

Machine Translation

COMP90042

Natural Language Processing

Lecture 17

Semester 1 2021 Week 9
Jey Han Lau



THE UNIVERSITY OF

MELBOURNE

Introduction

- **Machine translation (MT)** is the task of translating text from one **source language** to another **target language**

虽然北风呼啸，但天空依然十分清澈



However, the sky remained clear under the strong north wind

Why?

- Removes language barrier
- Makes information in any languages accessible to anyone
- But translation is a classic “AI-hard” challenge
 - ▶ Difficult to preserve the meaning and the fluency of the text after translation

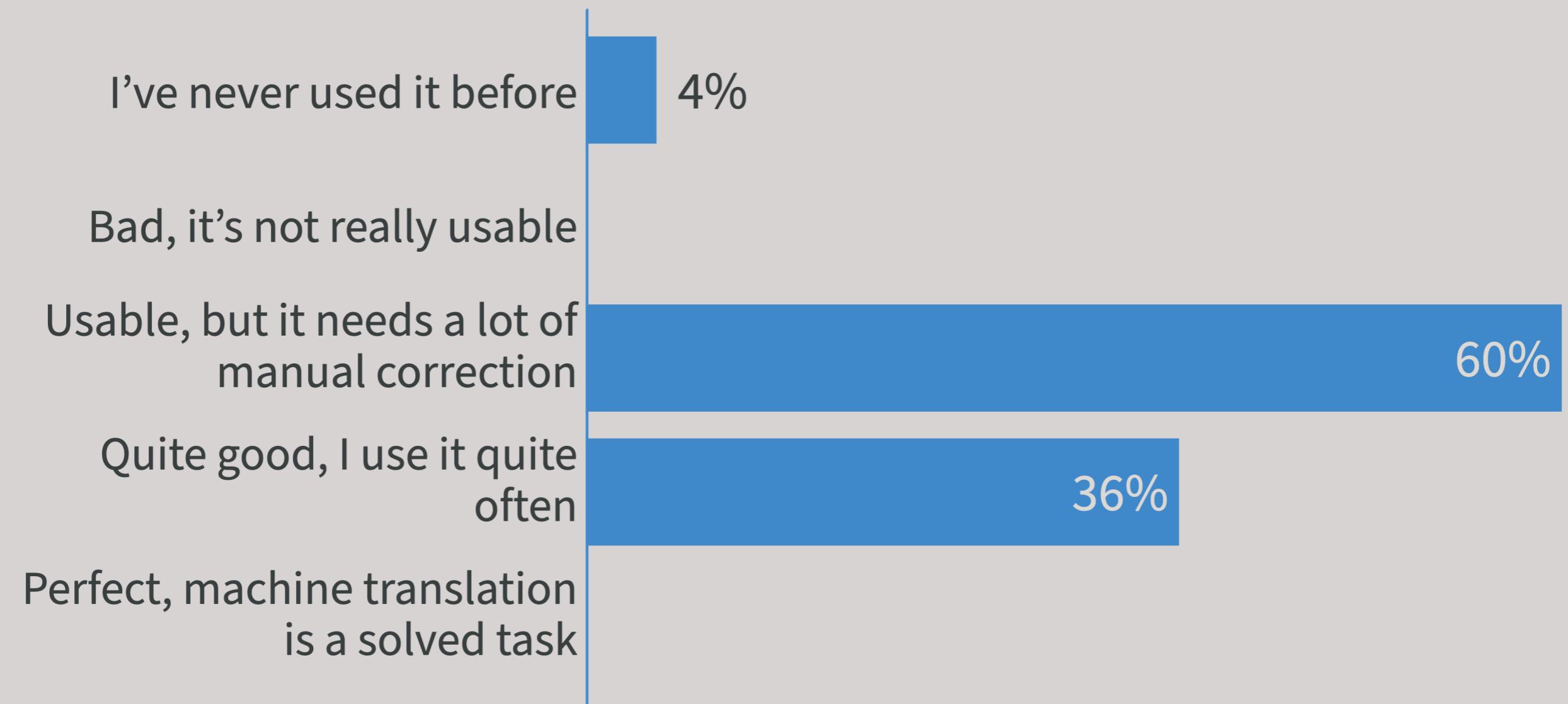
How good do you find machine translation (e.g. Google Translate)?

- I've never used it before
- Bad, it's not really usable
- Usable, but it needs a lot of manual correction
- Quite good, I use it quite often
- Perfect, machine translation is a solved task

PollEv.com/jeyhanlau569



How good do you find machine translation (e.g. Google Translate)?



MT is Difficult

- Not just simple word for word translation
- Structural changes, e.g., syntax and semantics
- Multiple word translations, idioms
- Inflections for gender, case etc
- Missing information (e.g., determiners)

虽然 北 风 呼啸 , 但 天空 依然 十分 清澈
although north wind howls , but sky still extremely limpid

Outline

- Statistical Machine Translation
- Neural Machine Translation
- Attention Mechanism
- Evaluation

Statistical MT

Early Machine Translation

- Started in early 1950s
- Motivated by the Cold War to translate Russian to English
- Rule-based system
 - ▶ Use bilingual dictionary to map Russian words to English words
- Goal: translate 1-2 million words an hour within 5 years

Statistical MT

- Given French sentence f , aim is to find the best English sentence e
 - ▶ $\operatorname{argmax}_e P(e | f)$
- Use Baye's rule to decompose into two components
 - ▶ $\operatorname{argmax}_e \color{blue}{P(f|e)} \color{red}{P(e)}$

Language vs Translation Model

- $\operatorname{argmax}_e P(f|e)P(e)$
- $P(e)$: **language model**
 - ▶ learns how to write fluent English text
- $P(f|e)$: **translation model**
 - ▶ learns how to translate words and phrases from English to French

How to Learn LM and TM?

