Breaking down the Language Barrier with Statistical Machine Translation:
3) Phrase-based MT

http://www.phontron.com/class/sentan2014

Advanced Research Seminar I/III

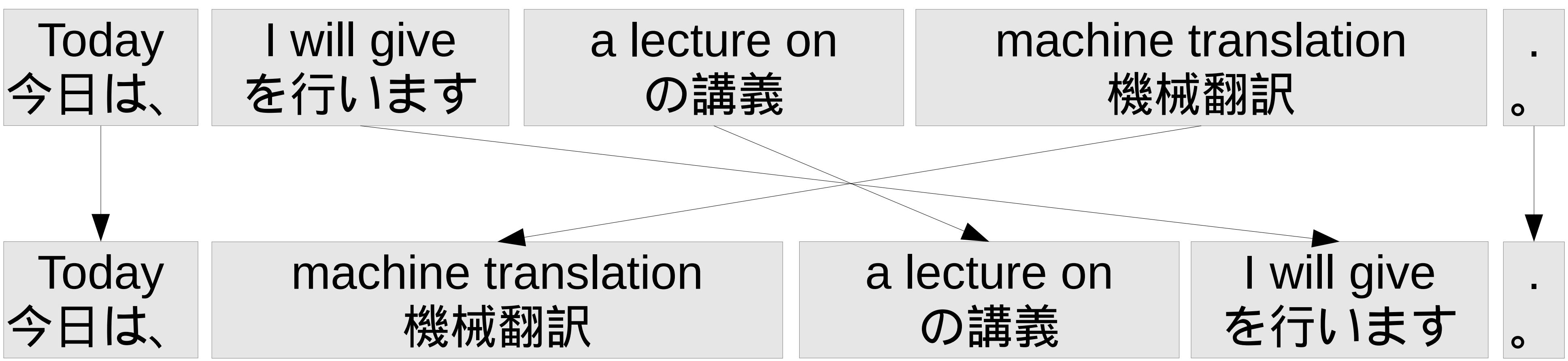
Graham Neubig

2014-2-04

How does machine translation work?

- Divide sentence into translatable patterns, reorder, combine

Today I will give a lecture on machine translation .



今日は、機械翻訳の講義を行います。

Assignment

- (Only one assignment this week)
- You are given a baseline machine translation system
  - LM/Alignment: Baseline from exercises 1, 2
  - TM: Phrases of up to length 4
  - SM: Uniform distribution
  - RM: Distortion penalty
  - Reordering Limit: 6
- Try to improve its accuracy by changing one of the features listed above, or anything else

Probabilistic Model for Translation

Formal Definition of Translation

- A translation is defined as (in opposite order)
  - Output sentence E
  - Derivation D
  - Input sentence F

E = hello where is the station

D =

D_{ep} =	hello	where is	the station
D_{fp} =	こんにちは	どこ です か	駅 は
D_o =	0	2	1

F = こんにちは 駅 は どこ です か

Finding the Best Translation

- We define the "best" translation as the one with the highest posterior probability of E given F

$$\hat{E} = \underset{E}{\operatorname{argmax}}\, P(E \mid F)$$

- We can calculate this by summing over D

$$\hat{E} = \underset{E}{\operatorname{argmax}} \sum_{D} P(D, E \mid F)$$

- But this is inefficient, so approximate using the max

$$\hat{E} \approx \underset{E}{\operatorname{argmax}}\, P(D, E \mid F)$$

Probabilistic Modeling of Translation

- We want a probability of D and E given F: $P(D,E|F)$
- Use Bayes's law and note that P(F) doesn't affect results

$$P(D,E|F)=P(D,E,F)/P(F)$$
$$\propto P(D,E,F)$$

- And split the probabilities further

$$\begin{aligned}P(D,E,F)\propto\ & P(E)* && \text{Language Model}\\ & P(D_{ep}|E)* && \text{Segmentation Model}\\ & P(D_{fp}|D_{ep},E)* && \text{Translation Model}\\ & P(D_{order}|D_{fp},D_{ep},E)* && \text{Reordering Model}\\ & P(F|D_{order},D_{fp},D_{ep},E) && \text{Always P=1 (F is decided by D)}\end{aligned}$$

Language Model

- Calculate the probability of the output words using *n*-gram

$$E = \{e_1, \ldots, e_I\}$$

$$P(E) = \prod_{i=1}^{I+1} P(e_i | e_{i-N+1}, \ldots, e_{i-1})$$

e.g. bigram

E = hello where is the station

P(E) = P(hello|<s>) * P(where|hello) * P(is|where)
* P(the|is) * P(station|the) * P(</s>|station)

Segmentation Model

- Measures the probability of dividing E into segments

$$D_{ep} = \{ep_1, \ldots, ep_K\}$$

$P(D_{ep} \mid E)$	E =	hello where is the station	
	D_{ep} =	hello	where is	the station

- This is less important than other models
- Often just use uniform probability

$$P(D_{ep} \mid E) = 1 / Z_{ep}$$

- Sometimes use proportional to number of phrases

$$P(D_{ep} \mid E) = e^{-\alpha_{ep} K} / Z_{ep}$$

(fewer phrases → longer, more reliable phrases)

Translation Model

- Probability of translating phrases $D_{ep} \rightarrow D_{fp}$
  - Because $P(E|D_{ep}) = 1$, we can remove E

$$P(D_{fp} | D_{ep}, E) = P(D_{fp} | D_{ep})$$

- We often assume that the translation probability of phrases is independent

$$P(D_{fp} | D_{ep}) = \prod_{k=1}^{K} P(fp_k | ep_k)$$

D_{ep} = hello | where is | the station

P(こんにちは |hello) * P(どこ です か |where is) * P(駅 は |the station)

D_{fp} = こんにちは | どこ です か | 駅 は

Reordering Model

- Probability of choosing a particular ordering

$$P(D_o | D_{fp}, D_{ep}, E) = P(D_o | D_{fp}, D_{ep})$$

D_{ep} =	hello	where is	the station
D_{fp} =	こんにちは	どこ です か	駅 は
D_o =	0	2	1

Reordering Model: Distortion Penalty (1)

- Think about no reordering:

D_{ep} = hello | the station | where is

D_{fp} = <s> | こんにちは | 駅 は | どこ です か | </s>

0 1 2 3 4 5 6 7

+1 +1 +1 +1

$\text{first}(fp_k) - \text{last}(fp_{k-1}) == +1$

- With reordering:

D_{ep} = hello | where is | the station

D_{fp} = <s> | こんにちは | どこ です か | 駅 は | </s>

0 1 4 5 6 2 3 7

+1 +3 -4 +4

+3 -4 +4

$\text{dist}(fp_{k-1}, fp_k) = |\text{first}(fp_k) - \text{last}(fp_{k-1}) - 1|$

- Distortion is distance from +1:

Reordering Model: Distortion Penalty (2)

- Distortion is the distance from +1:

$$\text{dist}(fp_{k-1}, fp_k) = |\text{first}(fp_k) - \text{last}(fp_{k-1}) - 1|$$

D_{ep} =		hello	where is	the station	
D_{fp} =	<s>	こんにちは	どこ です か	駅 は	</s>
	0	1	4 5 6	2 3	7
		+1	+3	-4	+4
dist=		0	2	5	3

- We put an exponential penalty on distortion:

$$P(D_o | D_{fp}, D_{ep}) = \frac{\prod_{k=1}^{K+1} e^{-\alpha_o \text{dist}(fp_{k-1}, fp_k)}}{Z_o}$$

Reordering Model: Lexicalized Reordering Probability

- Each phrase has a reordering type

D_{ep} =		hello	where is	the station	
D_{fp} =	<s>	こんにちは	どこ です か	駅 は	</s>
	0	1	4 5 6	2 3	7
type(fp_{k-1}, fp_k)=	mono	disc.	swap	disc.	

last(fp_{k-1}) + 1 = first(fp_k) → mono

last(fp_k) + 1 = first(fp_{k-1}) → swap

otherwise → disc

- Calculate probability from training data and multiply

$$P(D_o|D_{fp}, D_{ep}) = \frac{\prod_{k=1}^{K+1} P(\text{type}(fp_{k-1}, fp_k)|fp_{k-1}, ep_{k-1})}{Z_o}$$

Putting Everything Together

$$P(D,E,F) \propto P(E) *$$ Language Model

$$P(D_{ep} | E) *$$ Segmentation Model

$$P(D_{fp} | D_{ep}, E) *$$ Translation Model

$$P(D_{order} | D_{fp}, D_{ep}, E)$$ Reordering Model

D =

D_{ep} =	hello	where is	the station
D_{fp} =	こんにちは	どこ です か	駅 は
D_o =	0	2	1

LM (bigram) = P(hello|<s>) * P(where|hello) * P(is|where) * P(the|is) * P(station|the) * P(</s>|station)

SM (expon) = $e^{-\alpha_{ep}}$ * $e^{-\alpha_{ep}}$ * $e^{-\alpha_{ep}}$

TM =P(こんにちは |hello) * P(どこ です か |where is) * P(駅 は |the station)

RM (distort) = $e^{-\alpha_o * 0}$ $e^{-\alpha_o * 2}$ $e^{-\alpha_o * 5}$ $e^{-\alpha_o * 3}$

Log Probabilities

$$\log P(D,E,F) \propto \log P(E) + \quad \text{Language Model}$$
$$\log P(D_{ep}|E) + \quad \text{Segmentation Model}$$
$$\log P(D_{fp}|D_{ep},E) + \quad \text{Translation Model}$$
$$\log P(D_{order}|D_{fp},D_{ep},E) \quad \text{Reordering Model}$$

D =

D_{ep} =	hello	where is	the station
D_{fp} =	こんにちは	どこ です か	駅 は
D_o =	0	2	1

LM (bigram) = log P(hello|<s>) + log P(where|hello) + log P(is|where) + log P(the|is) + log P(station|the) + log P(</s>|station)

SM (expon) = $-\alpha_{ep} + -\alpha_{ep} + -\alpha_{ep}$

TM =log P(こんにちは |hello) + log P(どこ です か |where is) + log P(駅 は |the station)

RM (distort) = $-\alpha_o * 0 + -\alpha_o * 2 + -\alpha_o * 5 + -\alpha_o * 3$

Search for Machine Translation

Search for Machine Translation

- We want to find the best scoring hypothesis

$$\hat{E} = \underset{E}{\operatorname{argmax}}\, P(D, E \mid F)$$

- Problem: Millions of possible hypotheses for one sentence!
- Solution: Efficient dynamic programming and approximate search algorithms.

