Language in 10 minutes

- http://mt-class.org/jhu/lin10.html
- **By Friday**: Group up (optional, max size 2), choose a language (not one y'all speak) and a date
- First presentation: Yuan on Thursday
- Yuan will start assigning groups and people who miss the deadline

Brief review

$$p(e \mid f) = \underset{e}{\text{argmax}}\ p(f \mid e)p(e)$$

- Last week: language modeling: p(e)
- This week: translation modeling: p(f | e)
    - In particular, *alignment*

Brief review: language modeling

- Modeling P(e)
- Dumb idea: maintain a huge table of complete sentences and relative frequencies
- Better idea: **n-grams**
  - Chain rule and conditional independence for an n–1 word history
- Problems remain, but they work pretty well
- Not discussed: **smoothing**

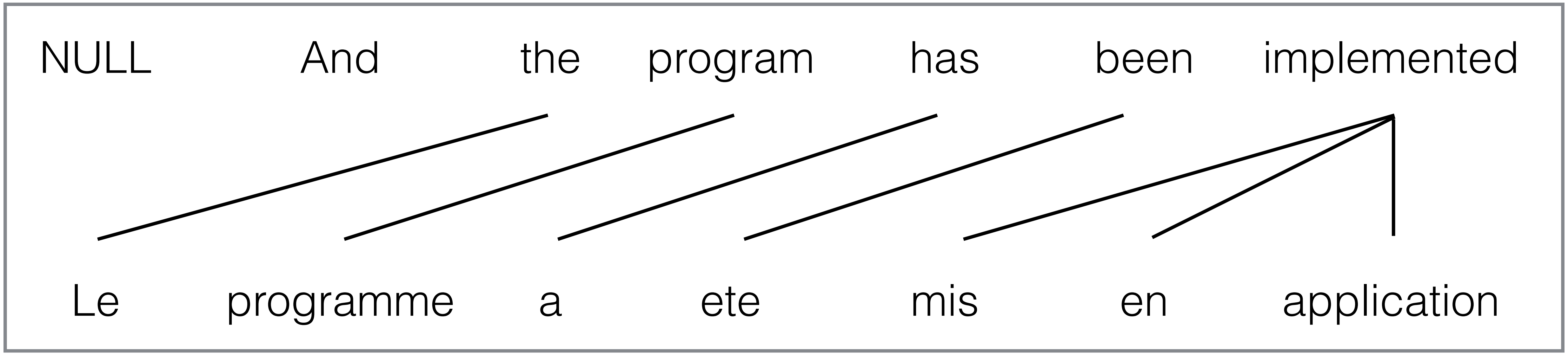
Alignments

$$\boxed{p(e \mid f) = \underset{e}{\textsf{argmax}}\ p(f \mid e)p(e)}$$

- How to model p(f | e)?
- What's the dumb idea?

Alignments

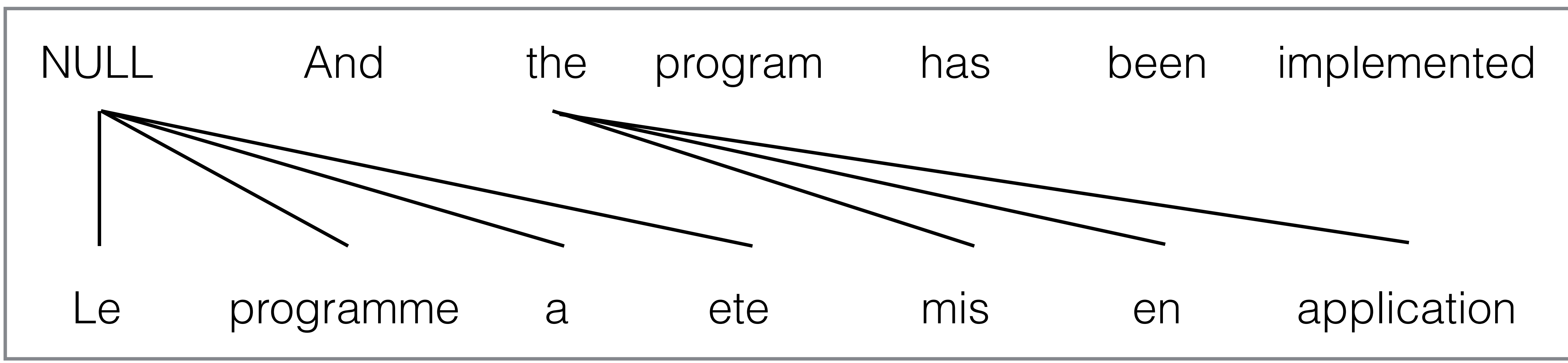
- Each French word *f* is generated by exactly one English word *e*



- Alignment vector a = $\langle 2,3,4,5,6,6,6 \rangle$

Alignments

- Each French word *f* is generated by exactly one English word *e*



- Alignment vector a = $\langle 0,0,0,0,2,2,2 \rangle$

Alignments

- How many possible alignments?
  - Each of m French word has $l=|E|+1$ choices, so m^{l+1}

A bit more formally

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

- Define a *conditional model* projecting the translations through the alignments

- We also introduce a *conditional independence* assumption: every word is translated independently

Brainstorm

5 minutes, with a neighbor or two

- Is this idea a good one?
- What are some of its limitations?
- What else should be modeled?