- Language model:
 - ▶ Text statistics in large **monolingual corpora**
- Translation model:
 - ▶ Word co-occurrences in **parallel corpora**
 - ▶ i.e. English-French sentence pairs

Parallel Corpora

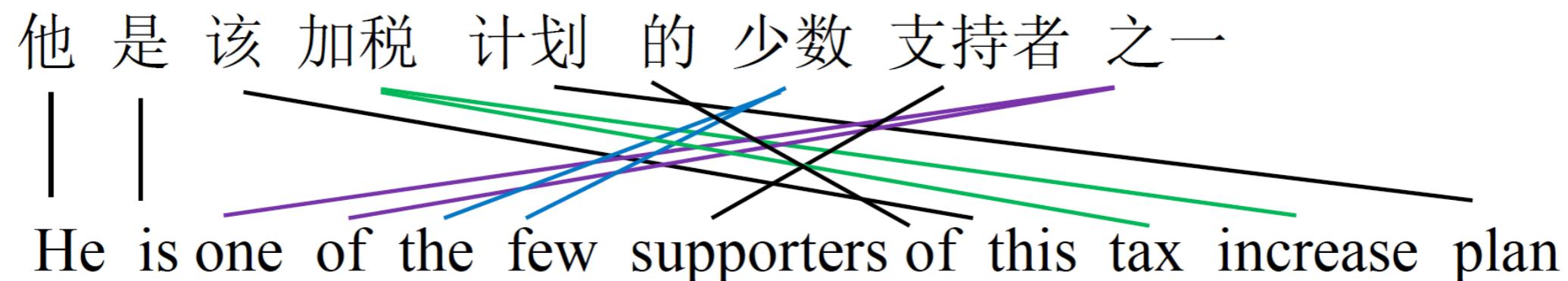
- One text in multiple languages
- Produced by human translation
 - ▶ Bible, news articles, legal transcripts, literature, subtitles
 - ▶ Open parallel corpus:
<http://opus.nlpl.eu/>



The Rosetta Stone

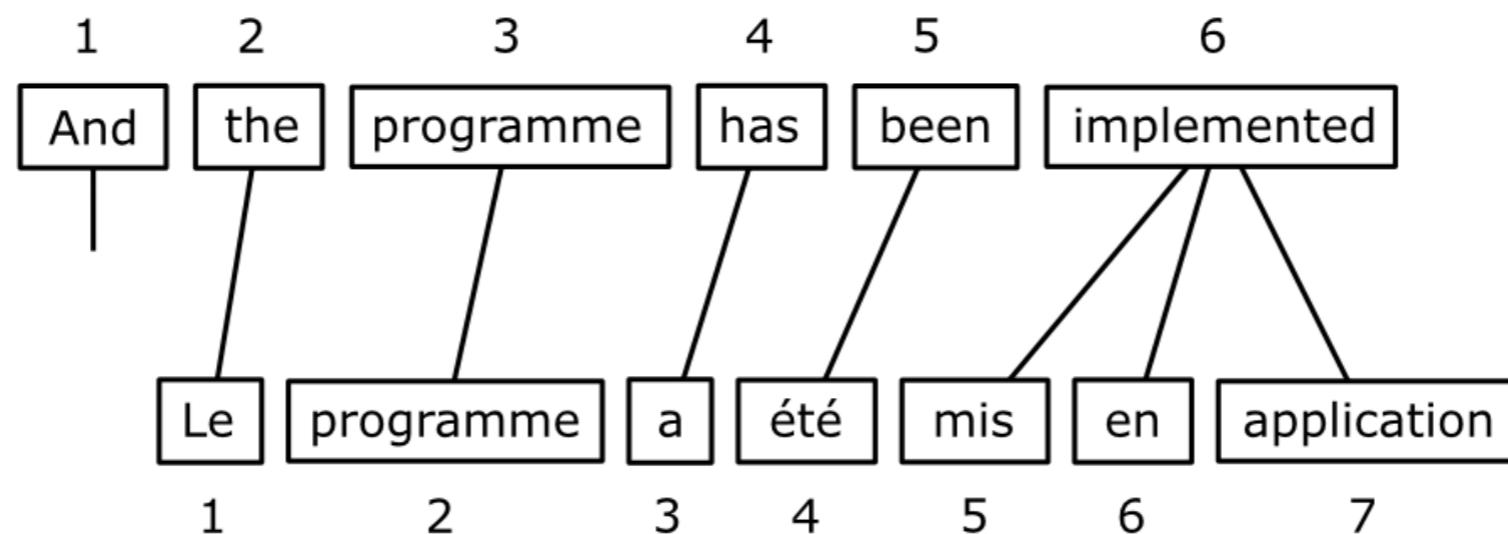
Models of Translation

- How to learn $P(f|e)$ from parallel text?
- We only have sentence pairs; **words are not aligned** in the parallel text
- I.e. we don't have word to word translation