Starting Simple

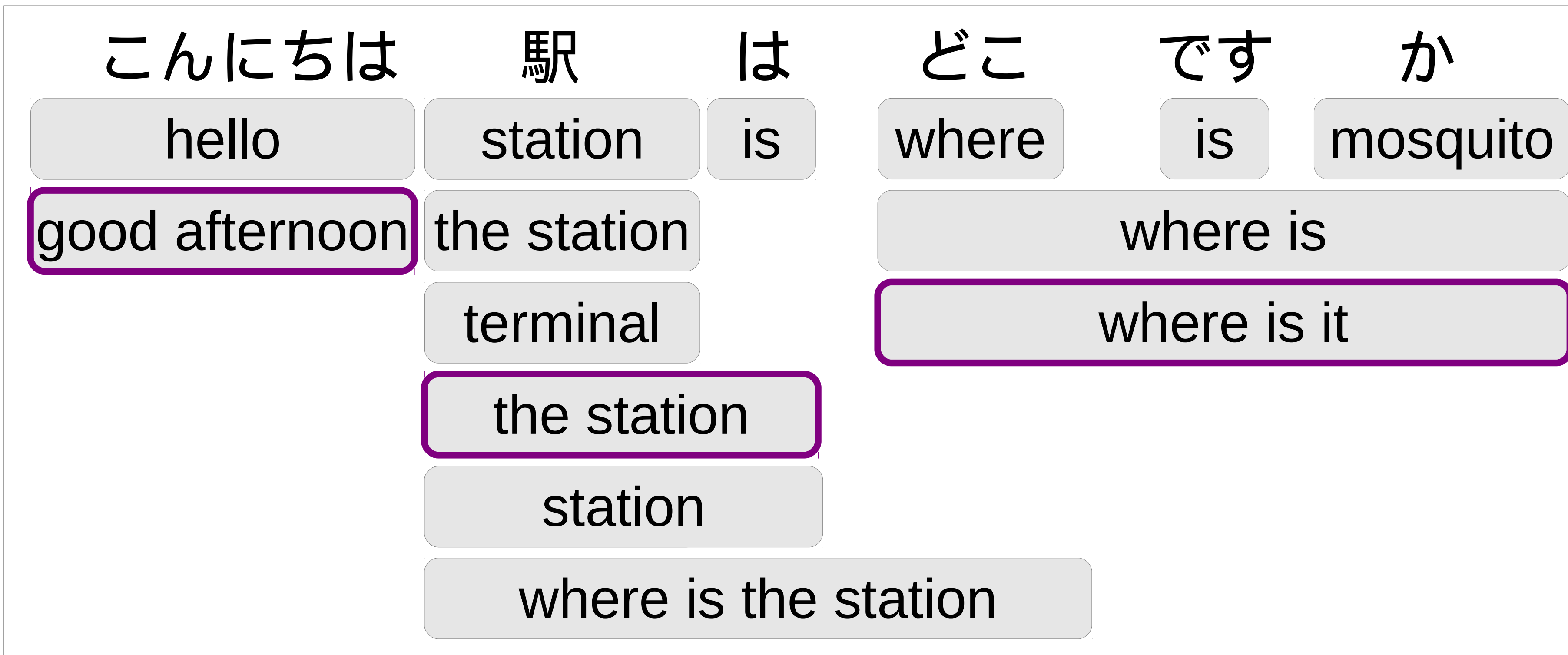
- What do we do when we only have the translation model probability?

$$P(D, E, F) \approx P(D_{fp} | D_{ep})$$

(No reordering for now)

Simple Search

- Given an input
- Expand all of the possible translations



- Find the set of translations maximizing

$$P(D_{fp}|D_{ep}) = \prod_{k=1}^{K} P(fp_k|ep_k)$$

but how?

This Man Has an Answer!

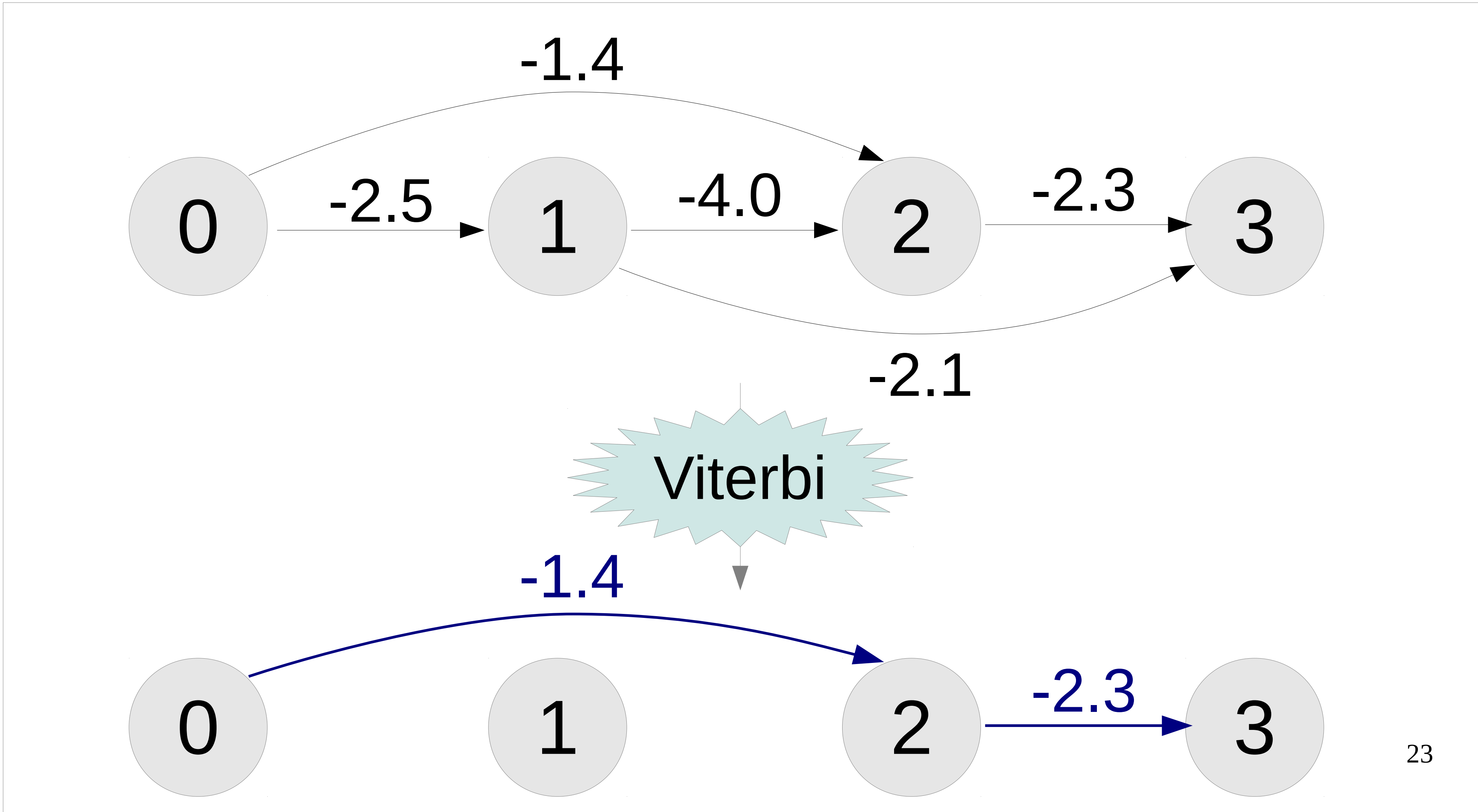
Andrew Viterbi

(Professor UCLA →Founder of Qualcomm)

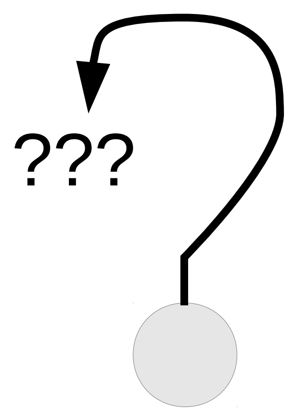
Viterbi Algorithm

The Viterbi Algorithm

- Efficient way to find the highest scoring path in a graph

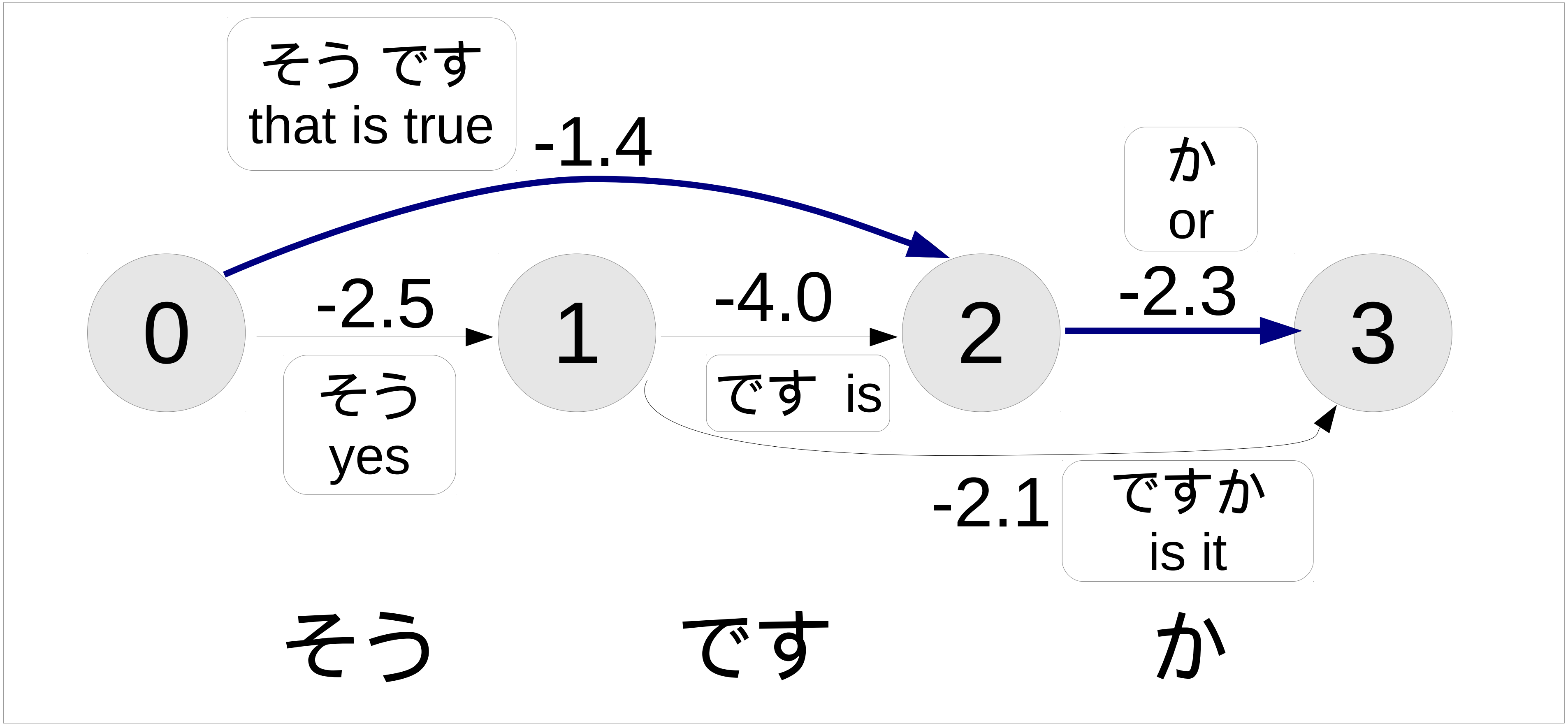


Graph?! What?!



