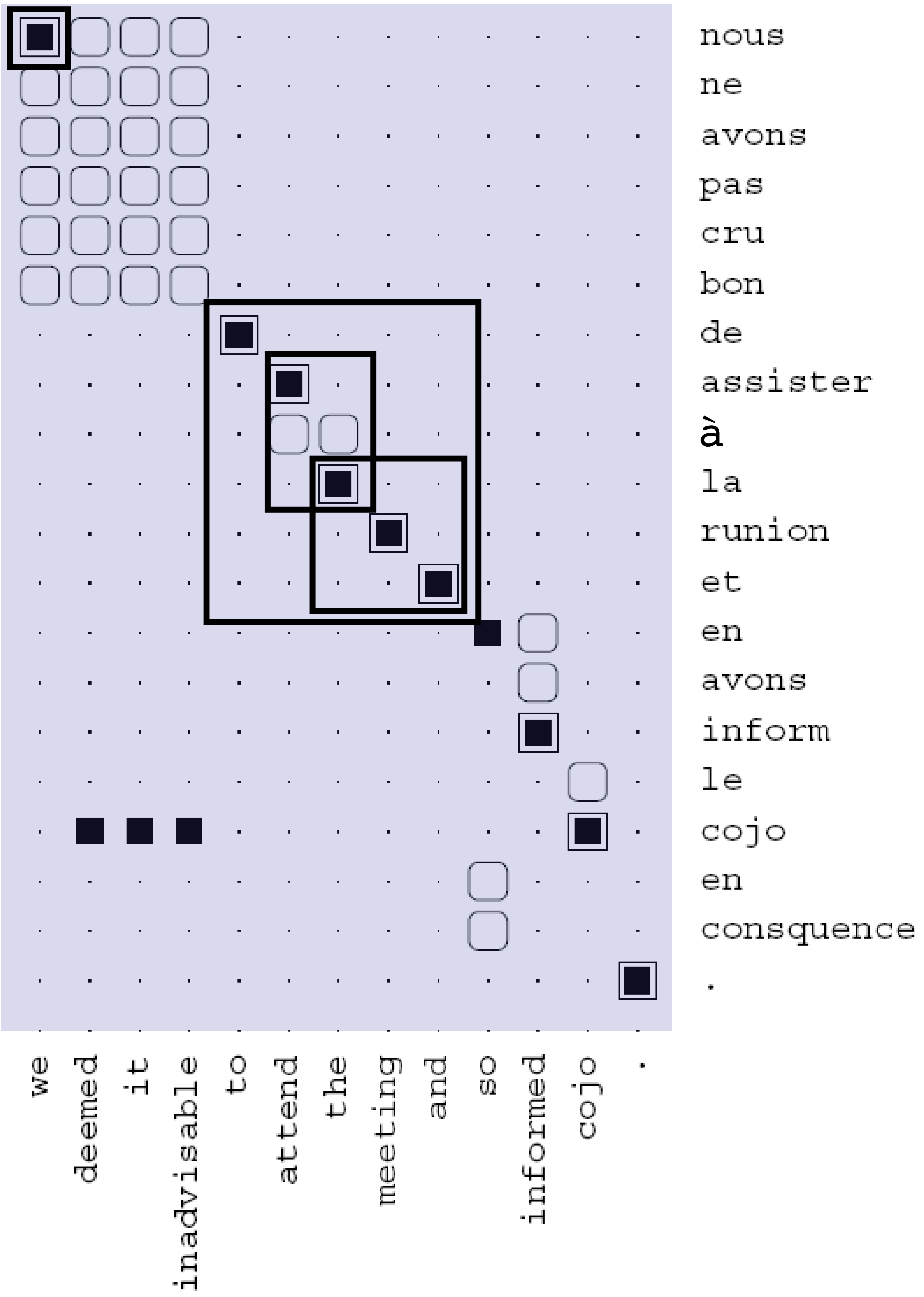
- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

d'assister à la reunion et ||| to attend the meeting and

assister à la reunion ||| attend the meeting

la reunion and ||| the meeting and

nous ||| we



# Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

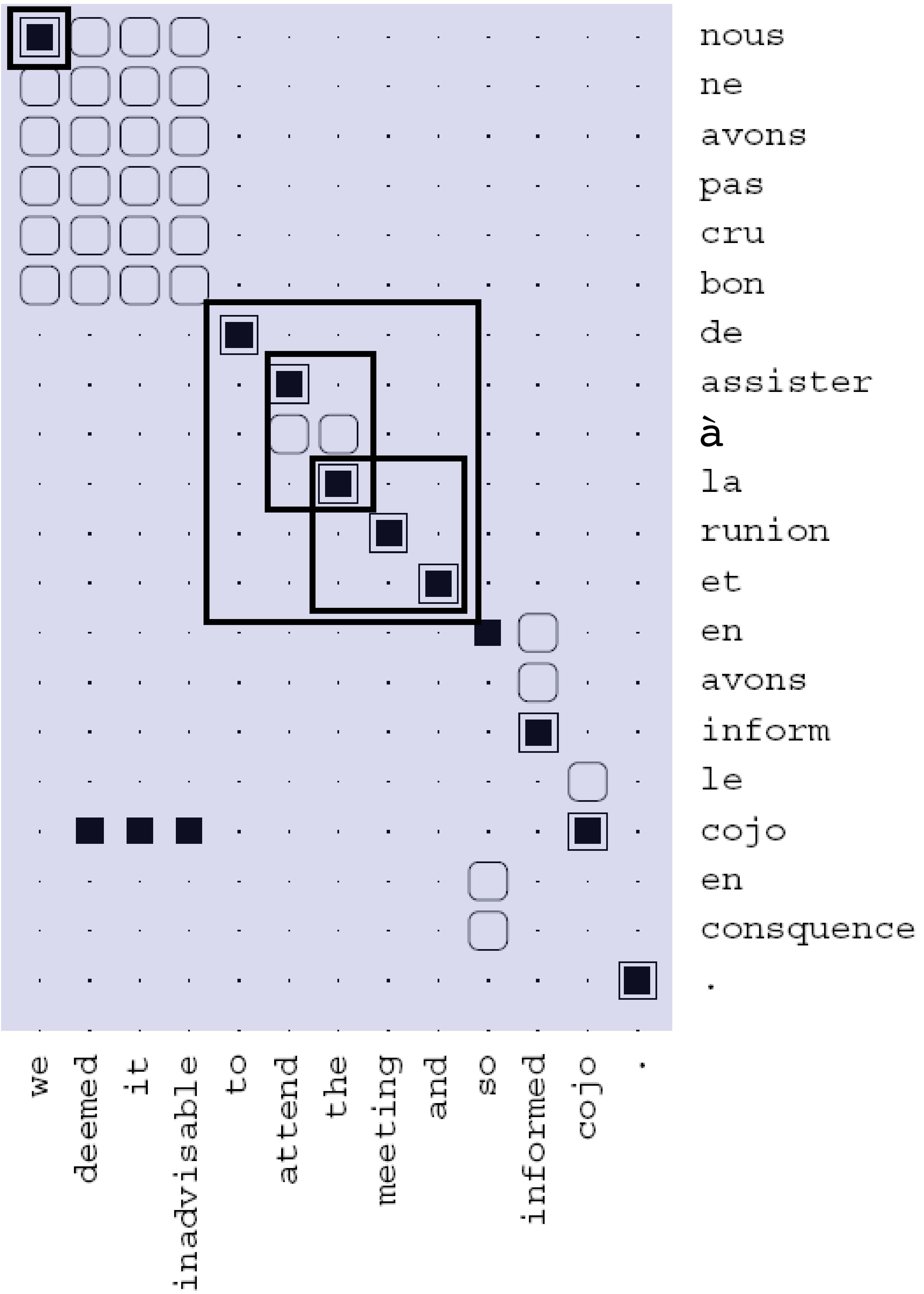
d'assister à la reunion et ||| to attend the meeting and