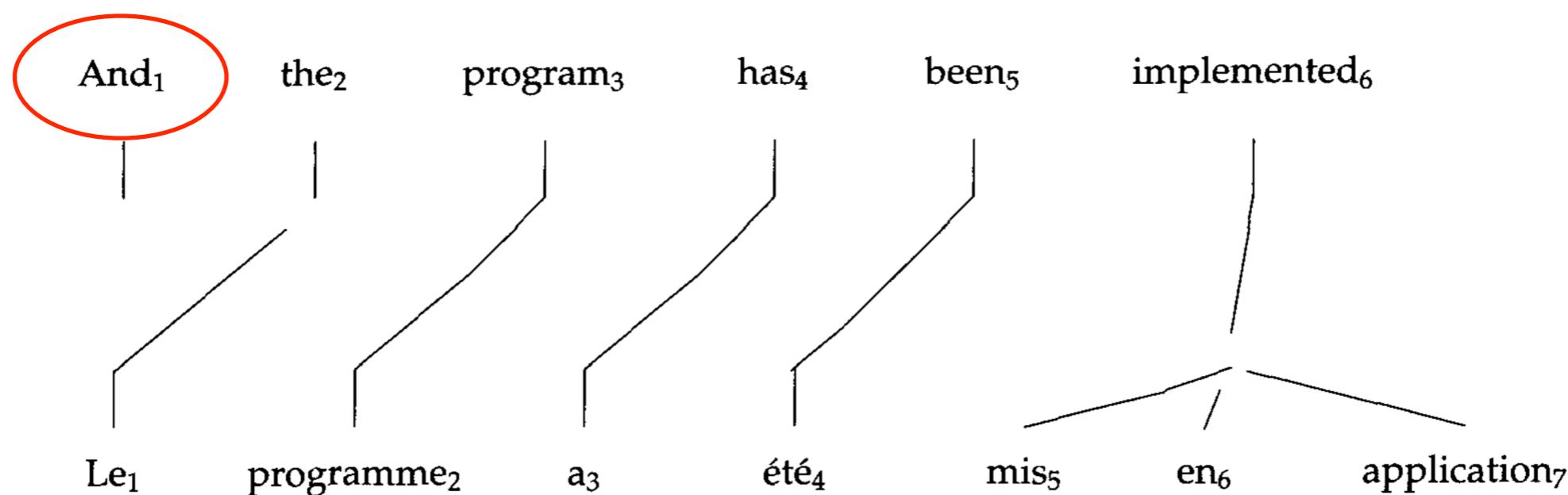
Alignment

- Idea: introduce **word alignment** as a latent variable into the model
 - ▶ $P(f, a | e)$
- Use algorithms such as expectation maximisation (EM) to learn (e.g. GIZA++)