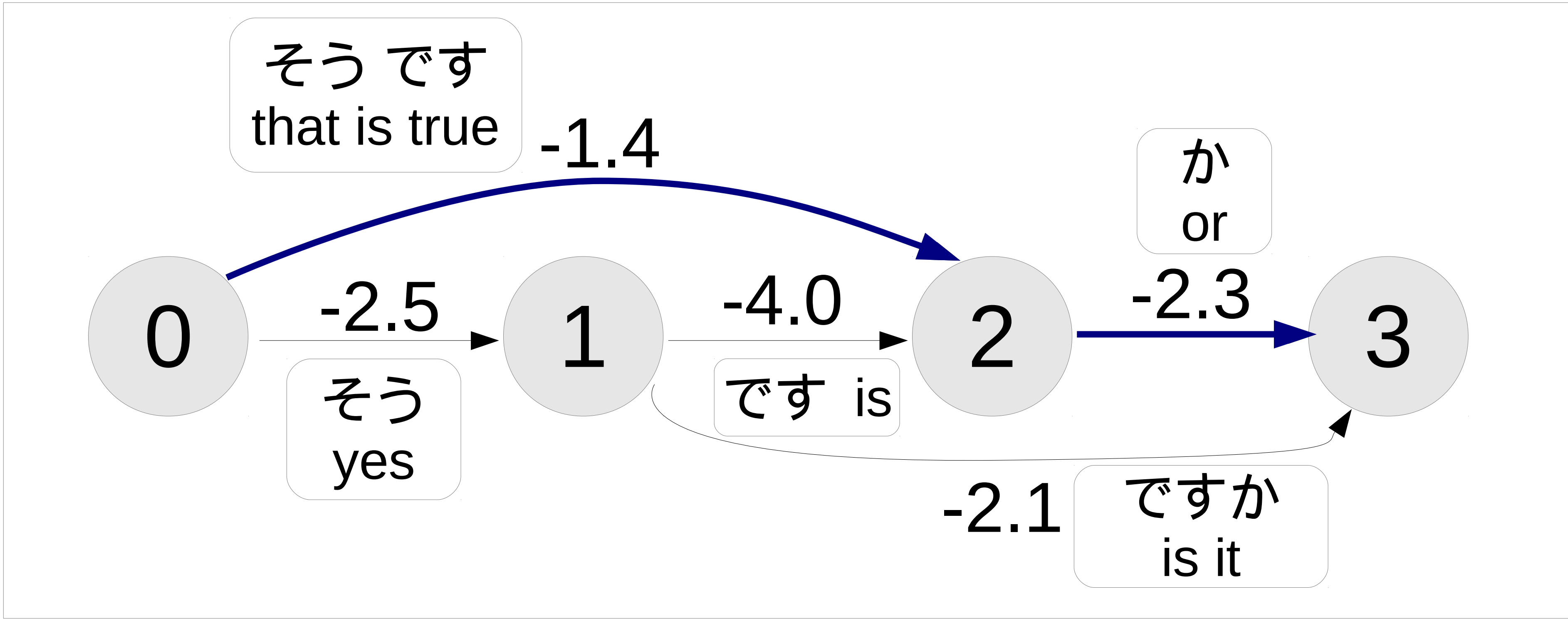
(Let Me Explain!)

Translations as Graphs



- Each node is a position in the sentence
- Each edge is a phrase
- Each path is a full translation

Graph Weights



- Each edge has a weight equal to $\log P(fp_k | ep_k)$
- Each path has a weight equal to sum of edges

$$\log P(D_{fp} | D_{ep}) = \sum_{k=1}^{K} \log P(fp_k | ep_k)$$

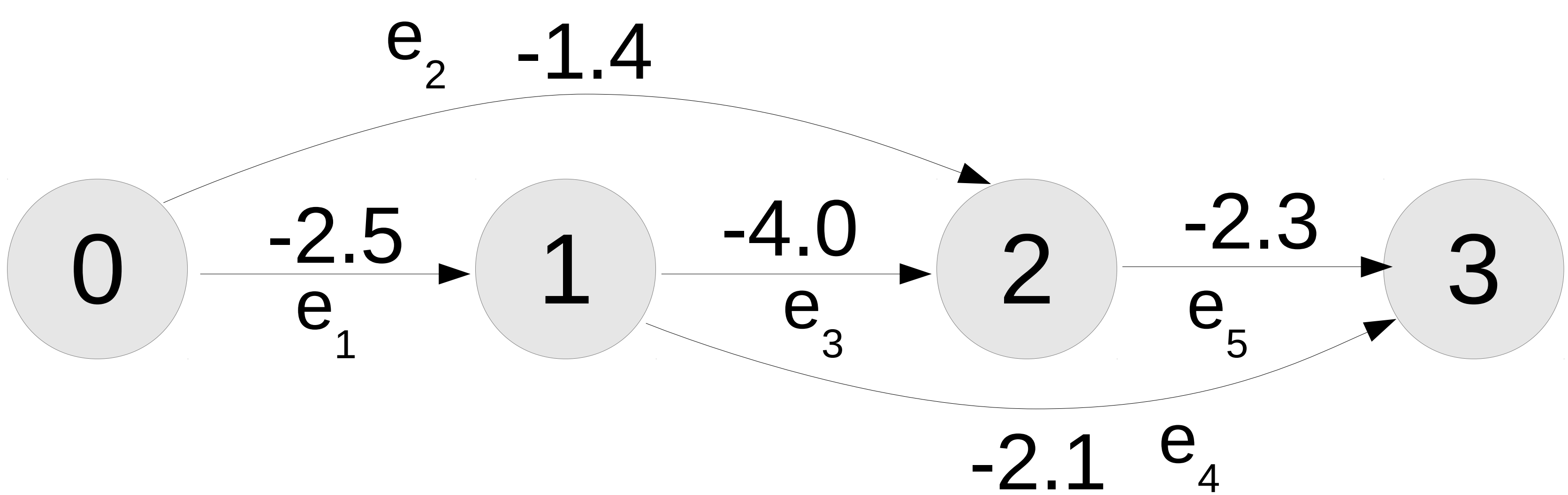
- Highest scoring path is best translation!

Ok Viterbi, Tell Me More!

- The Viterbi Algorithm has two steps
  - In forward order, find the score of the best path to each node
  - In backward order, create the best path

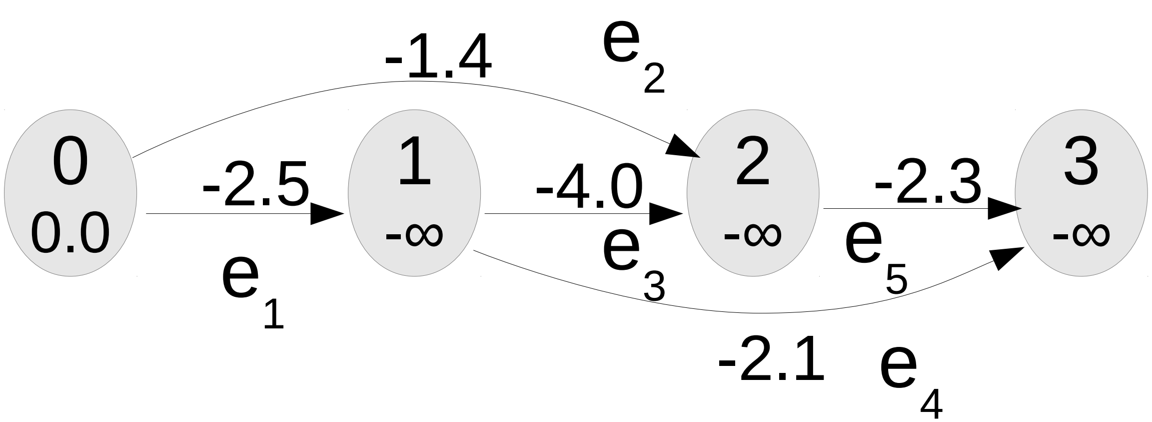
Forward Step

Forward Step



best_score[0] = 0
for each *node* in the *graph* (ascending order)
    best_score[*node*] = -∞
    for each incoming *edge* of *node*
        score = *best_score*[*edge.prev_node*] + *edge.score*
        if *score* > *best_score*[*node*]
            best_score[*node*] = *score*
            best_edge[*node*] = *edge*

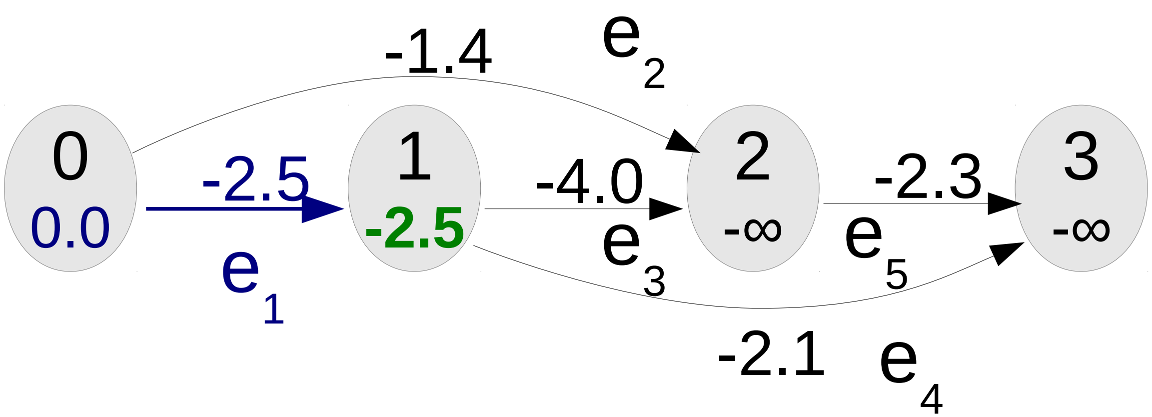
Example:



Initialize:

best_score[0] = 0

Example:



Initialize:

best_score[0] = 0

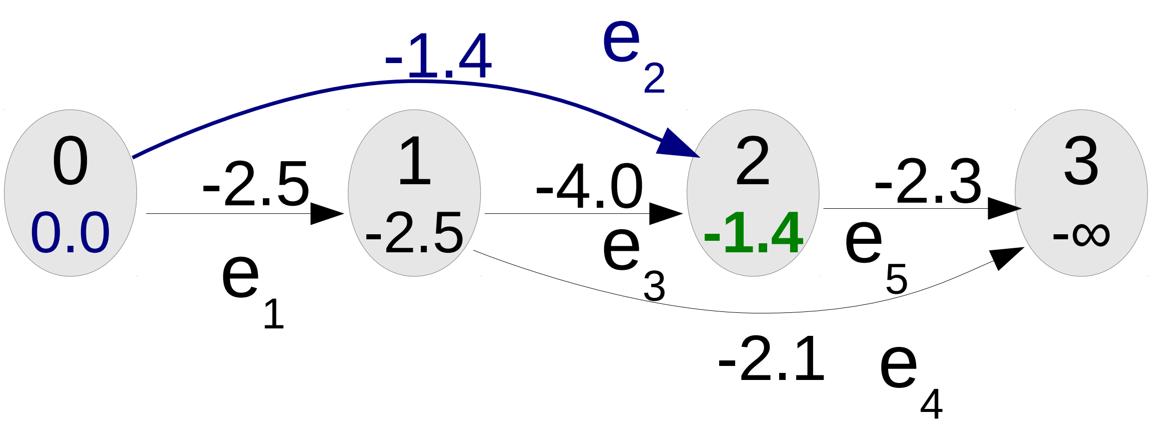
Check e_1:

score = 0 + -2.5 = -2.5 (> -∞)

best_score[1] = -2.5

best_edge[1] = e_1

Example:



<u>Initialize:</u>

best_score[0] = 0

<u>Check e_1:</u>

score = 0 + -2.5 = -2.5 (> -∞)

best_score[1] = -2.5

best_edge[1] = e_1

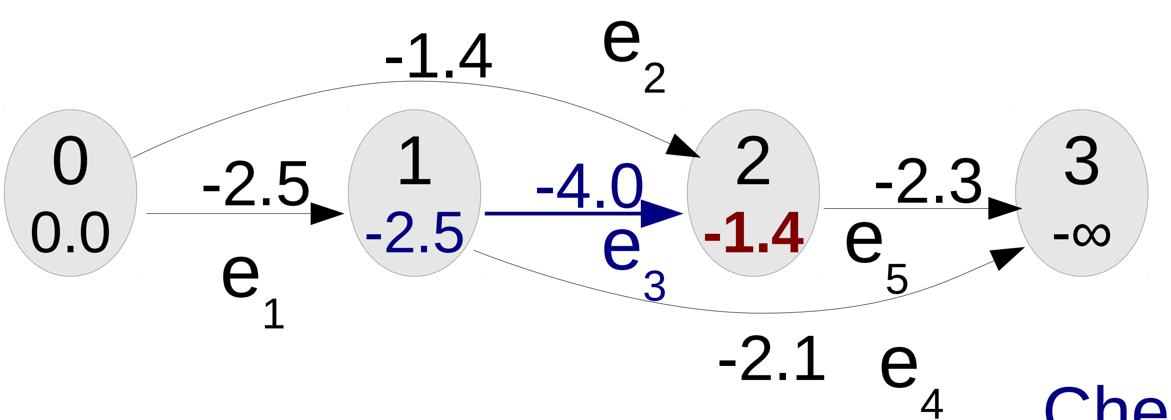
<u>Check e_2:</u>

score = 0 + -1.4 = -1.4 (> -∞)

best_score[2] = -1.4

best_edge[2] = e_2

Example:



Initialize:

best_score[0] = 0

Check e_1:

score = 0 + -2.5 = -2.5 (> -∞)

best_score[1] = -2.5

best_edge[1] = e_1

Check e_2:

score = 0 + -1.4 = -1.4 (> -∞)

best_score[2] = -1.4

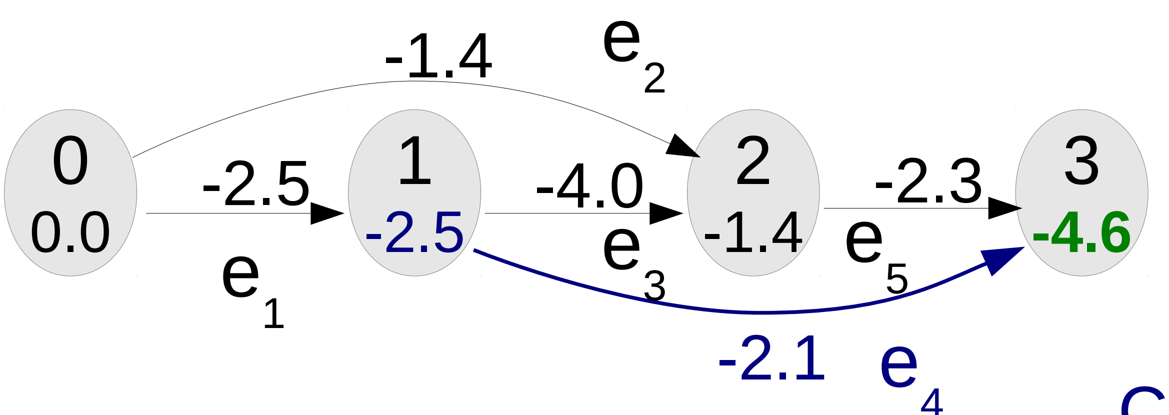
best_edge[2] = e_2

Check e_3:

score = -2.5 + -4.0 = -6.5 (< -1.4)

No change!

Example:



Initialize:

best_score[0] = 0

Check e_1:

score = 0 + -2.5 = -2.5 (> -∞)

best_score[1] = -2.5

best_edge[1] = e_1

Check e_2:

score = 0 + -1.4 = -1.4 (> -∞)

best_score[2] = -1.4

best_edge[2] = e_2

Check e_3:

score = -2.5 + -4.0 = -6.5 (< -1.4

No change!

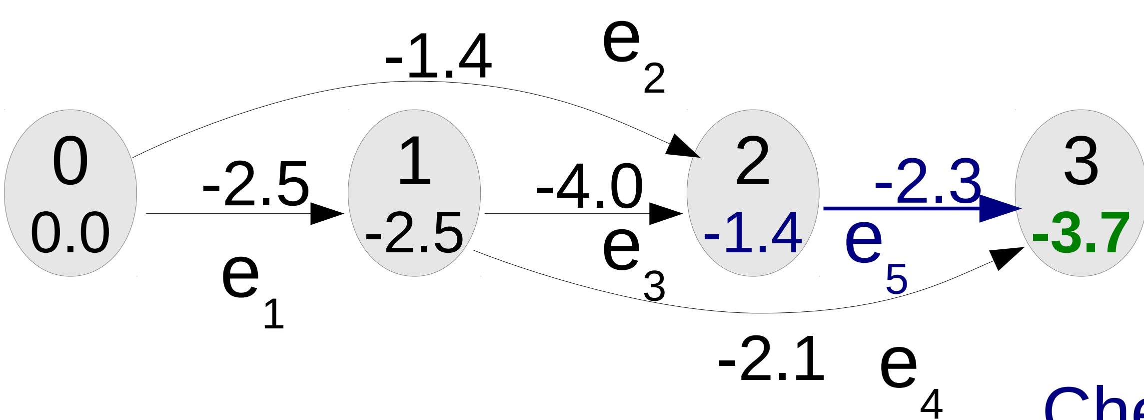
Check e_4:

score = -2.5 + -2.1 = -4.6 (> -∞)

best_score[3] = -4.6

best_edge[3] = e_4

Example:



Initialize:

best_score[0] = 0

Check e_1:

score = 0 + -2.5 = -2.5 (> -∞)

best_score[1] = -2.5

best_edge[1] = e_1

Check e_2:

score = 0 + -1.4 = -1.4 (> -∞)

best_score[2] = -1.4

best_edge[2] = e_2

Check e_3:

score = -2.5 + -4.0 = -6.5 (< -1.4)

No change!

Check e_4:

score = -2.5 + -2.1 = -4.6 (> -∞)

~~best_score[3] = -4.6~~

~~best_edge[3] = e_4~~

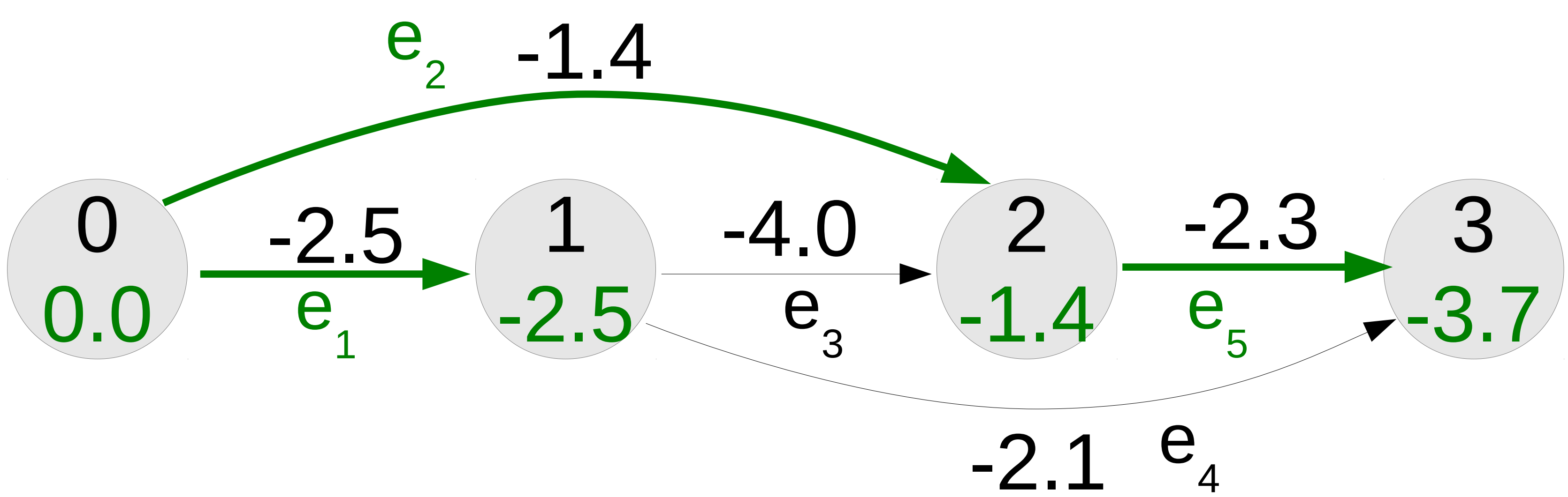
Check e_5:

score = -1.4 + -2.3 = -3.7 (> -4.6)

best_score[3] = -3.7

best_edge[3] = e_5

Result of Forward Step

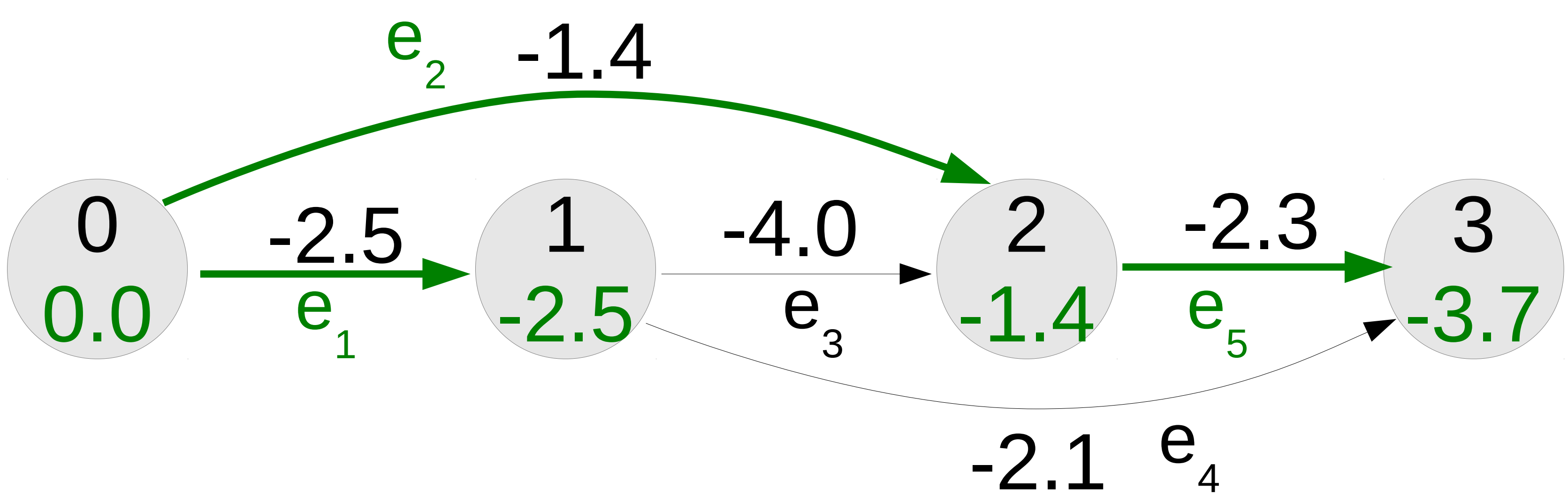


best_score = (0.0, -2.5, -1.4, -3.7)

best_edge = (NULL, e_1, e_2, e_5)

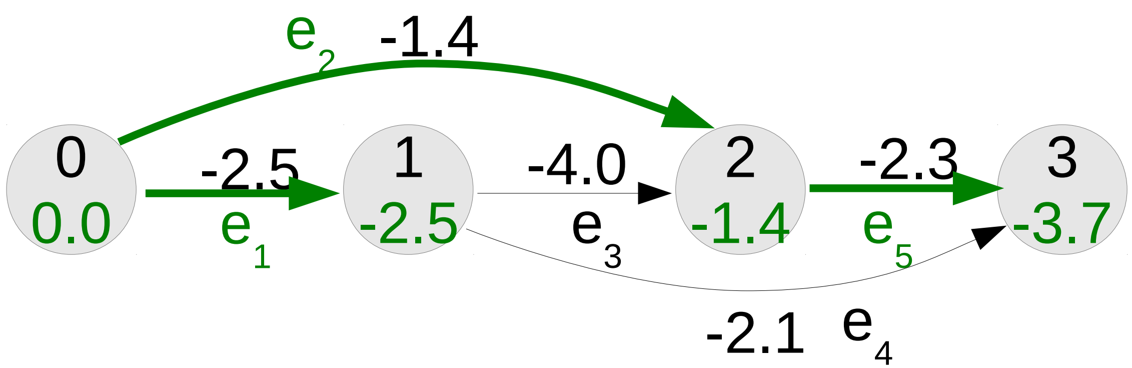
Backward Step

Backward Step



best_path = [ ]
next_edge = *best_edge*[*best_edge*.length – 1]
while *next_edge* != NULL
    add *next_edge* to *best_path*
    next_edge = *best_edge*[*next_edge.prev_node*]
reverse best_path

Example of Backward Step

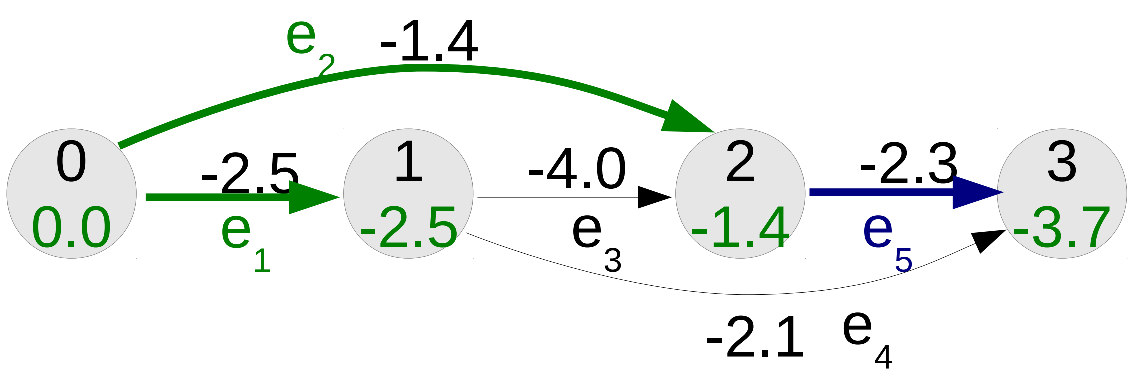


Initialize:

best_path = []

next_edge = best_edge[3] = e_5

Example of Backward Step



Initialize:

best_path = []

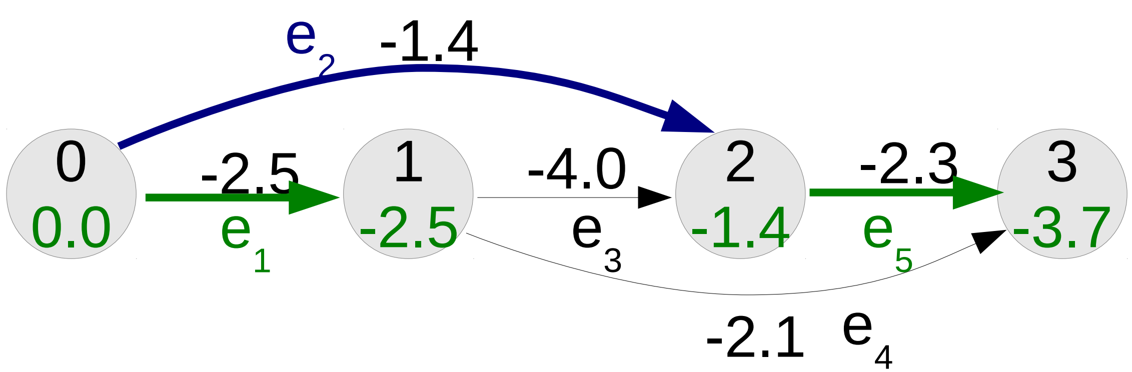
next_edge = best_edge[3] = e_5

Process e_5:

best_path = [e_5]

next_edge = best_edge[2] = e_2

Example of Backward Step



Initialize:

best_path = []

next_edge = best_edge[3] = e_5

Process e_2:

best_path = [e_5, e_2]

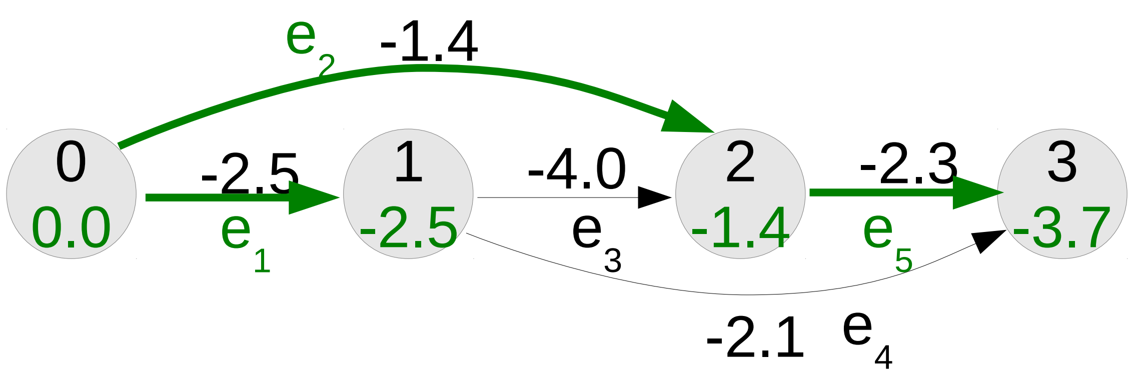
next_edge = best_edge[0] = NULL

Process e_5:

best_path = [e_5]

next_edge = best_edge[2] = e_2

Example of Backward Step



Initialize:

best_path = []
next_edge = best_edge[3] = e_5

Process e_5:

best_path = [e_5]
next_edge = best_edge[2] = e_2

Process e_5:

best_path = [e_5, e_2]
next_edge = best_edge[0] = NULL

Reverse:

best_path = [e_2, e_5]