assister à la reunion ||| attend the meeting

la reunion and ||| the meeting and

nous ||| we

…



# Phrase Extraction

- Find contiguous sets of aligned words in the two languages that don't have alignments to other words

d'assister à la reunion et ||| to attend the meeting and

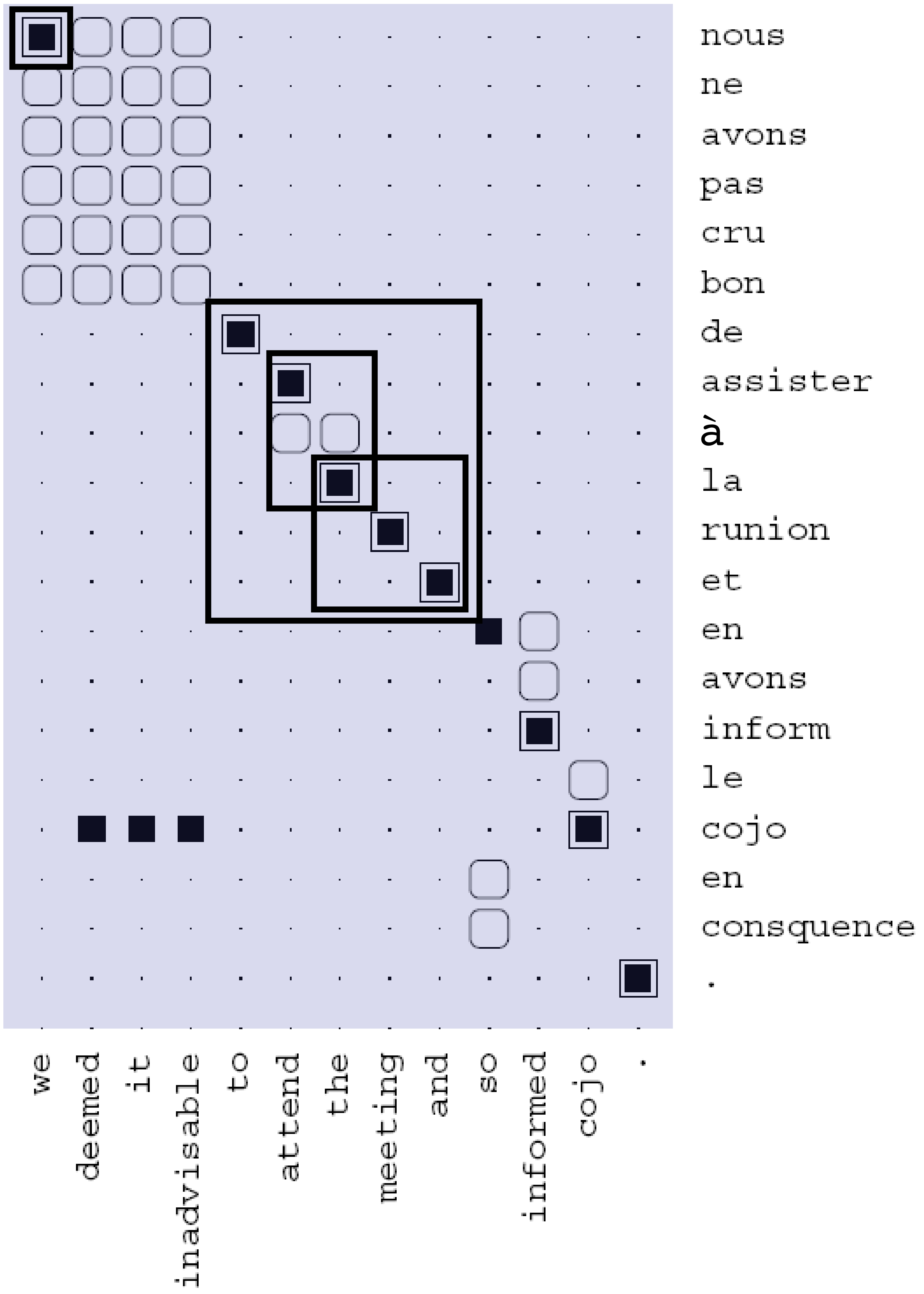
assister à la reunion ||| attend the meeting

la reunion and ||| the meeting and

nous ||| we

…

- Lots of phrases possible, count across all sentences and score by frequency

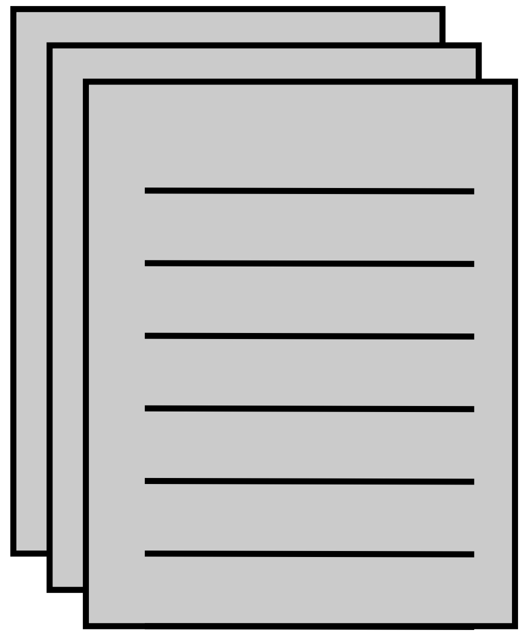


# Language Modeling

# Phrase-Based MT

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Language model P(e)