IBM Alignment Models

- Proposed by IBM researchers (under CLSP's Fred Jelinek) in the late 80s / early 90s
- Aside: "Rip Van Winkle" event
  - Transcript at cs.jhu.edu/~post/bitext

IBM Model 1

- Input: English words, $e_1 \ldots e_l$, French length m
- For each French word position $i \in 1 \ldots m$
  - Choose an English source index $q(j \mid i, l, m) = \frac{1}{l+1}$
  - Choose a translation $t(f_i \mid e_{a_i})$

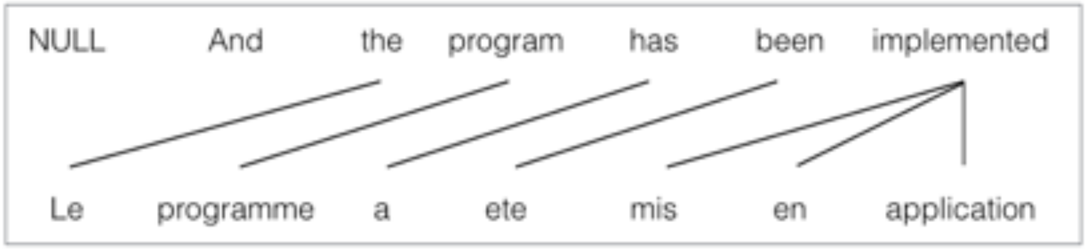
IBM Model 1

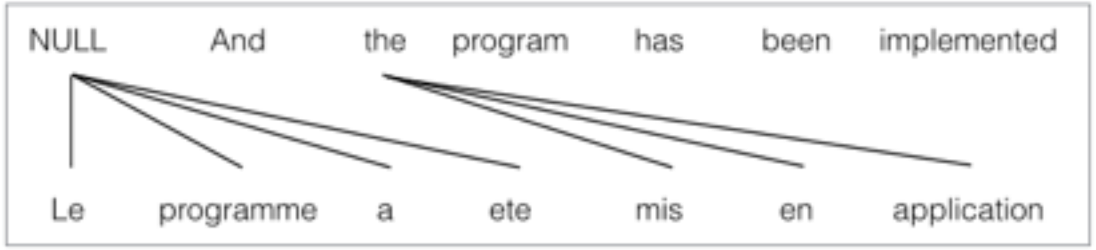
- $t(f \mid e)$ is just a table

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

IBM Model 1

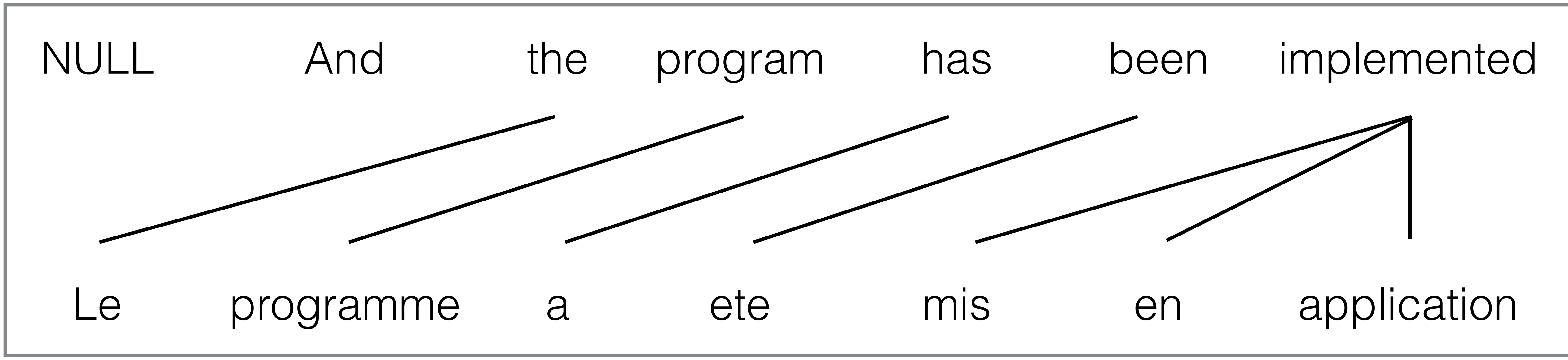
- Important notes
  - Alignment is based on word *positions*, not word *identities*
  - Alignment probabilities are *uniform*





  - Words are translated independently

IBM Model 1



- On board: $p(f, a \mid e) = ?$

IBM Model 2

- Input: English words, $e_1 \ldots e_l$, French length m
- For each French word position $i \in 1..m$
  - Choose an English source index $q(j \mid i, l, m)$
  - Choose a translation $t(f_i \mid e_{a_i})$

IBM Model 2

- Only difference: $q(j \mid i, l, m)$ is now a table instead of uniform
- What do you think of this model?
- How many parameters are there?

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

Tasks

- The models tell us how (we pretend) the data came to be

- There are now two tasks we care about

  - *Inference:* given a sentence pair (e,f), what is the most probable alignment?