Search with a Language Model

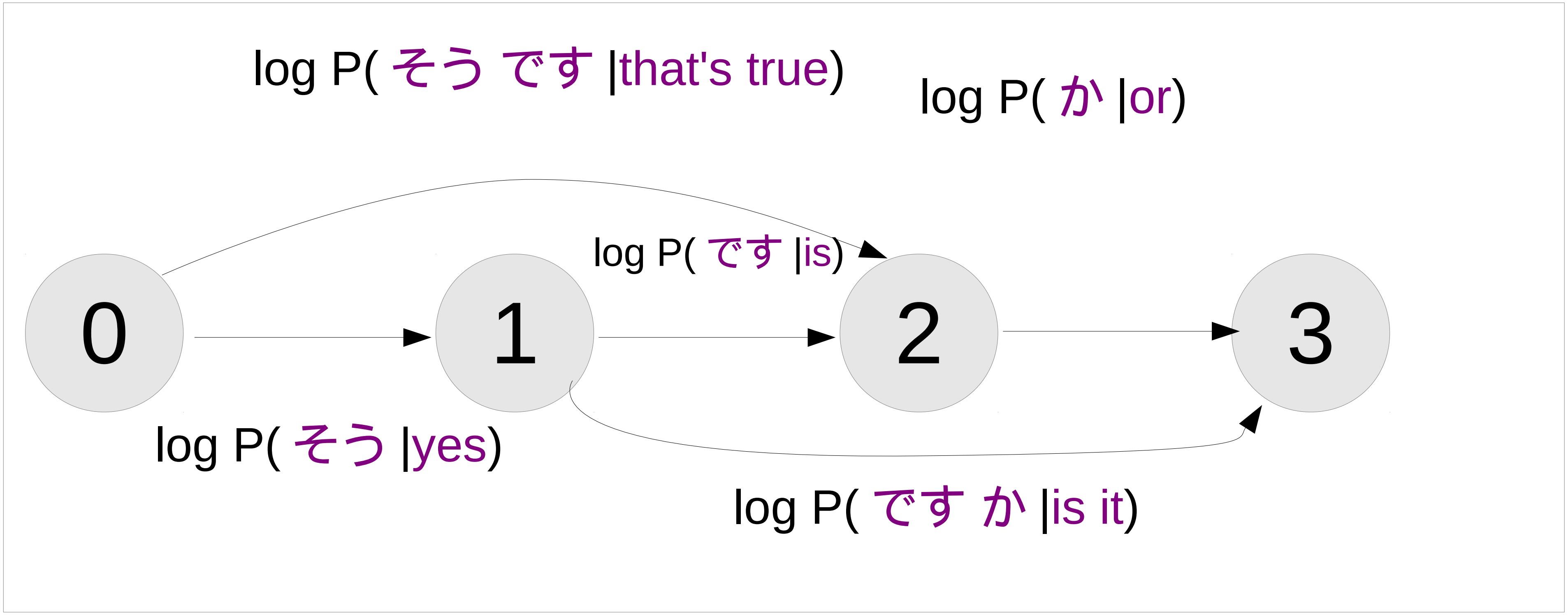
Language Model Probabilities

- Next let's add language model probabilities

$$P(D, E, F) \approx P(E)\,P(D_{fp} | D_{ep})$$

(still no reordering)

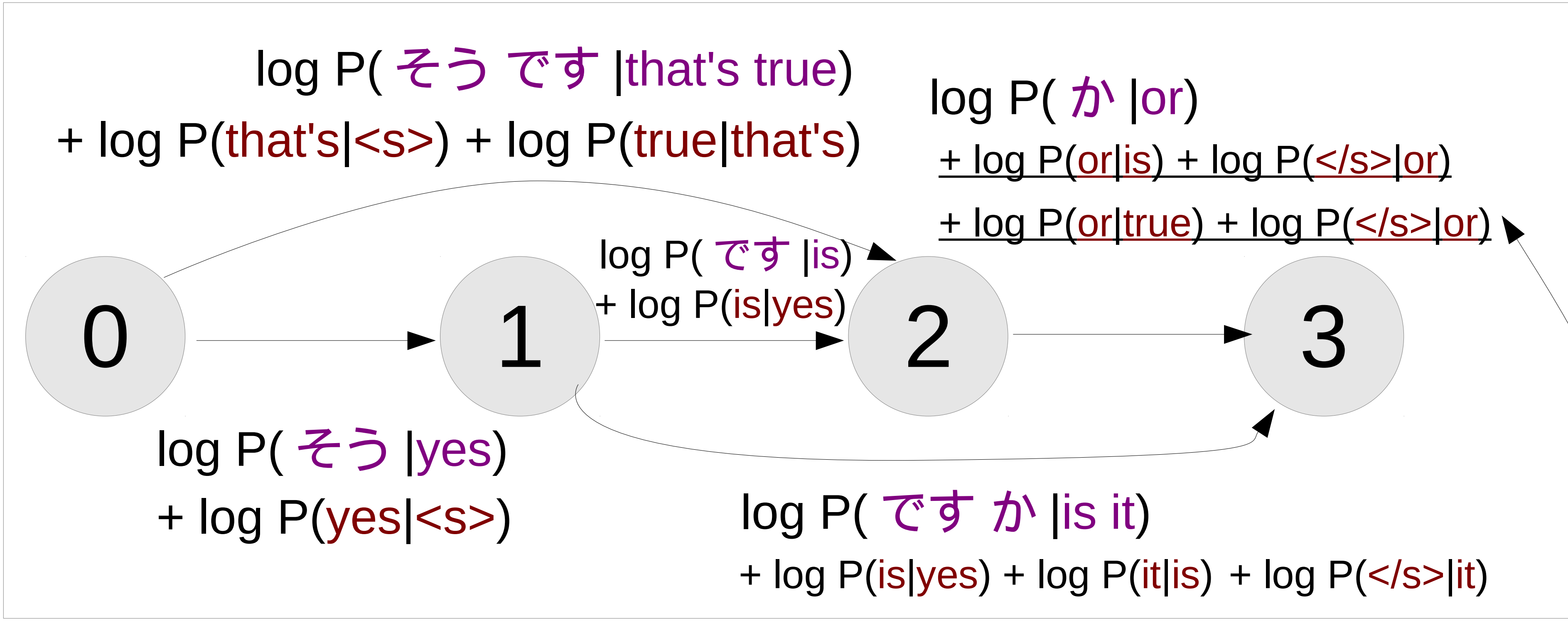
Graph Weights Review



- Each edge has a weight equal to $\log P(fp_k | ep_k)$
- Each path has a weight equal to sum of edges

$$\log P(D_{fp} | D_{ep}) = \sum_{k=1}^{K} \log P(fp_k | ep_k)$$

Adding Language Model Weights?

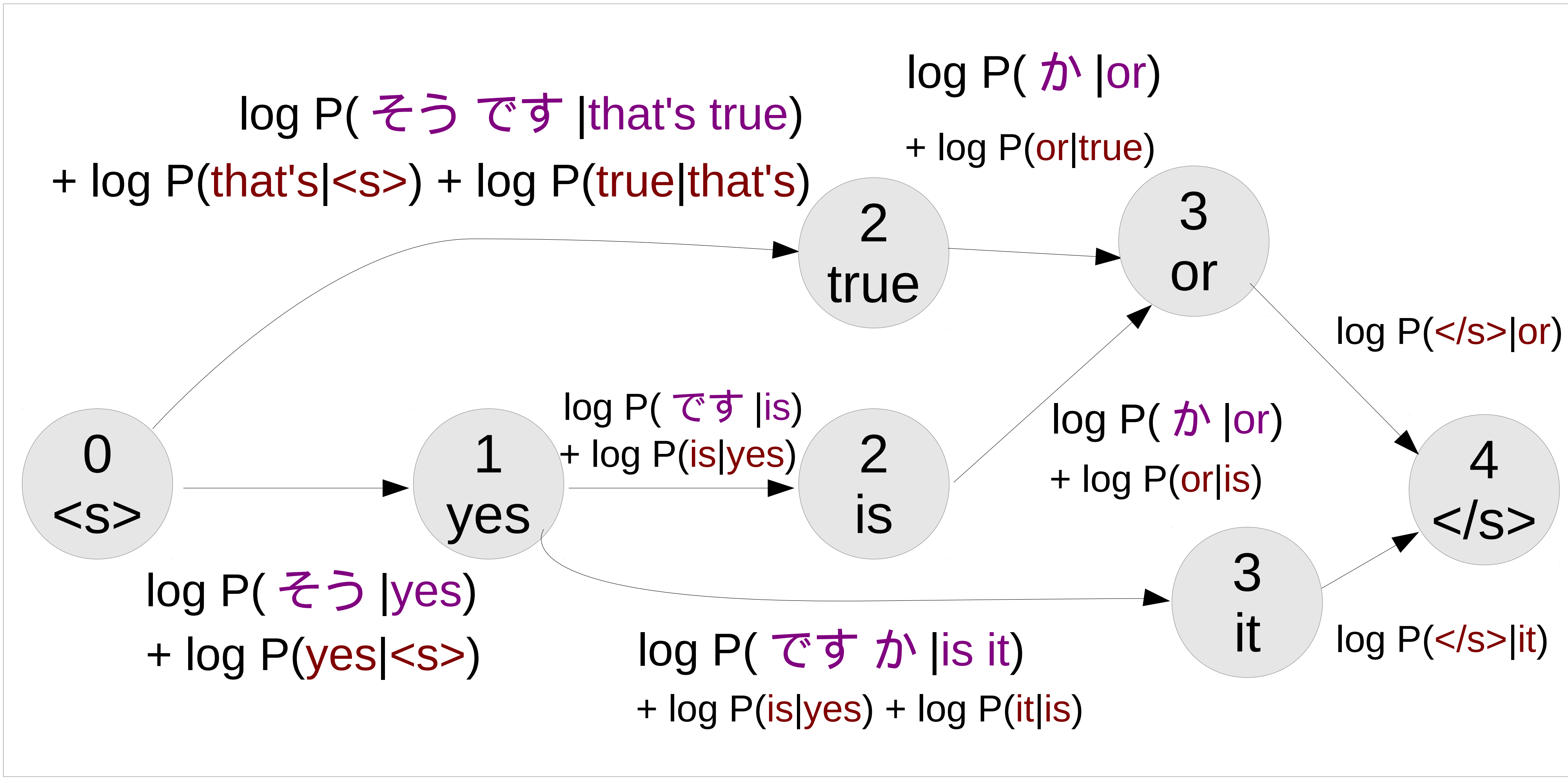


- How do we add word bigrams?

PROBLEM
We cannot decide which history to use!

Augmenting the States

- Remember the last translated word in every state!