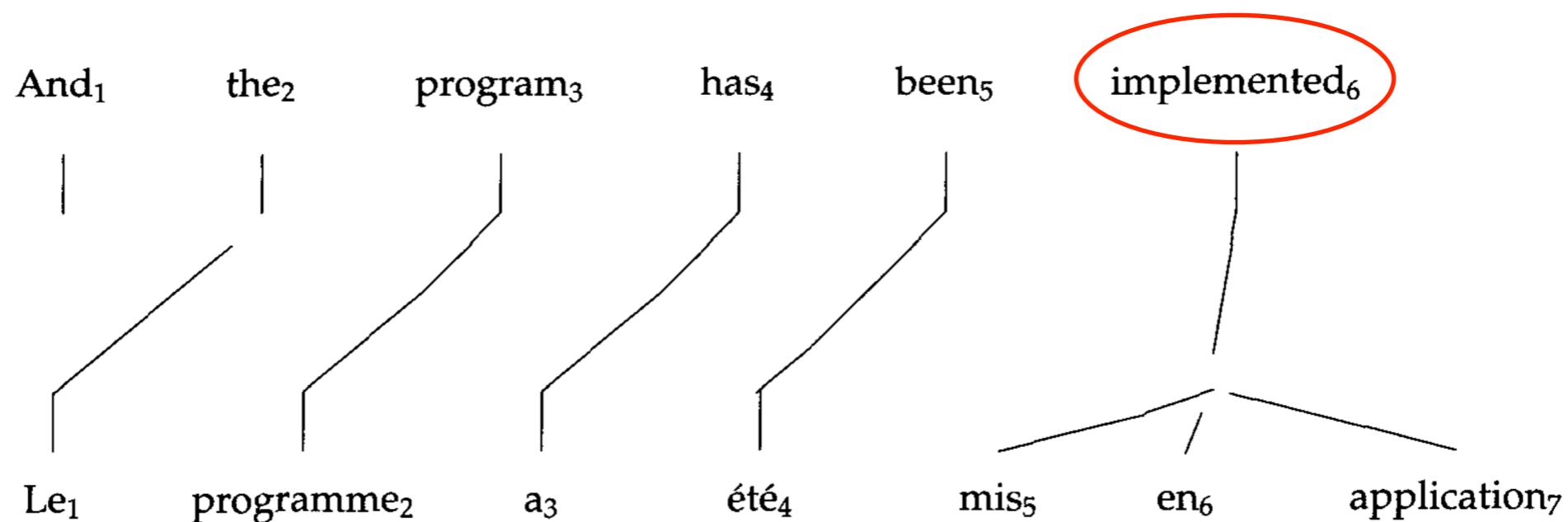
Complexity of Alignment

- Some words are dropped and have no alignment



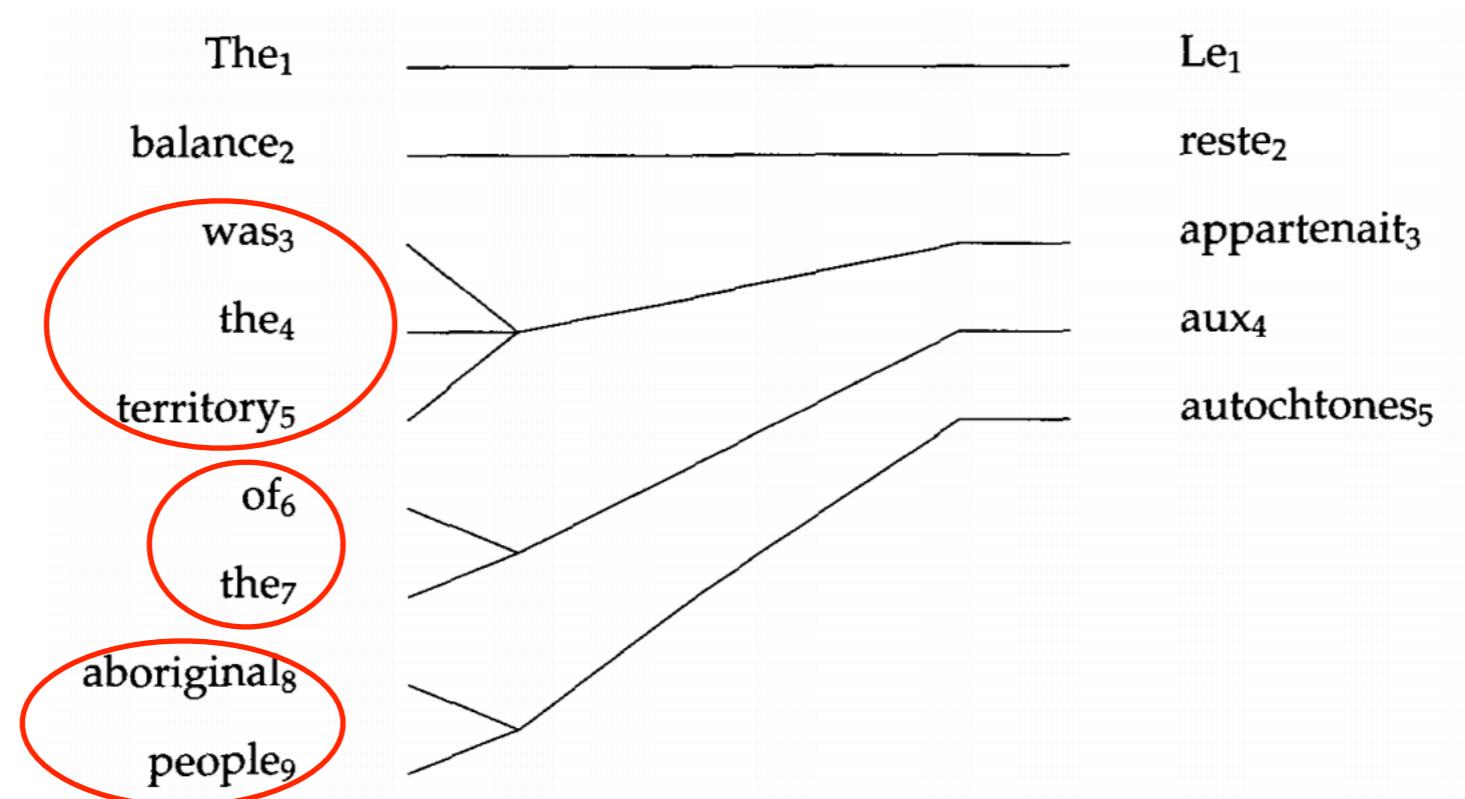
Complexity of Alignment

- One-to-many alignment



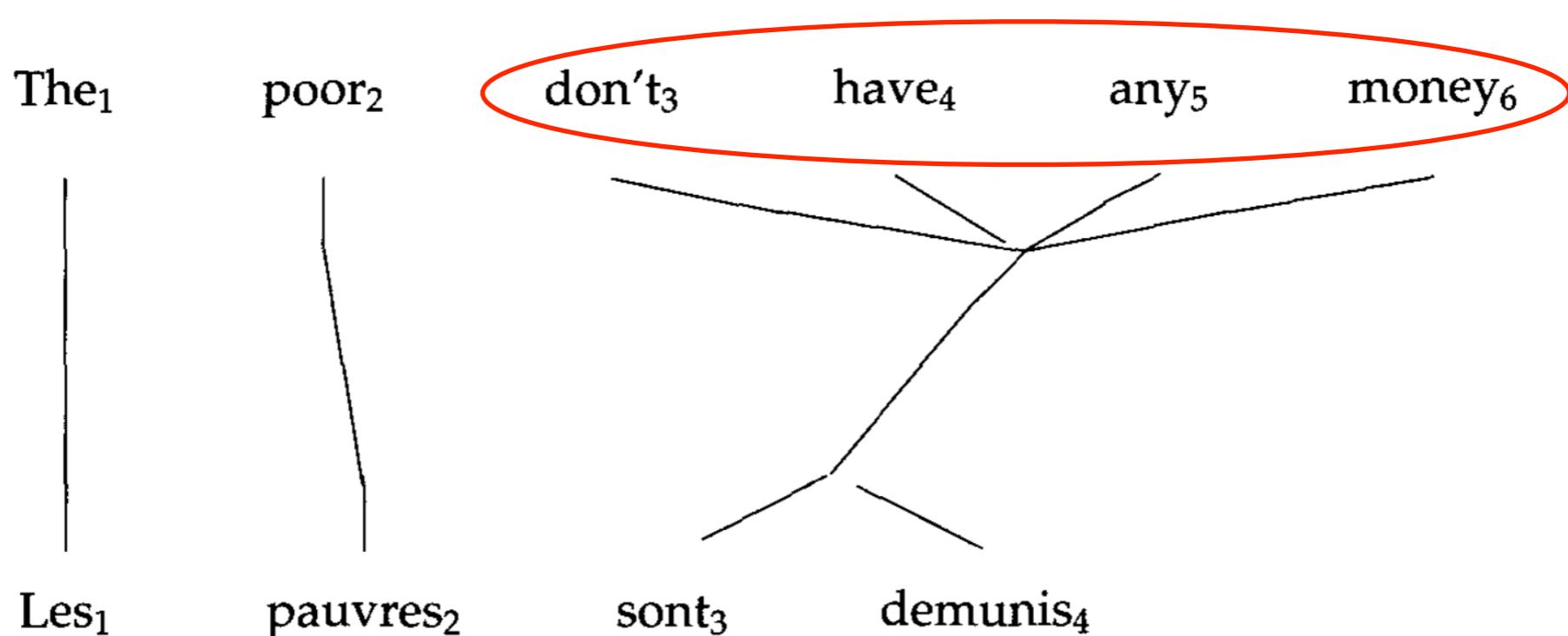
Complexity of Alignment

- Many-to-one alignment



Complexity of Alignment