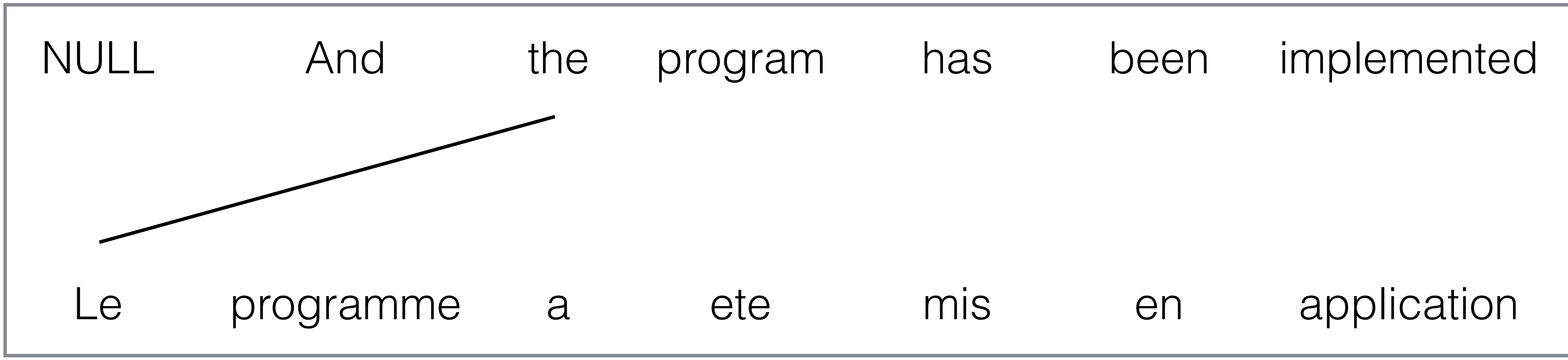
  - *Estimation:* how do we get the parameters $t(f \mid e)$ and $q(j \mid i, l, m)$?

Task 1: Inference

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

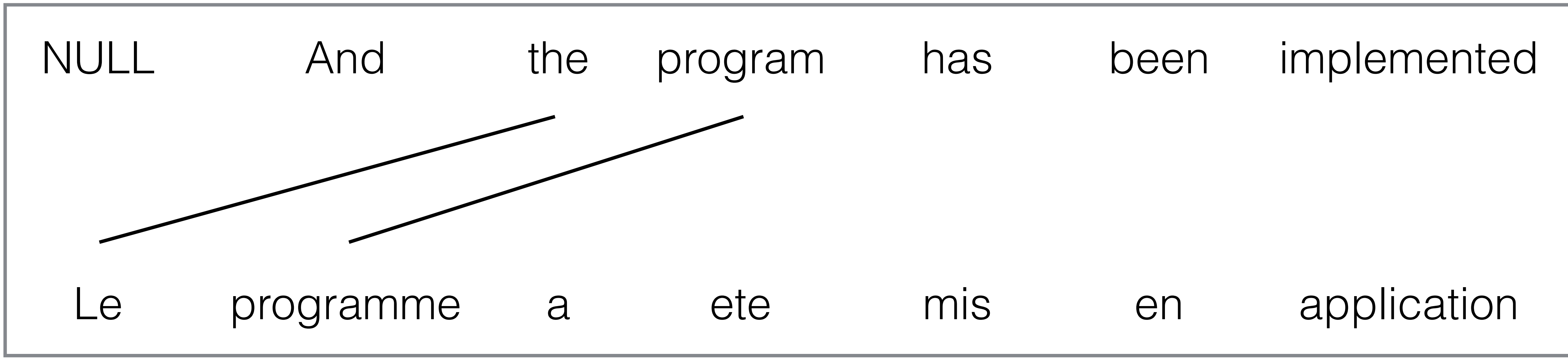
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



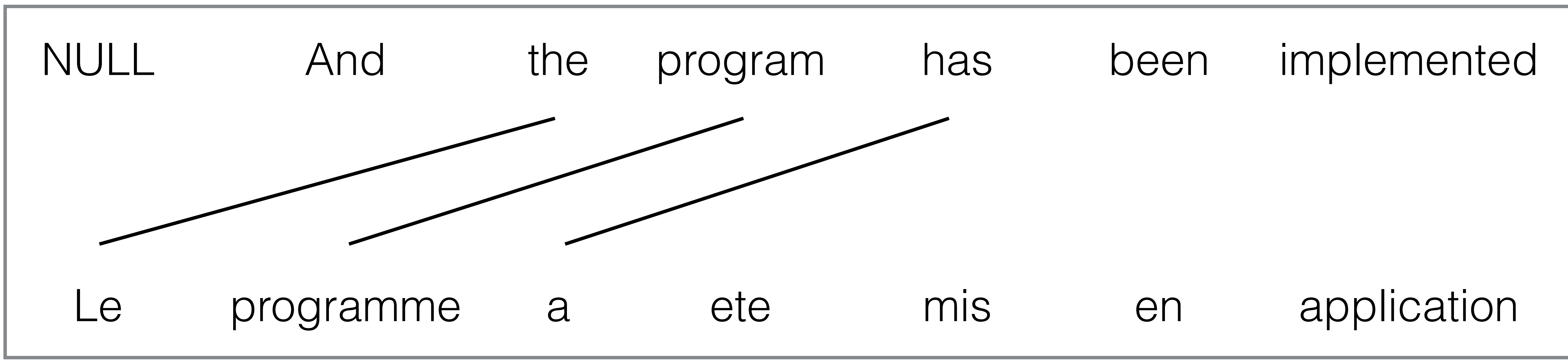
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