Unlabeled English data

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

“Translate faithfully but make fluent English”

# N-gram Language Models

I visited San ______ put a distribution over the next word

# N-gram Language Models

I visited San ______     put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

# N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

# N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

Maximum likelihood estimate of this probability from a corpus

# N-gram Language Models

I visited San ______ put a distribution over the next word

- Simple generative model: distribution of next word is a multinomial distribution conditioned on previous n-1 words

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})}$$

Maximum likelihood estimate of this probability from a corpus

- Just relies on counts, even in 2008 could scale up to 1.3M word types, 4B n-grams (all 5-grams occurring >40 times on the Web)

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

$$P(x|\text{visited San}) = (1 - \lambda)\frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

# Smoothing N-gram Language Models

I visited San ______      put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

$$P(x|\text{visited San}) = (1-\lambda)\frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

smooth this too!

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

$$P(x|\text{visited San}) = (1-\lambda)\frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

smooth this too!

- One technique is "absolute discounting:" subtract off constant *k* from numerator, set lambda to make this normalize (*k*=1 is like leave-one-out)

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

$$P(x|\text{visited San}) = (1-\lambda)\frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

smooth this too!

- One technique is "absolute discounting:" subtract off constant *k* from numerator, set lambda to make this normalize (*k*=1 is like leave-one-out)

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x) - k}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

# Smoothing N-gram Language Models

I visited San ______ put a distribution over the next word!

- Smoothing is very important, particularly when using 4+ gram models

$$P(x|\text{visited San}) = (1-\lambda)\frac{\text{count}(\text{visited San}, x)}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

smooth this too!

- One technique is "absolute discounting:" subtract off constant *k* from numerator, set lambda to make this normalize (*k*=1 is like leave-one-out)

$$P(x|\text{visited San}) = \frac{\text{count}(\text{visited San}, x) - k}{\text{count}(\text{visited San})} + \lambda\frac{\text{count}(\text{San}, x)}{\text{count}(\text{San})}$$

- Kneser-Ney smoothing: this trick, plus low-order distributions modified to capture fertilities (how many distinct words appear in a context)

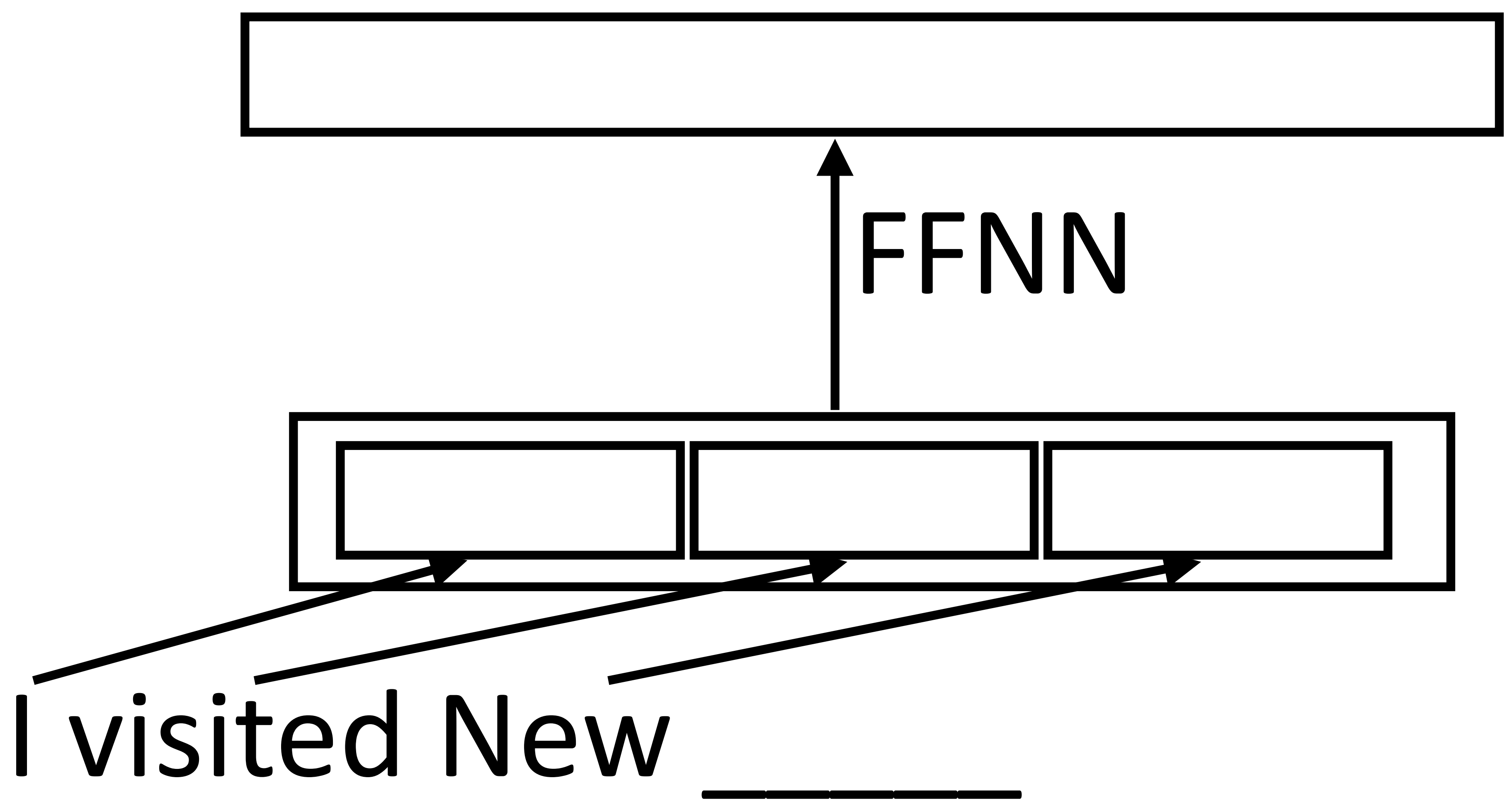
# Neural Language Models

- Early work: feedforward neural networks looking at context

Mnih and Hinton (2003)