- Many-to-many alignment



Statistical MT: Summary

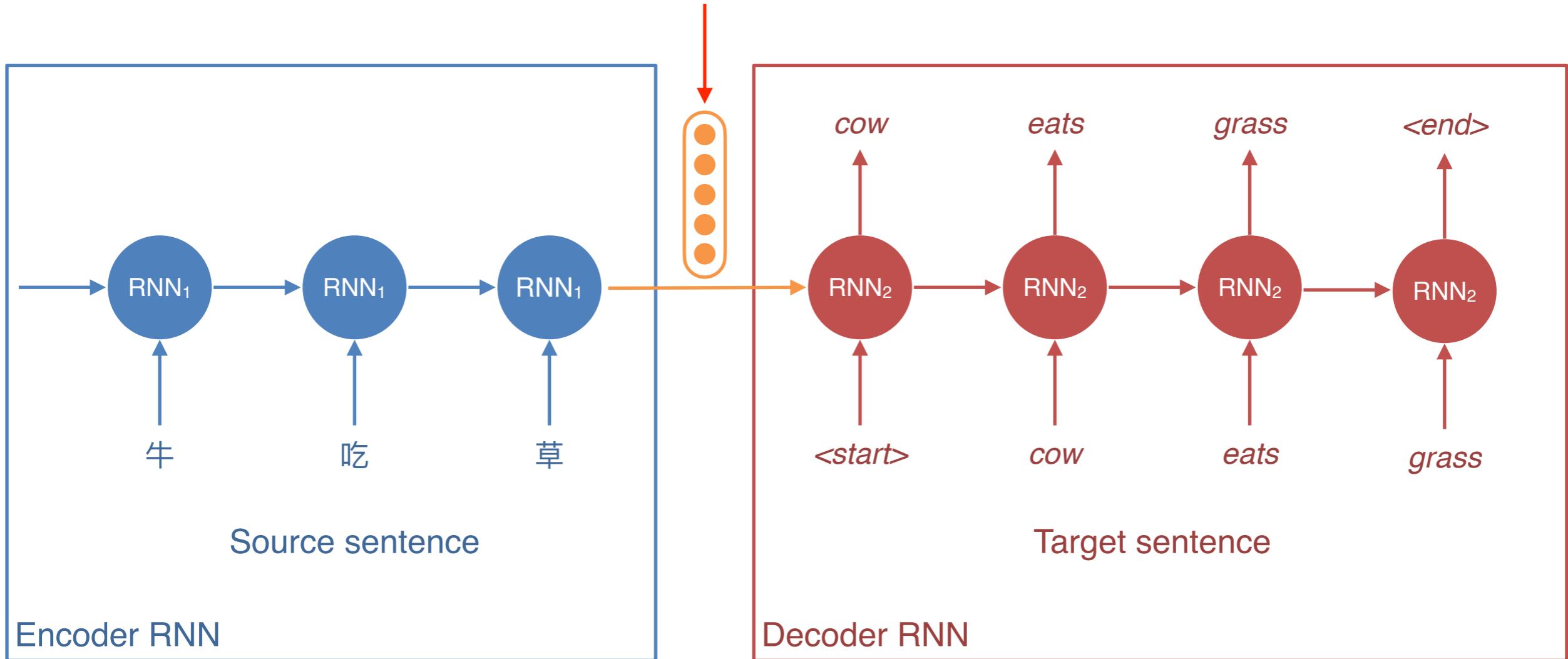
- A very popular field of research in NLP prior to 2010s
- Lots of feature engineering
- State-of-the-art systems are very complex
 - ▶ Difficult to maintain
 - ▶ Significant effort needed for new language pairs