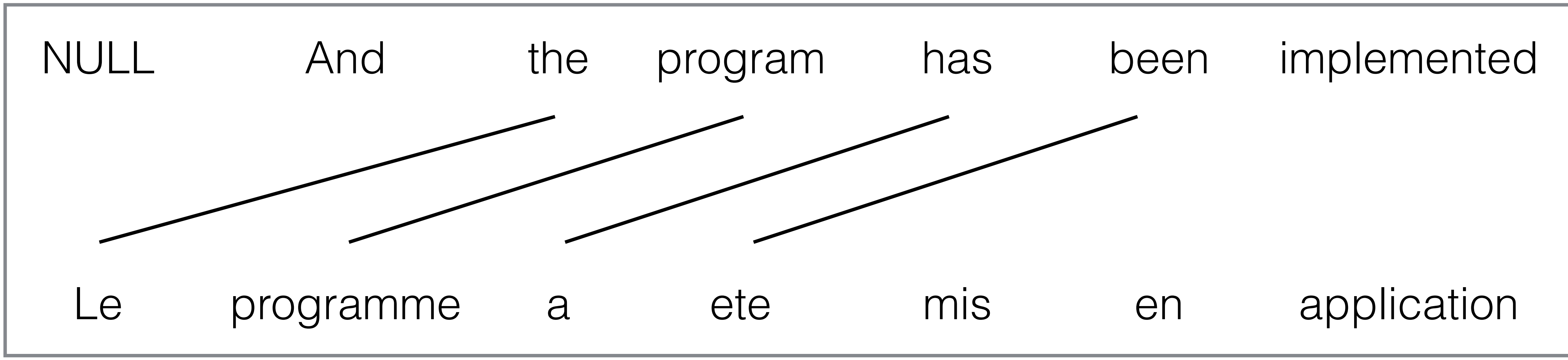
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



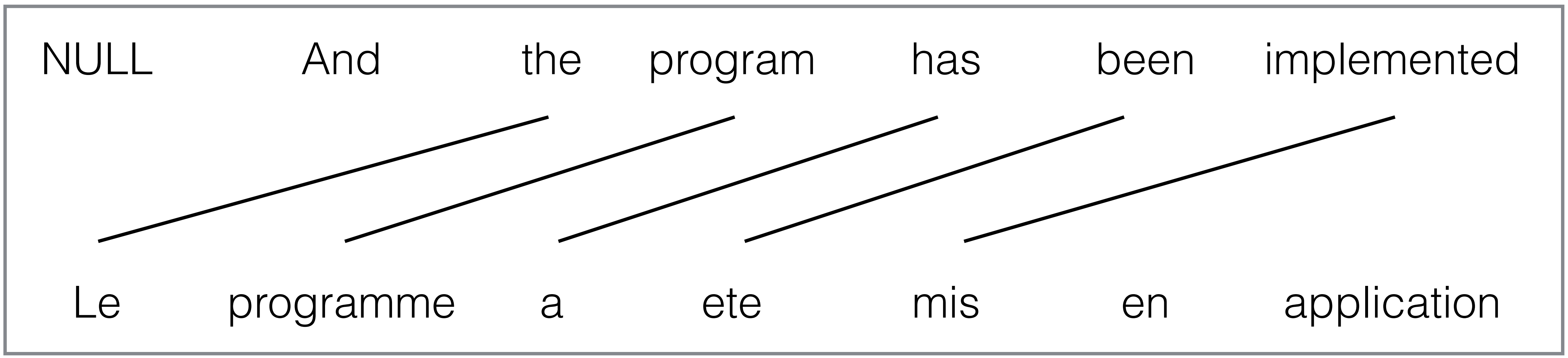
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



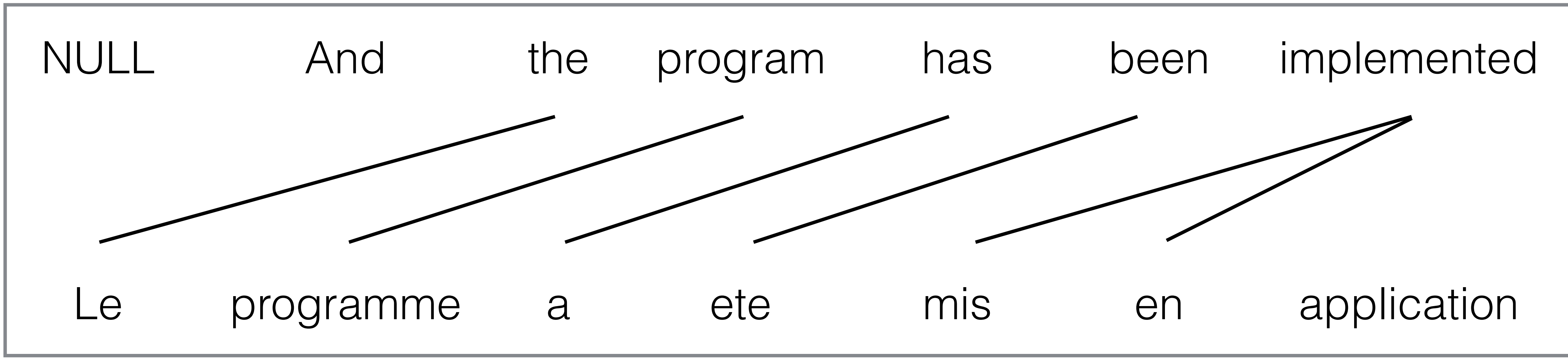
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