# Neural Language Models

- Early work: feedforward neural networks looking at context



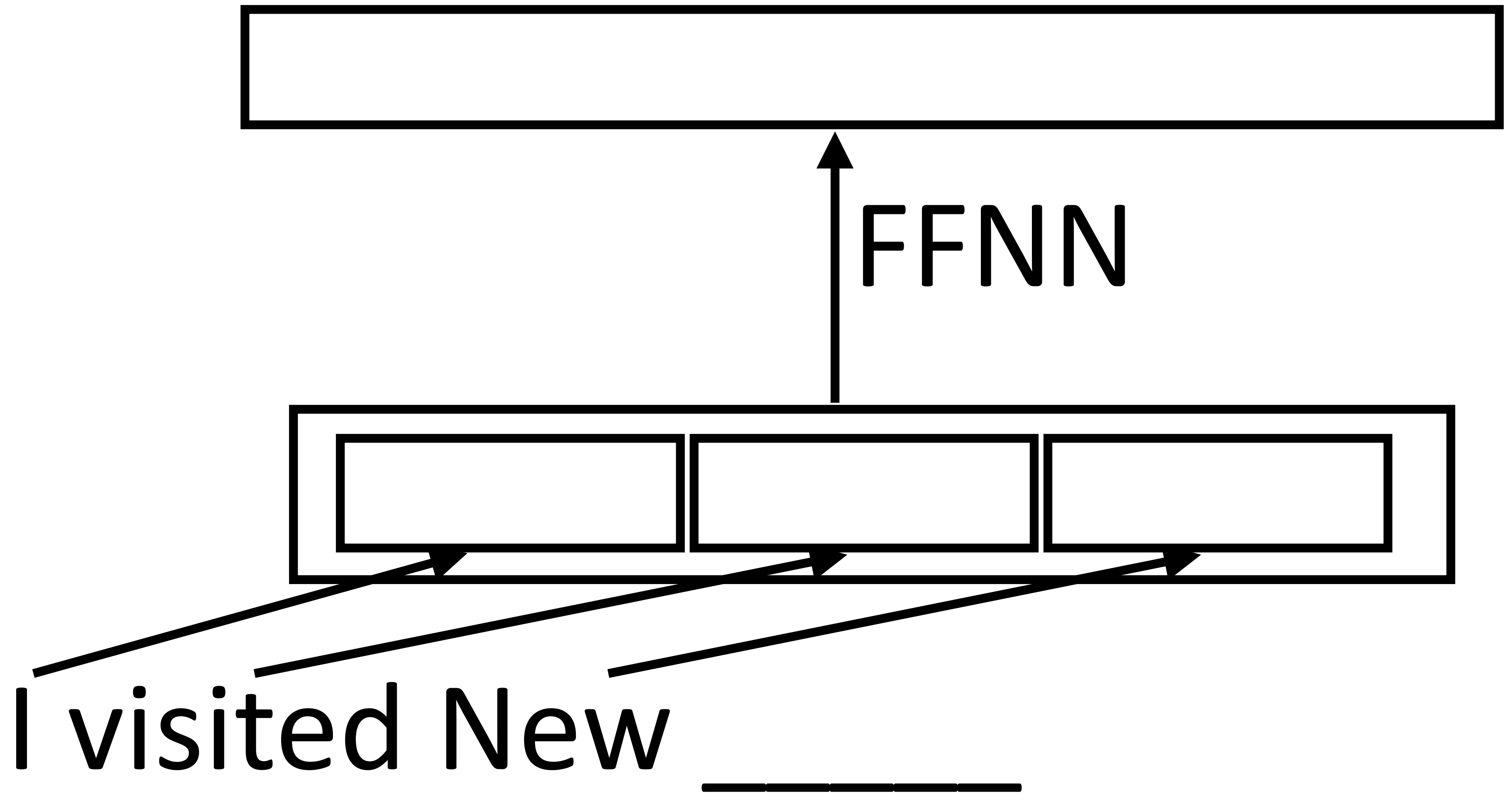

$P(w_i|w_{i-n},\ldots,w_{i-1})$

Mnih and Hinton (2003)

# Neural Language Models

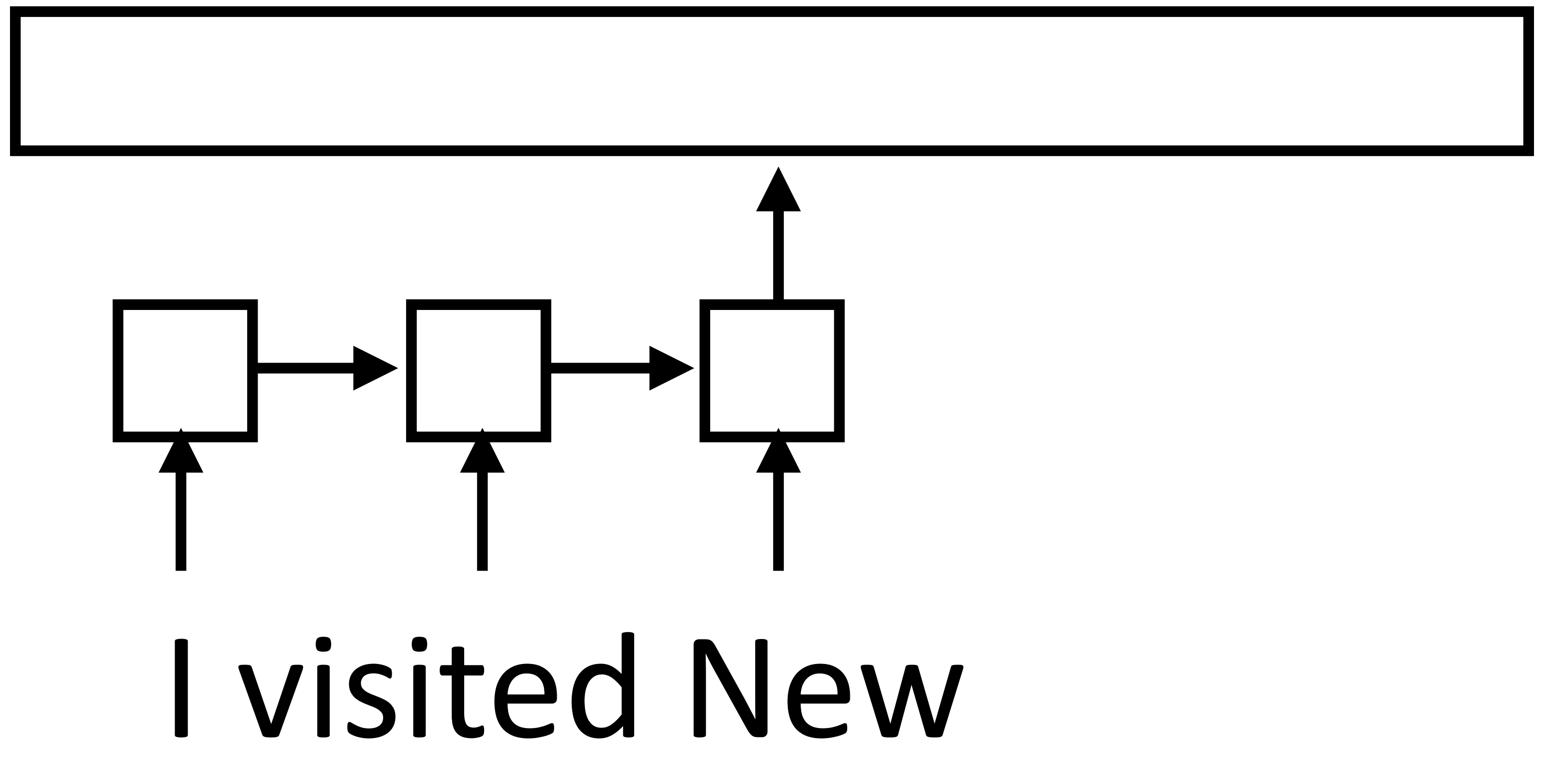
- Early work: feedforward neural networks looking at context