Neural Machine Translation

Introduction

- Neural machine translation is a new approach to do machine translation
- Use a single neural model to directly translate from source to target
- Requires parallel text
- Architecture: encoder-decoder model
 - ▶ 1st RNN to encode the source sentence
 - ▶ 2nd RNN to decode the target sentence

This vector encodes the whole **source sentence**;
it is used as the initial state for **decoder RNN**



$$h_i = RNN_1(h_{i-1}, x_i)$$

$$s_t = RNN_2(s_{t-1}, y_t)$$

$$s_1 = h_{|x|}$$

Neural MT

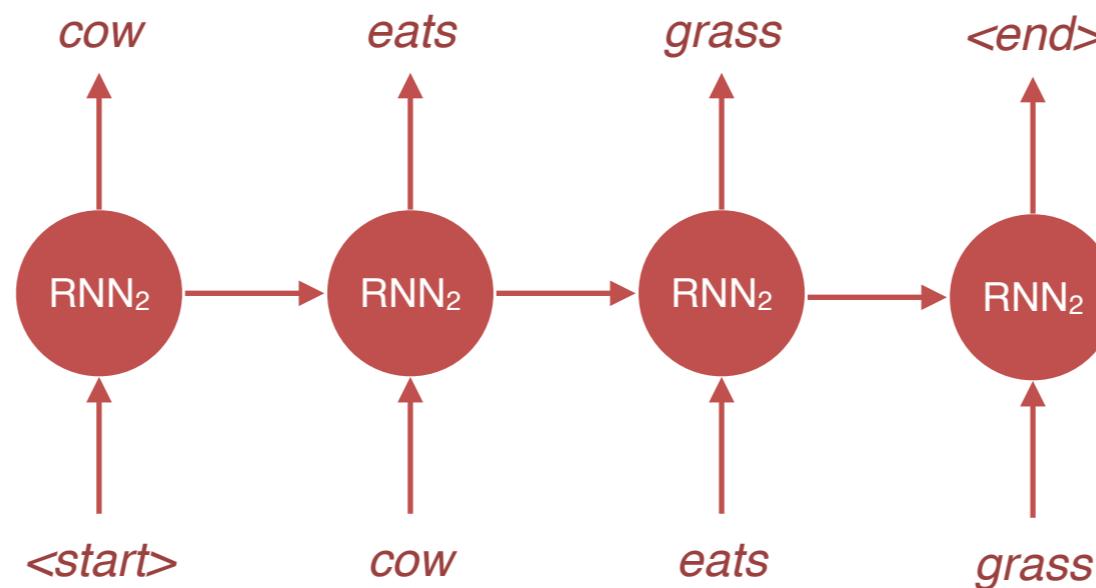
- The decoder RNN can be interpreted as a **conditional language model**
 - ▶ Language model: **predicts the next word given previous words in target sentence y**
 - ▶ Conditional: **prediction is also conditioned on the source sentence x**
- $P(y|x) = P(y_1|x)P(y_2|y_1, x)\dots P(y_t|y_1, \dots, y_{t-1}, x)$