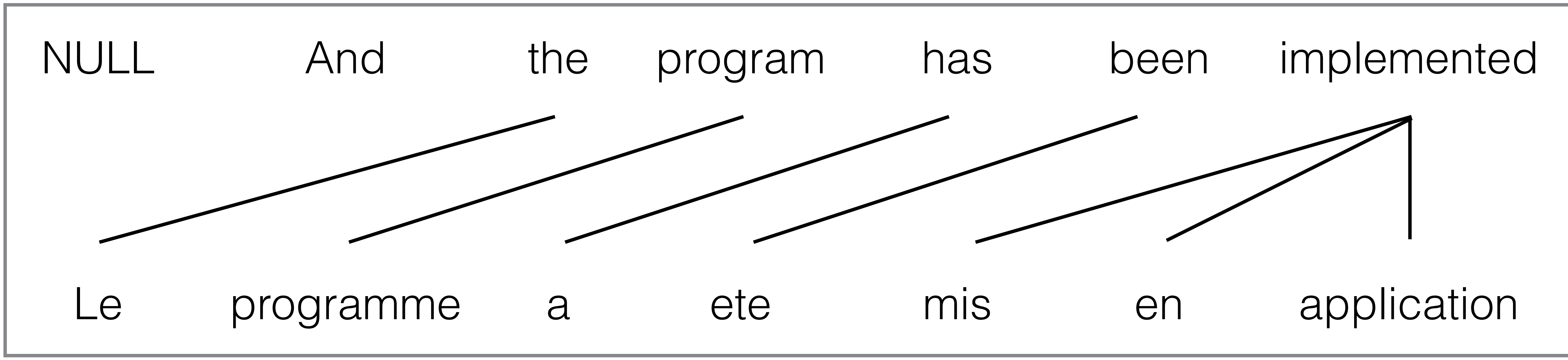
- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Task 1: Inference



- Input: a sentence pair (e,f), the model (*t(•)* and *q(•)*)
- Knowledge: target words generated independently

Homework 1

- The *inference* task is what you're doing in Homework 1

- The metric is Alignment Error Rate (AER)

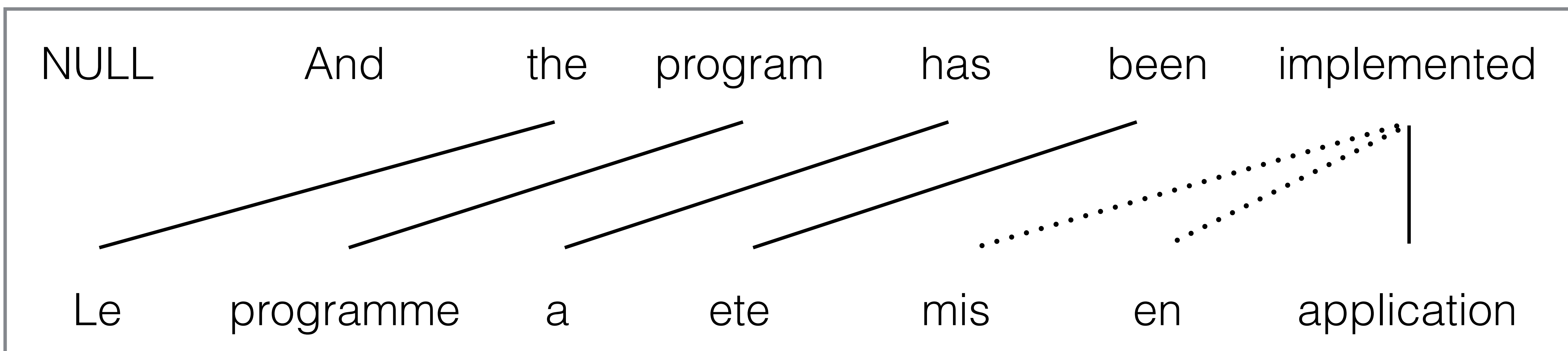
  - Alignment links are labeled as one of (S)ure or (P)ossible, S ⊆ P

  - Precision: $\frac{|A \cap P|}{|P|}$ Recall: $\frac{|A \cap S|}{|S|}$

Homework 1

- Precision: $\dfrac{|A \cap P|}{|P|}$ Recall: $\dfrac{|A \cap S|}{|S|}$

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



Task 2: Parameter Estimation

- Computing alignments is useful as an intermediate test of new alignment methods, but we actually don't care about the alignments themselves
- What we really need is to compute the parameters of the model: $t(f \mid e)$ and $q(j \mid i, l, m)$