$$P(w_i|w_{i-n},\ldots,w_{i-1})$$




- Variable length context with RNNs:

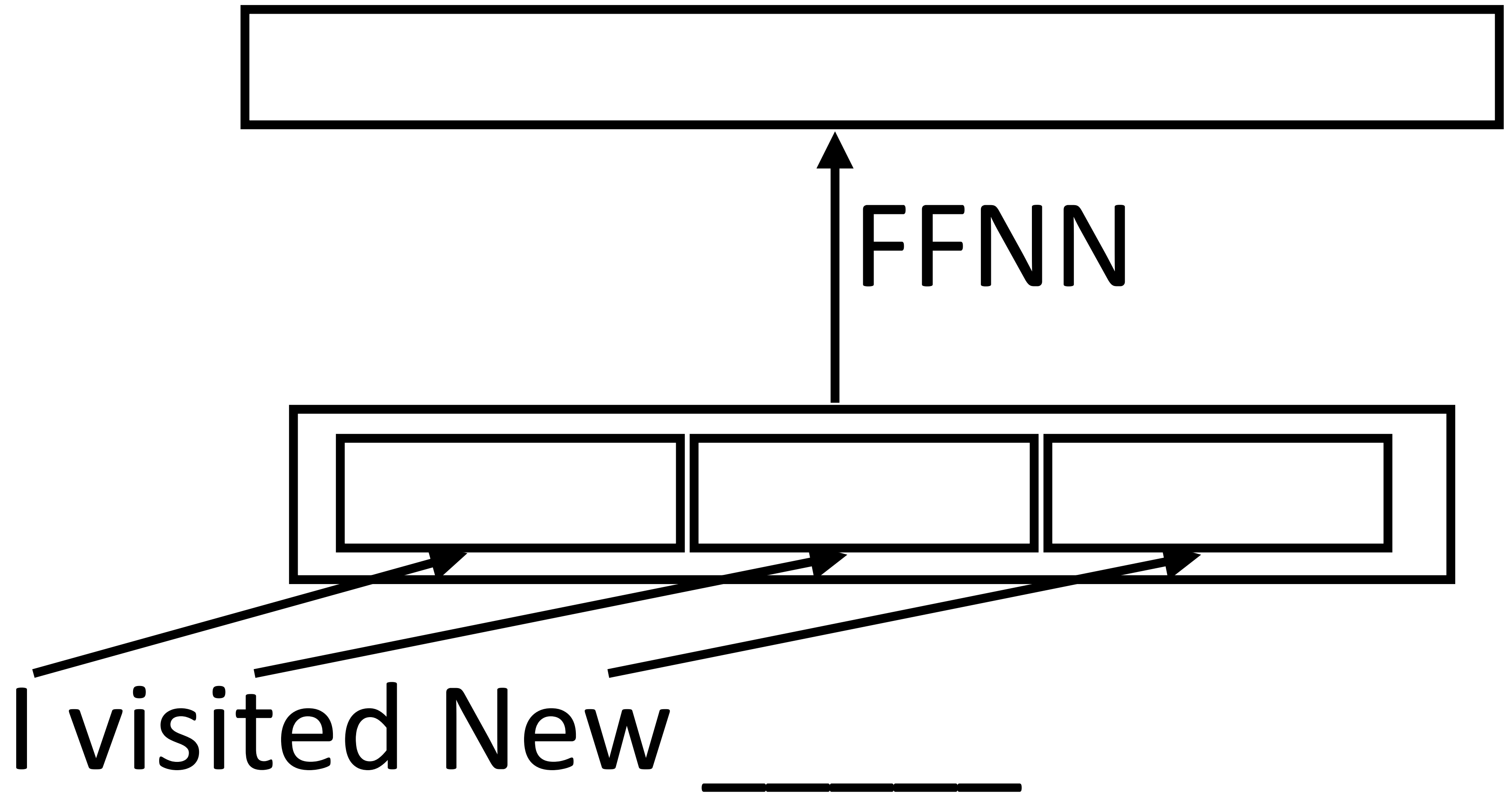
$$P(w_i|w_1,\ldots,w_{i-1})$$




Mnih and Hinton (2003)