- If using longer n-grams, remember n-1 words

Search with Reordering

Reordering

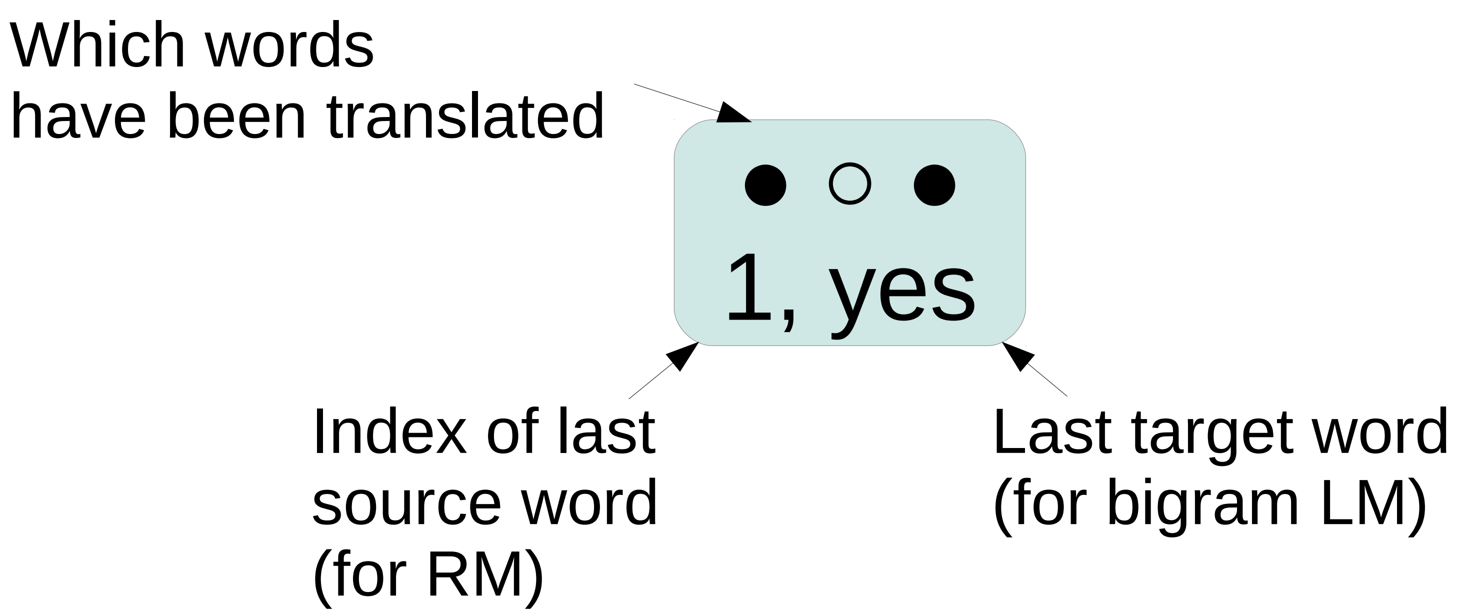
- Next let's allow reordering and add probabilities

$$P(D,E,F) \approx P(E)\,P(D_{fp} \mid D_{ep})\,P(D_o \mid D_{fp}, D_{ep})$$

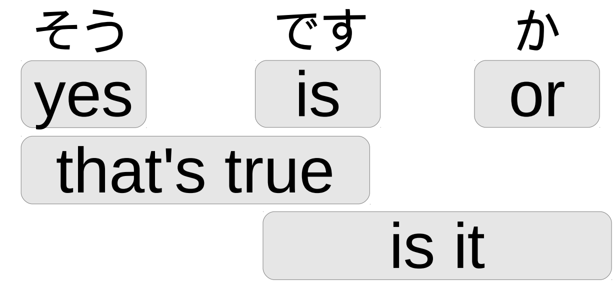
- What order do we calculate in?
- Basic idea:
  - Generate target sentence from left to right
  - Remember which words have been covered, last word translated

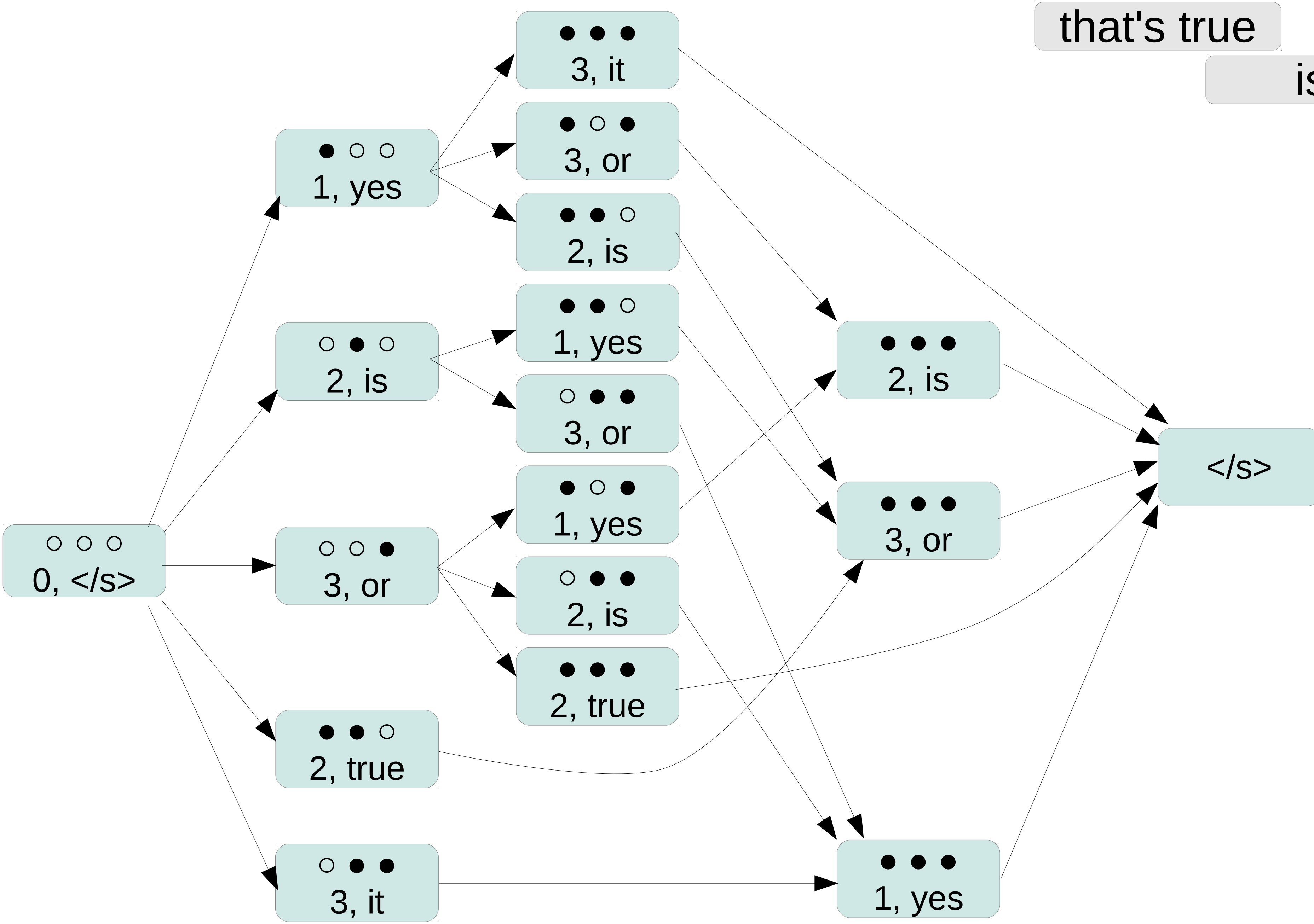
State Example

- Each state includes information about



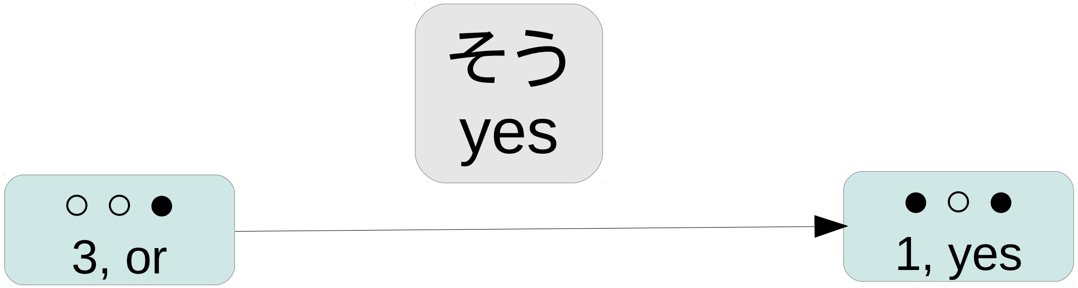
Search Example





Calculating Edge Probabilities

- Can calculate all probabilities based on the nodes/edge



log P(そう |yes)

+ log P(yes|or)

$+ \log e^{-\alpha o*3}$

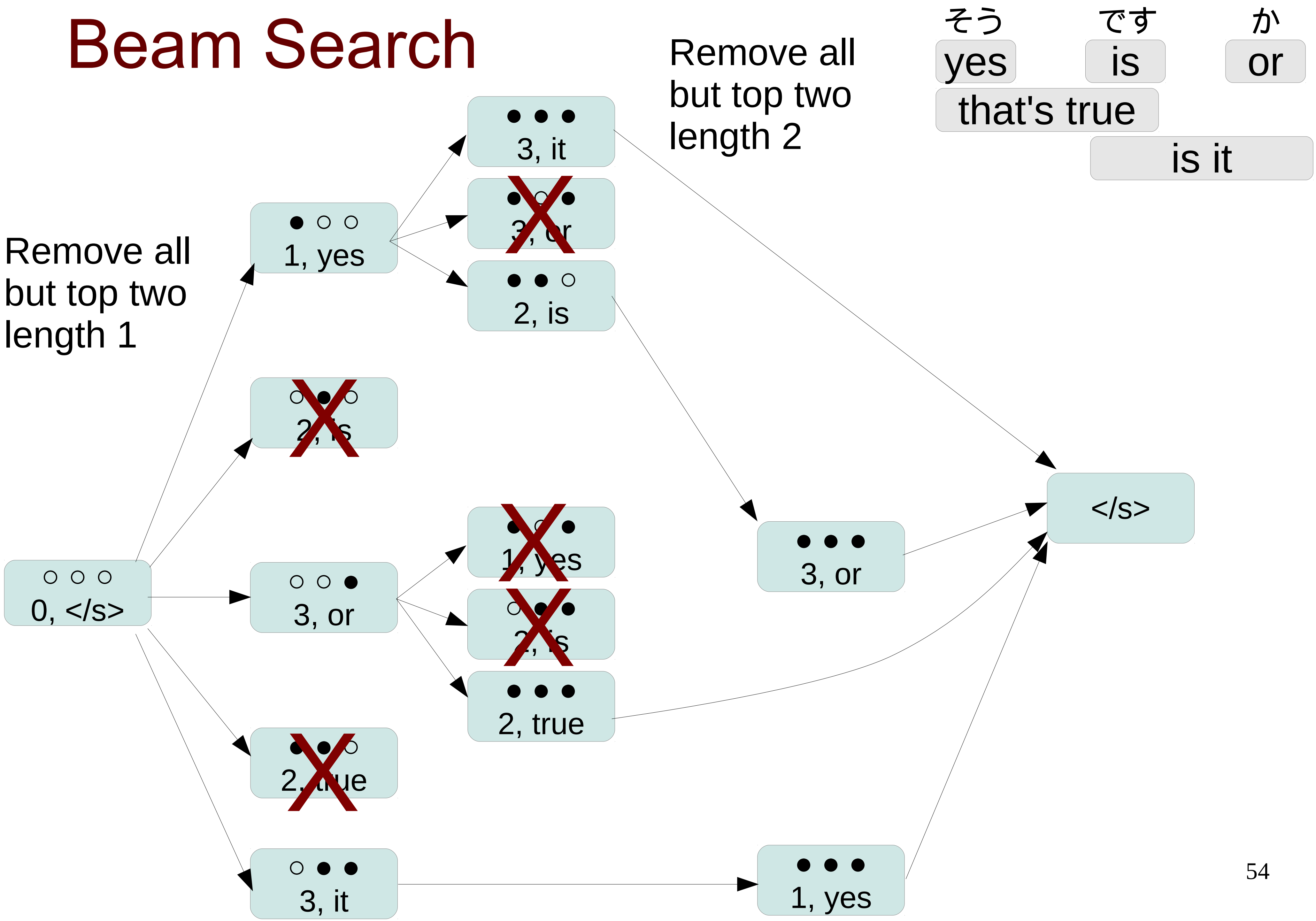
Problem!