Estimation

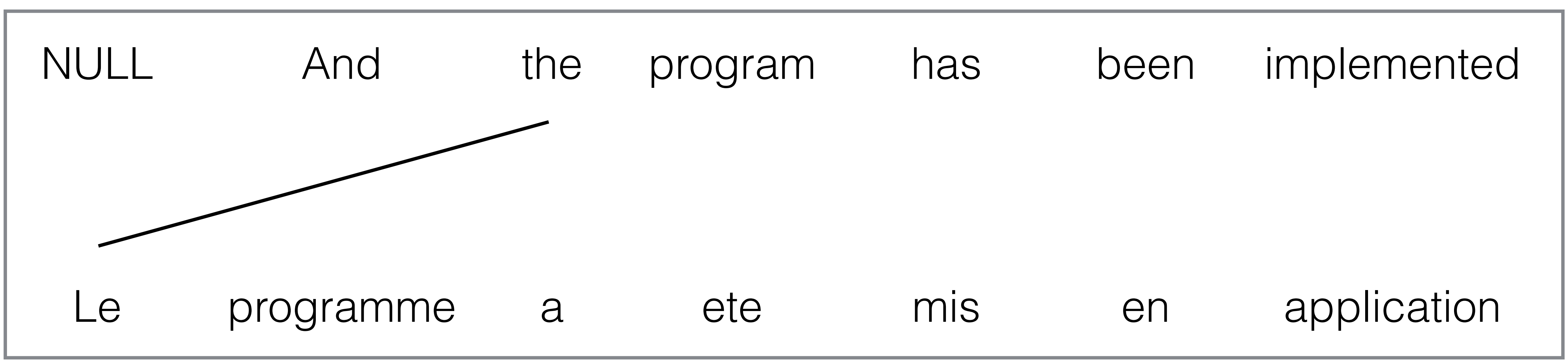
- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?

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

Estimation

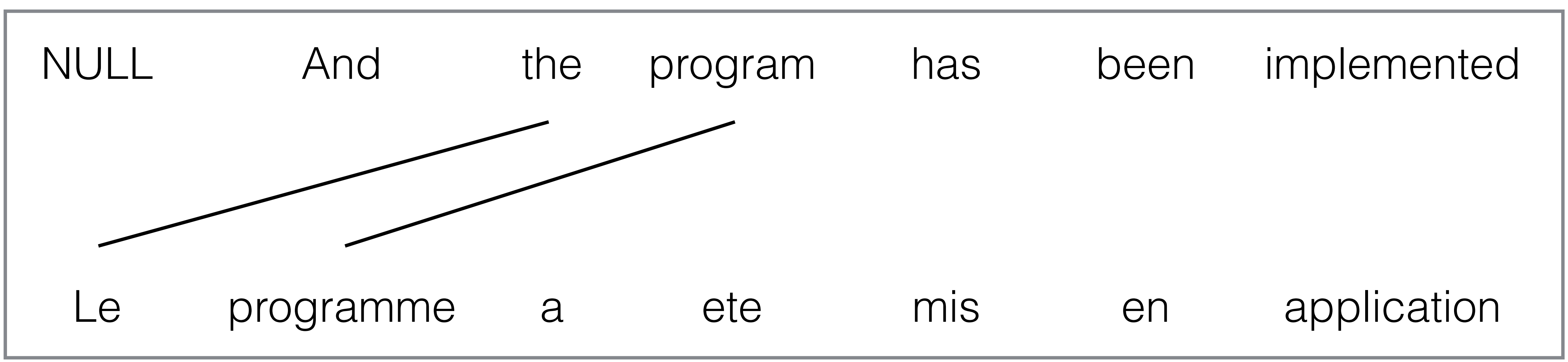
- Easy! Just like n-grams: count and normalize (forget smoothing)

- Board exercise: what are the equations and values if this were our corpus?



Estimation

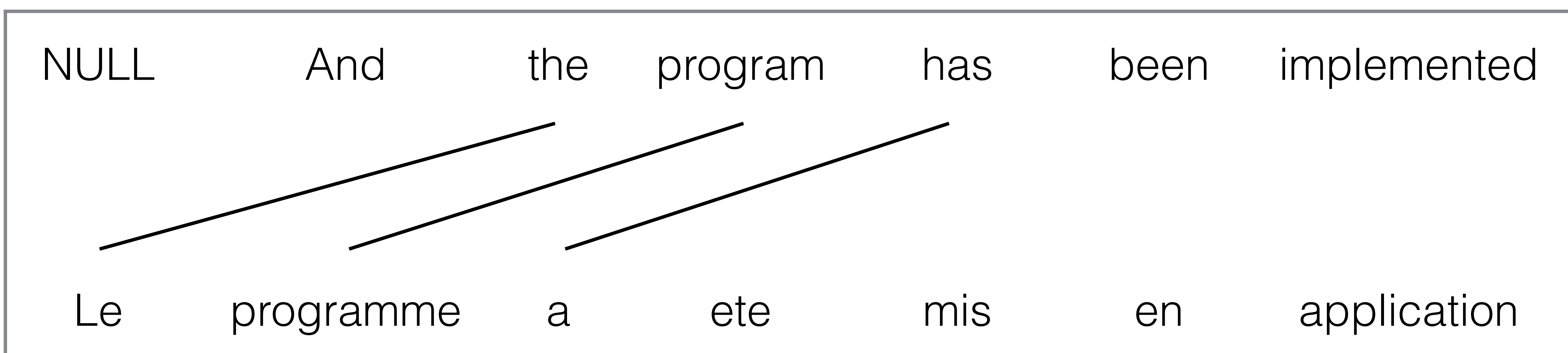
- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?