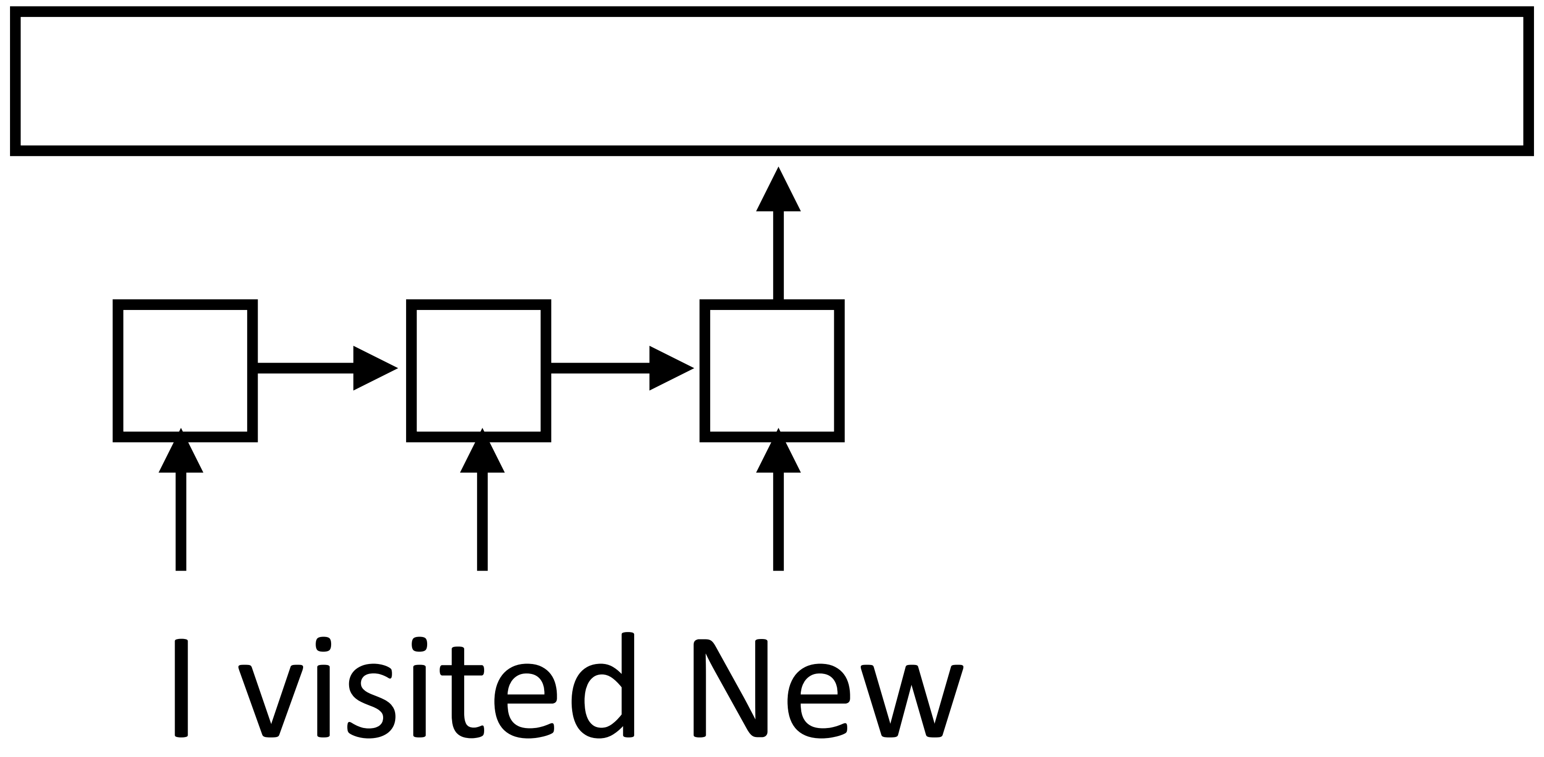
# Neural Language Models

- Early work: feedforward neural networks looking at context




$$P(w_i | w_{i-n}, \ldots, w_{i-1})$$

$$P(w_i | w_1, \ldots, w_{i-1})$$



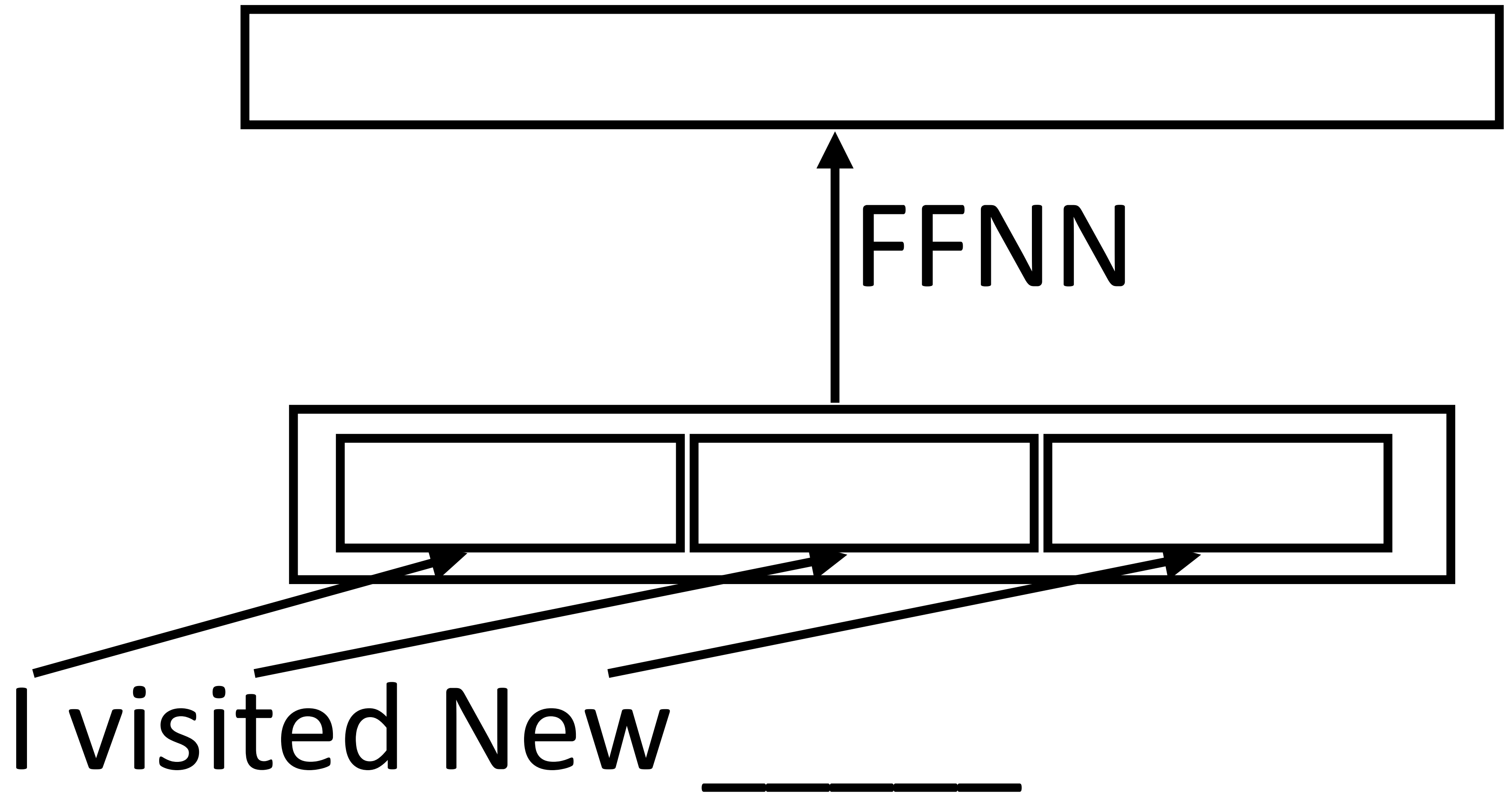

- Variable length context with RNNs:
  - Works like a decoder with no encoder

Mnih and Hinton (2003)

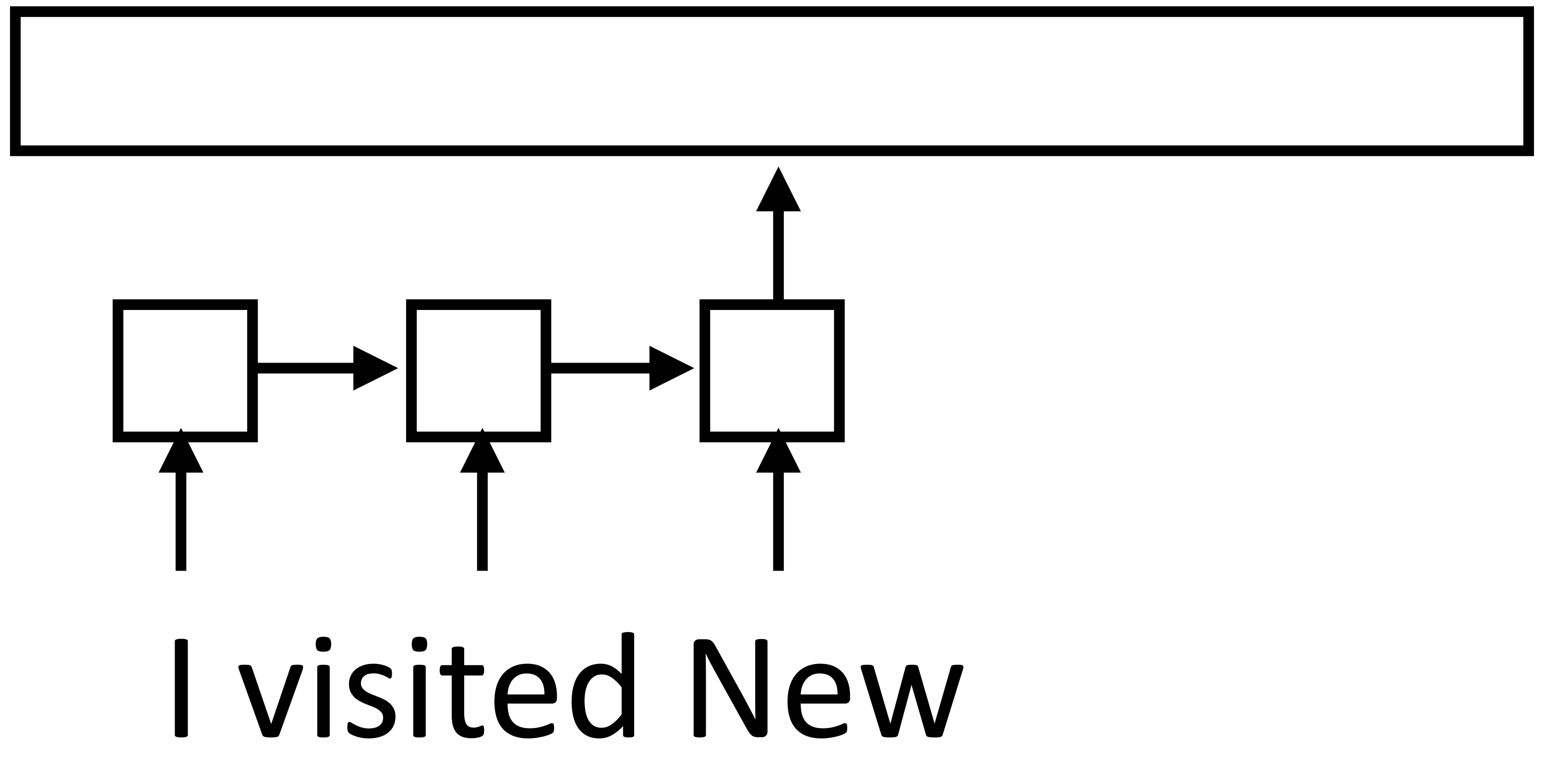
# Neural Language Models

- Early work: feedforward neural networks looking at context

$$P(w_i | w_{i-n}, \ldots, w_{i-1})$$




$$P(w_i | w_1, \ldots, w_{i-1})$$




- Variable length context with RNNs:
  - Works like a decoder with no encoder

- Slow to train over lots of data!

Mnih and Hinton (2003)

# Evaluation

# Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$

# Evaluation

- (One sentence) negative log likelihood: $$\sum_{i=1}^{n} \log p(x_i | x_1, \dots, x_{i-1})$$
- Perplexity: $2^{-\frac{1}{n}\sum_{i=1}^{n} \log_2 p(x_i | x_1, \dots, x_{i-1})}$

# Evaluation

- (One sentence) negative log likelihood: $\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$
- Perplexity: $2^{-\frac{1}{n} \sum_{i=1}^{n} \log_2 p(x_i | x_1, \ldots, x_{i-1})}$
  - NLL (base 2) averaged over the sentence, exponentiated

# Evaluation

- (One sentence) negative log likelihood: $$\sum_{i=1}^{n} \log p(x_i | x_1, \ldots, x_{i-1})$$
- Perplexity: $2^{-\frac{1}{n}\sum_{i=1}^{n} \log_2 p(x_i | x_1, \ldots, x_{i-1})}$
  - NLL (base 2) averaged over the sentence, exponentiated
  - NLL = -2 -> on average, correct thing has prob 1/4 -> PPL = 4. PPL is sort of like branching factor