- Exact search for phrase-based MT is NP-hard!
- Make two approximations:
  - Remove low-scoring hypotheses that have the same number of words translated ("Beam Search")
  - Limit maximum distortion

Beam Search

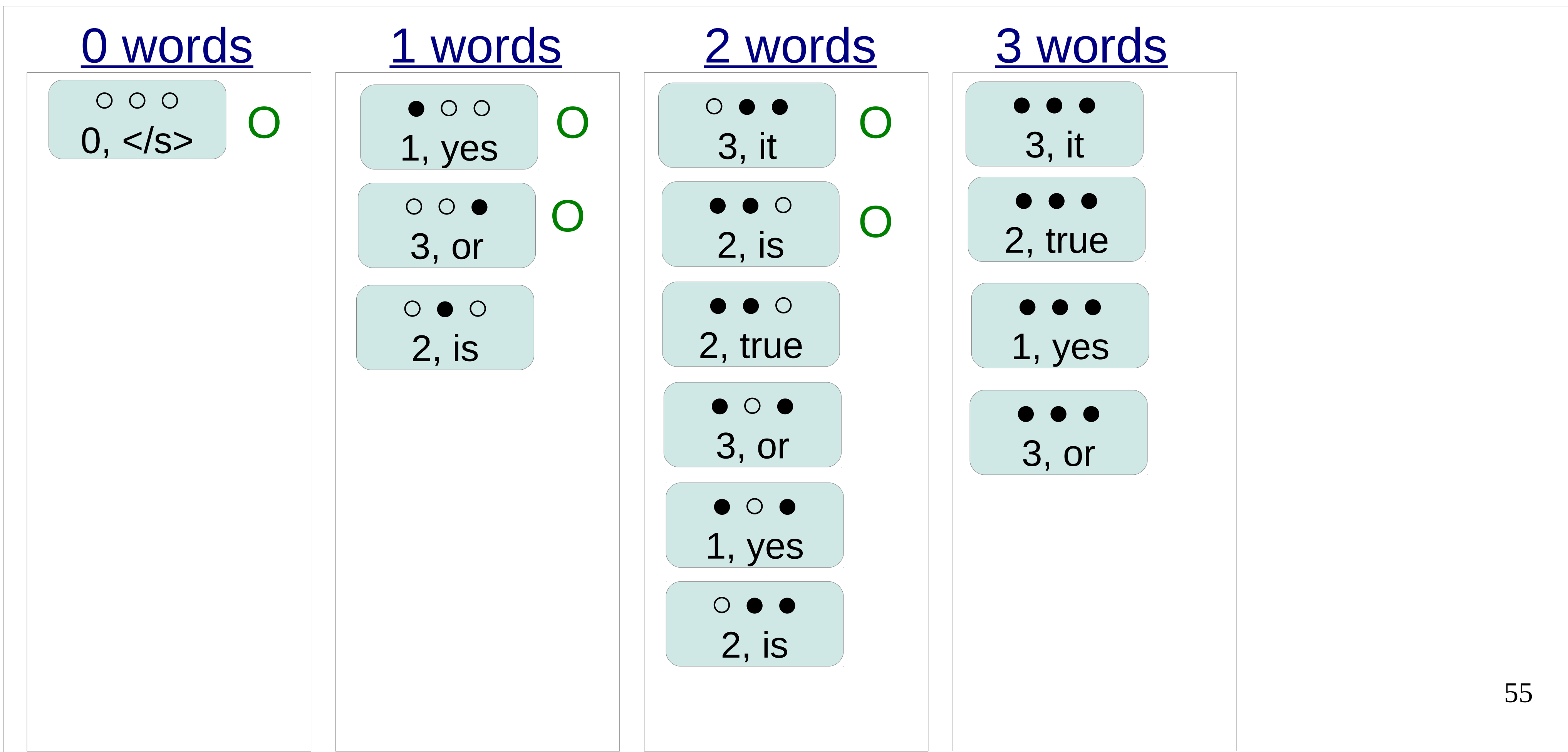
Remove all but top two length 1

Remove all but top two length 2

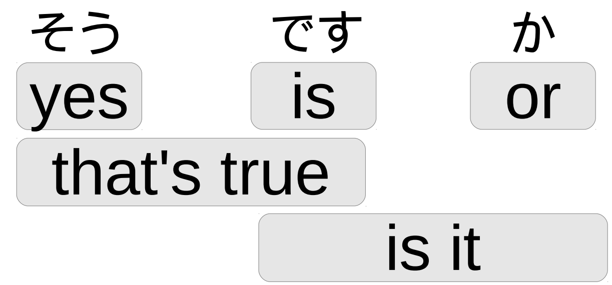


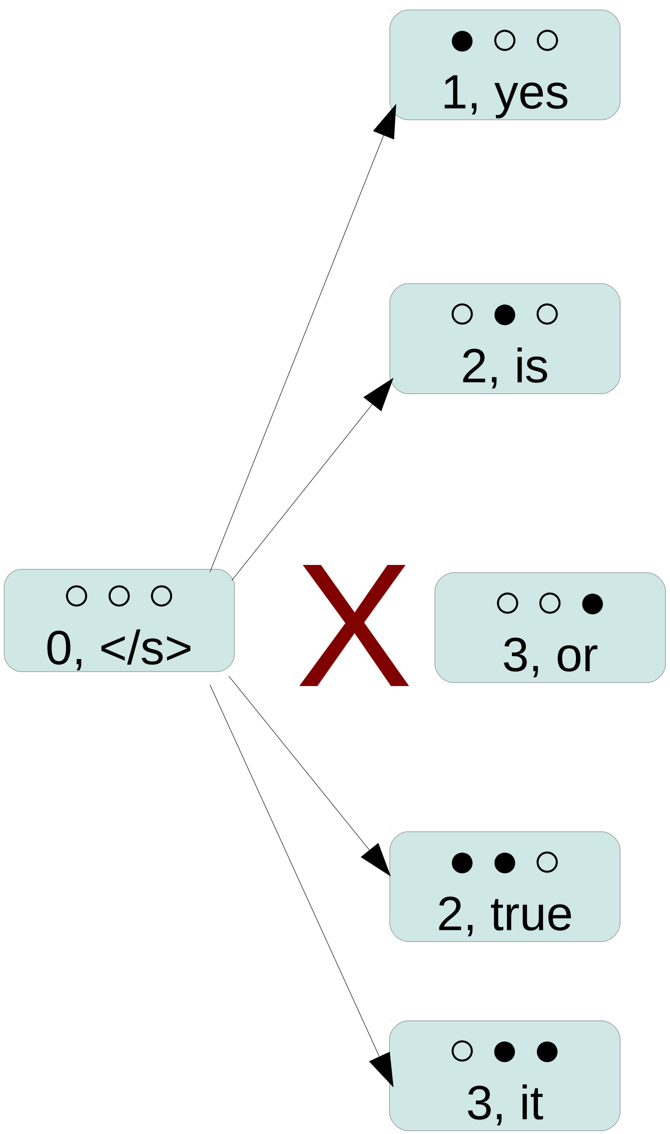
Stack Decoding

- We can also view this as “stacks” based on the number of words



Max Distortion Example





If distortion limit is “1”, then
this hypothesis has distortion of 2,
and will never be expanded

Evaluation

Evaluation

- We built a machine translation system, we need to know:
  - How good is our system?
  - Is system A better than system B?
  - What are the problems with our system?

Human Evaluation

- Adequacy: Is the meaning correct?
- Fluency: Is the sentence natural?
- Pairwise: Is X a better translation than Y?

太郎が花子を訪問した

	Taro visited Hanako	the Taro visited the Hanako	Hanako visited Taro
Adequate?	○	○	X
Fluent?	○	X	○
Better?	B, C	C	

Automatic Evaluation

- How well does the translation match a reference?
  - (or multiple references: more than one correct translation)
- BLEU: n-gram precision, brevity penalty [Papineni 03]

Reference: Taro visited Hanako

System: the Taro visited the Hanako

1-gram: 3/5
2-gram: 1/4
brevity penalty = 1.0

Brevity: min(1, |System|/|Reference|) = min(1, 5/3)

BLEU-2 = $(3/5*1/4)^{1/2}$ * 1.0
= 0.387

- Also METEOR (normalizes synonyms), TER (# of changes), RIBES (reordering)

Assignment

Assignment

- (Only one assignment this week)
- You are given a baseline machine translation system
  - LM/Alignment: Baseline from exercises 1, 2
  - TM: Phrases of up to length 4
  - SM: Uniform distribution
  - RM: Distortion penalty
  - Reordering Limit: 6
- Try to improve its accuracy by changing one of the features listed above, or anything else

Assignment Details

- Download the exercise from the web
- You can find a list of commands to run in run-translate.sh
- Send any files you changed, BLEU score before/after, and a short description of the change
  - Due date: February 12th, 23:59
  - Address: neubig@is.naist.jp