Estimation

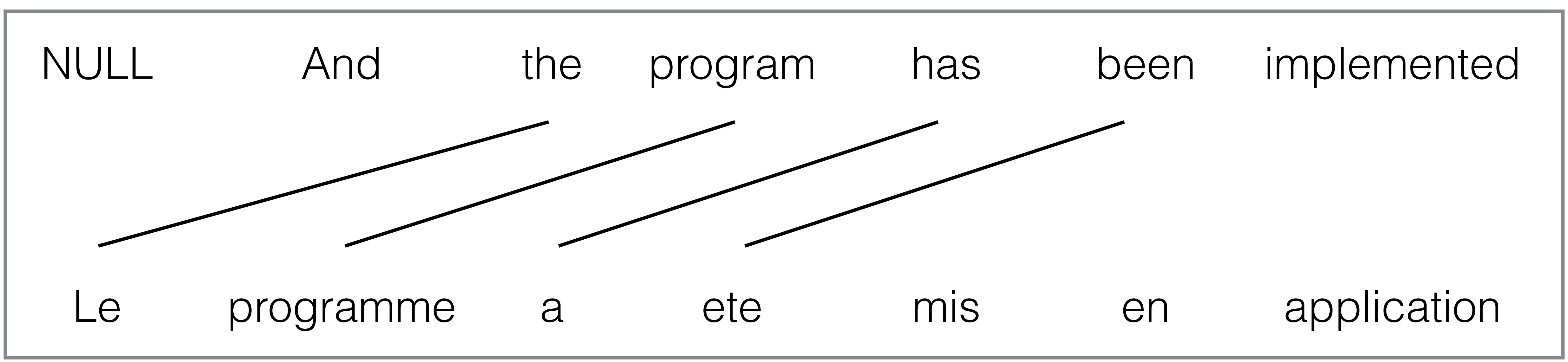
- Easy! Just like n-grams: count and normalize (forget smoothing)

- Board exercise: what are the equations and values if this were our corpus?



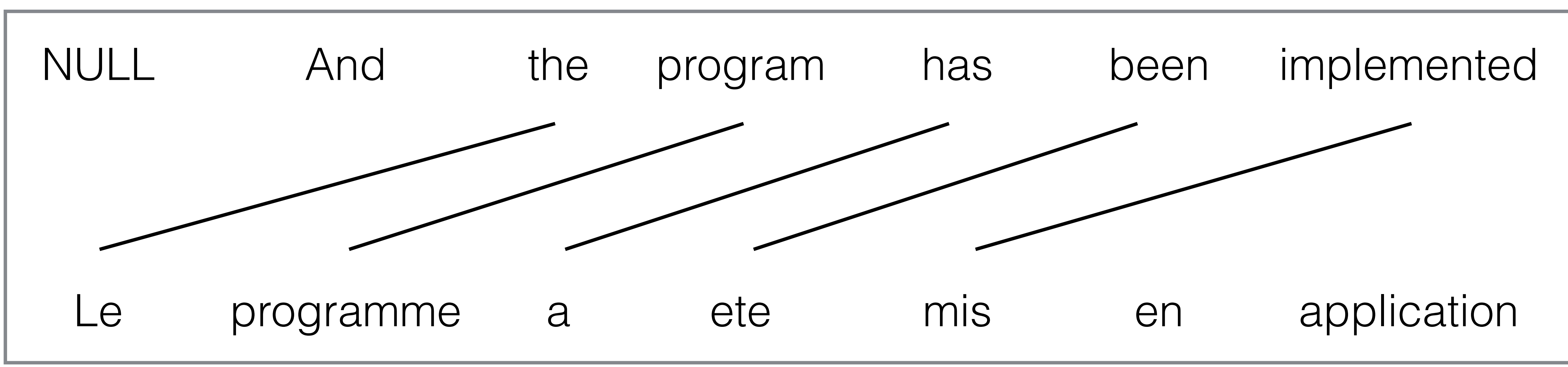
Estimation

- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?



Estimation

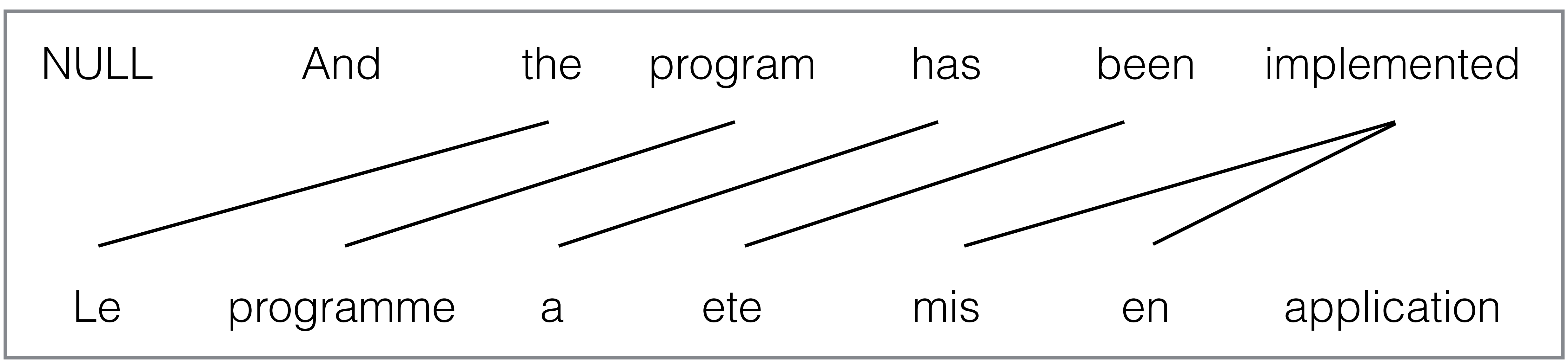
- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?