# Results

Merity et al. (2017), Melis et al. (2017)

# Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark

Merity et al. (2017), Melis et al. (2017)

# Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7

Merity et al. (2017), Melis et al. (2017)

# Results

- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7
- LSTM: PPL ~ 60-80 (depending on how much you optimize it)

Merity et al. (2017), Melis et al. (2017)

# Results

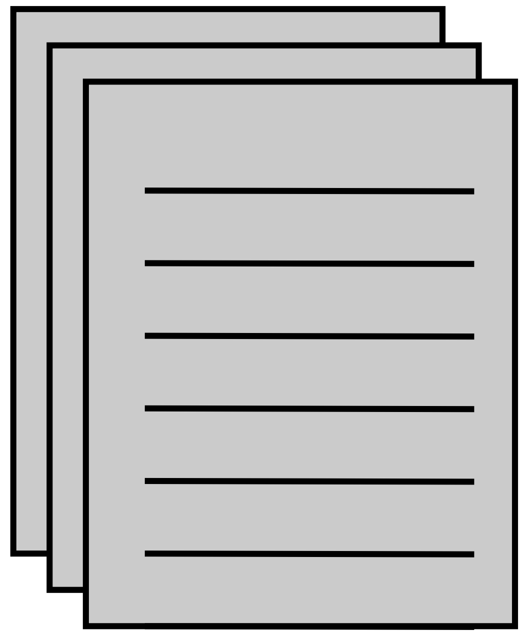
- Evaluate on Penn Treebank: small dataset (1M words) compared to what's used in MT, but common benchmark
- Kneser-Ney 5-gram model with cache: PPL = 125.7
- LSTM: PPL ~ 60-80 (depending on how much you optimize it)
- Melis et al.: many neural LM improvements from 2014-2017 are subsumed by just using the right regularization (right dropout settings). So LSTMs are pretty good

Merity et al. (2017), Melis et al. (2017)

# Phrase-Based MT

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Language model P(e)

Unlabeled English data

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

"Translate faithfully but make fluent English"

# Phrase-Based Decoding

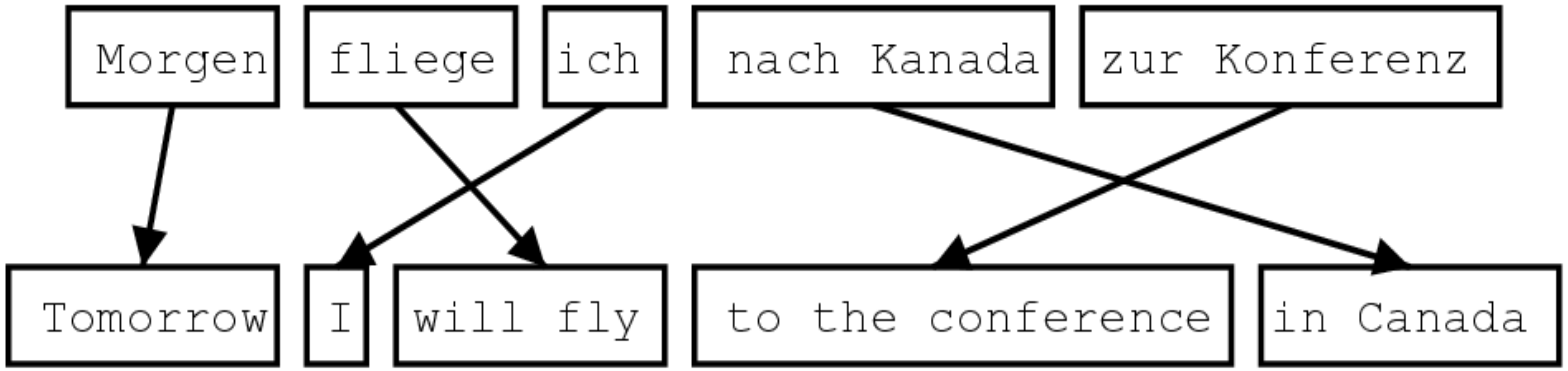
- Inputs:
  - Language model that scores $P(e_i|e_1, \ldots, e_{i-1}) \approx P(e_i|e_{i-n-1}, \ldots, e_{i-1})$
  - Phrase table: set of phrase pairs (**e**, **f**) with probabilities P(**f**|**e**)

# Phrase-Based Decoding

- Inputs:
  - Language model that scores $P(e_i|e_1,\ldots,e_{i-1}) \approx P(e_i|e_{i-n-1},\ldots,e_{i-1})$
  - Phrase table: set of phrase pairs (**e**, **f**) with probabilities P(**f**|**e**)
- What we want to find: **e** produced by a series of phrase-by-phrase translations from an input **f**, possibly with reordering:

# Phrase-Based Decoding

- Inputs:
  - Language model that scores $P(e_i|e_1,\ldots,e_{i-1}) \approx P(e_i|e_{i-n-1},\ldots,e_{i-1})$
  - Phrase table: set of phrase pairs (**e**, **f**) with probabilities P(**f**|**e**)
- What we want to find: **e** produced by a series of phrase-by-phrase translations from an input **f**, possibly with reordering:



# Moses

- Toolkit for machine translation due to Philipp Koehn + Hieu Hoang
  - Pharaoh (Koehn, 2004) is the decoder from Koehn's thesis

# Moses

- Toolkit for machine translation due to Philipp Koehn + Hieu Hoang
  - Pharaoh (Koehn, 2004) is the decoder from Koehn's thesis
- Moses implements word alignment, language models, and this decoder, plus *a ton* more stuff
  - Highly optimized and heavily engineered, could more or less build SOTA translation systems with this from 2007-2013

# Moses

- Toolkit for machine translation due to Philipp Koehn + Hieu Hoang
  - Pharaoh (Koehn, 2004) is the decoder from Koehn's thesis
- Moses implements word alignment, language models, and this decoder, plus *a ton* more stuff
  - Highly optimized and heavily engineered, could more or less build SOTA translation systems with this from 2007-2013
- Next time: results on these and comparisons to neural methods

# Takeaways

- Phrase-based systems consist of 3 pieces: aligner, language model, decoder
  - HMMs work well for alignment
  - N-gram language models are scalable and historically worked well
  - Decoder requires searching through a complex state space
- Lots of system variants incorporating syntax
- Next time: neural MT