Training Neural MT

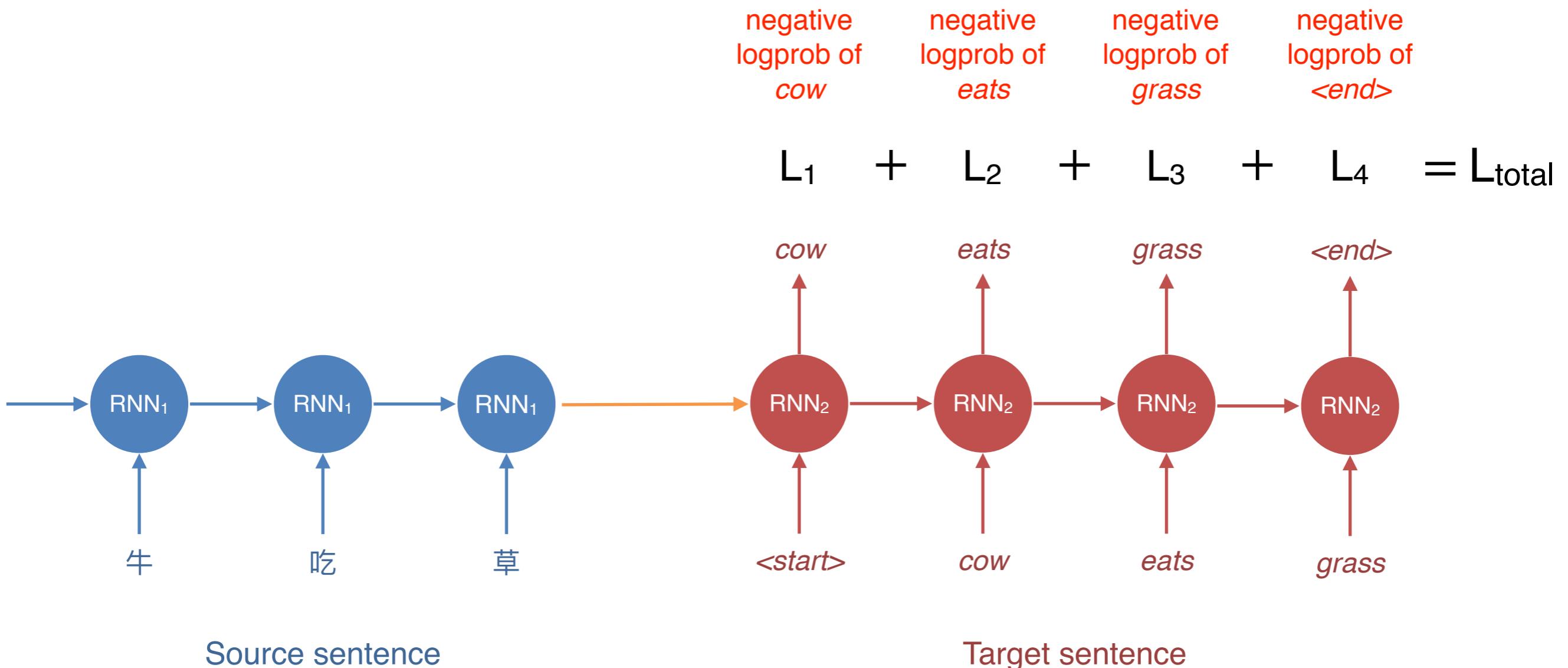
- Requires parallel corpus just like statistical MT
- Trains with next word prediction, just like a language model!

Language Model Training Loss

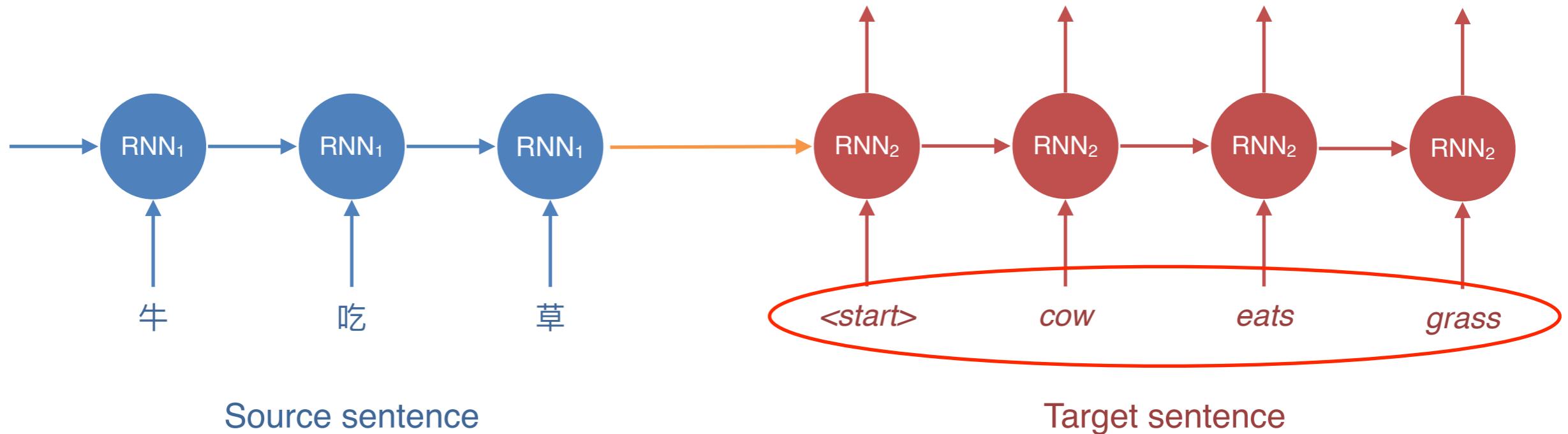
$$\begin{array}{cccc} \text{negative} & \text{negative} & \text{negative} & \text{negative} \\ \text{logprob of} & \text{logprob of} & \text{logprob of} & \text{logprob of} \\ \textit{cow} & \textit{eats} & \textit{grass} & \textit{<end>} \\ L_1 + L_2 + L_3 + L_4 = L_{\text{total}} \end{array}$$