Estimation

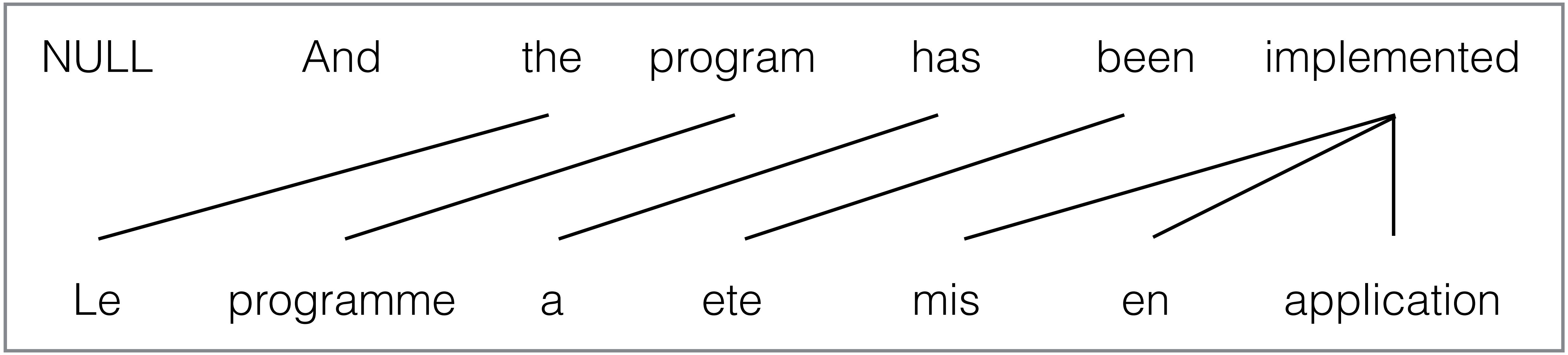
- Easy! Just like n-grams: count and normalize (forget smoothing)

- Board exercise: what are the equations and values if this were our corpus?



Estimation

- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?



Estimation from (e,f,a)

- Easy! Just like n-grams: count and normalize (forget smoothing)
- Board exercise: what are the equations and values if this were our corpus?

$$t(f \mid e) = \frac{c(e, f)}{c(e)}$$

$$q(j \mid i, l, m) = \frac{c(j, i, l, m)}{c(i, l, m)}$$

Estimation from (e,f)

- Unfortunately, we don't have alignments!
- (Even more unfortunately, alignments are a fuzzy concept)
- Chicken and egg problem
    - If we had the alignments, we could compute parameters
    - If we had the parameters, we could compute the alignments (how?)

Estimation from (e,f)

- This suggests an iterative solution:

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)
```

Estimation

- A few problems
  - We don’t actually care about the alignments
  - Bad init. might set us off in the wrong direction
- A “softer” approach: compute expectations over *all* alignments
- Weight the accumulated counts by the alignment probability

Estimation

- Each alignment link has a weight

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

- Counts now use this “soft” value instead of a hard count (1 or 0)

- Any issues here?

Estimation from (e,f)

- Old solution

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)
```

Estimation from (e,f)

- New solution

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
                count(f_j, e_i) += P(a_i = j | e_i, f_j)
                count(e_i) += P(a_i = j | e_i, f_j)
                count(j, i, l, m) += P(a_i = j | e_i, f_j)
                count(i, l, m) += P(a_i = j | e_i, f_j)
    t(f | e) = count(f, e) / count(e)
    q(j | i, l, m) = count(j, i, l, m) / count(i, l, m)
```

Estimation with EM

- Why does this work?
  - We are accumulating evidence (soft counts) for totally bogus alignments: all pairs of words that co-occur, e.g., t(*streetcar* | *le*)
- It works for the same reason you were able to solve the alignment exercise from the first day of class
  - Words that co-occur frequently continually steal probability mass from pairs that co-occur less often

Properties of EM

- The EM algorithm guarantees that data likelihood *does not decrease* across iterations

$$\log \mathcal{L}(t, q \mid E, F) = \log \prod_{n=1}^{N} \sum p(f^{(n)} \mid e^{(n)})$$

$$= \sum_{n=1}^{N} \log \sum_{a \in A} p(f^{(n)}, a \mid e^{(n)})$$

- EM can get stuck in *local optima*: subprime peaks in the global likelihood function

Assorted notes

- There are many known problems with these alignment models (garbage collection, initialization)
- Despite all this blabbing about modeling p(f | e), the IBM models are not actually used for translation!
- Who cares about alignment?

Thursday’s Agenda

- Read: Collins’ notes on Models 1 and 2, Koehn Chapter 4, Knight’s MT workbook
- We’ll cover new models
    - IBM Model 3, HMM model
- Time for questions on Homework 1 (due Feb. 17)
- Language in 10 minutes (Yuan)