Neural MT Training Loss

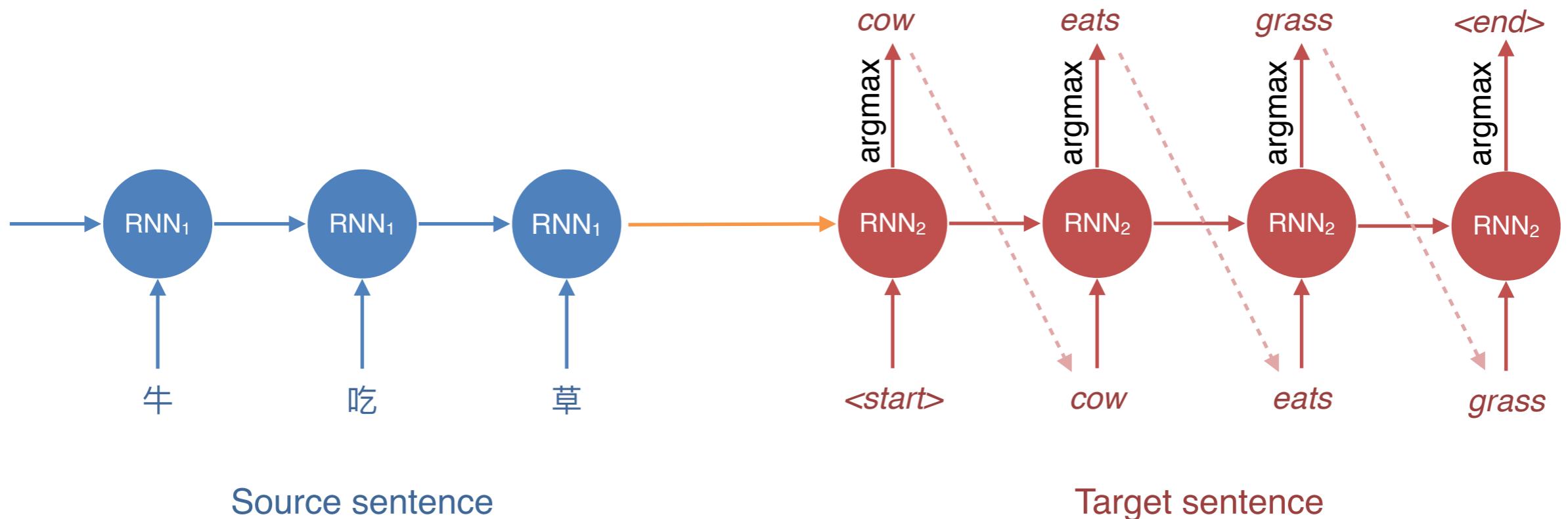


Training



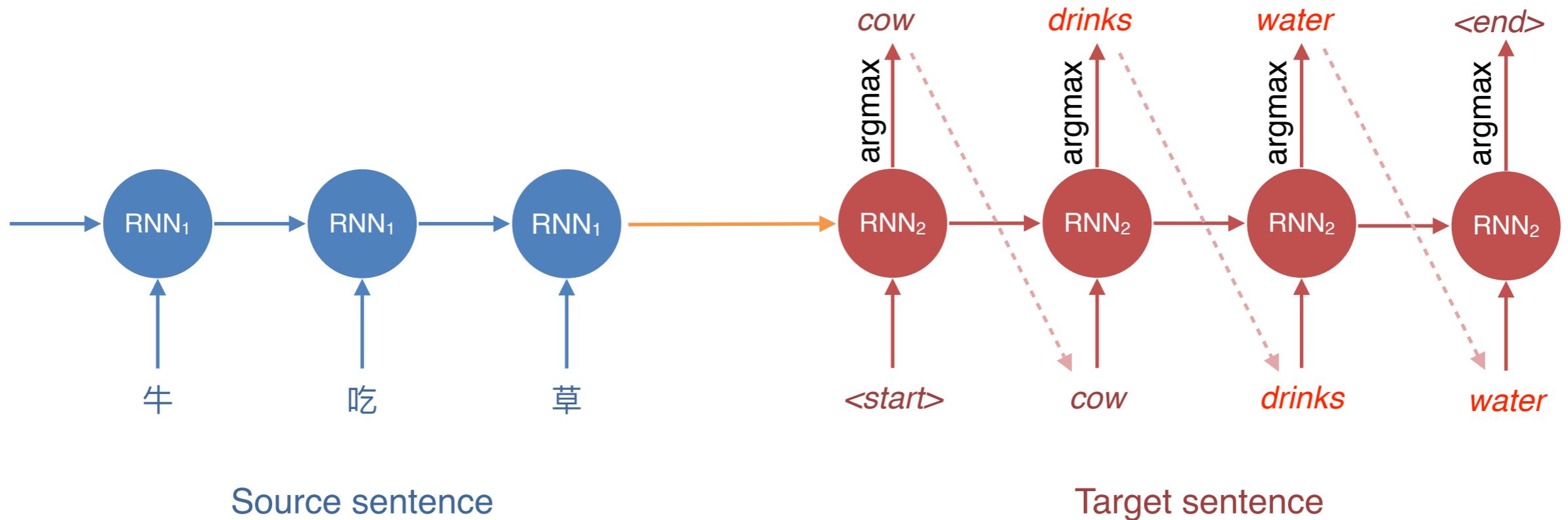
- During training, we have the target sentence
- We can therefore feed the right word from target sentence, one step at a time

Decoding at Test Time



- But at test time, we don't have the target sentence (that's what we're trying to predict!)
- **argmax:** take the word with the highest probability at every step

Exposure Bias



- Describes the discrepancy between training and testing
- Training: always have the ground truth tokens at each step
- Test: uses its own prediction at each step
- Outcome: model is unable to recover from its own error

Greedy Decoding

- argmax decoding is also called greedy decoding
- Issue: does not guarantee optimal probability
 $P(y | x)$

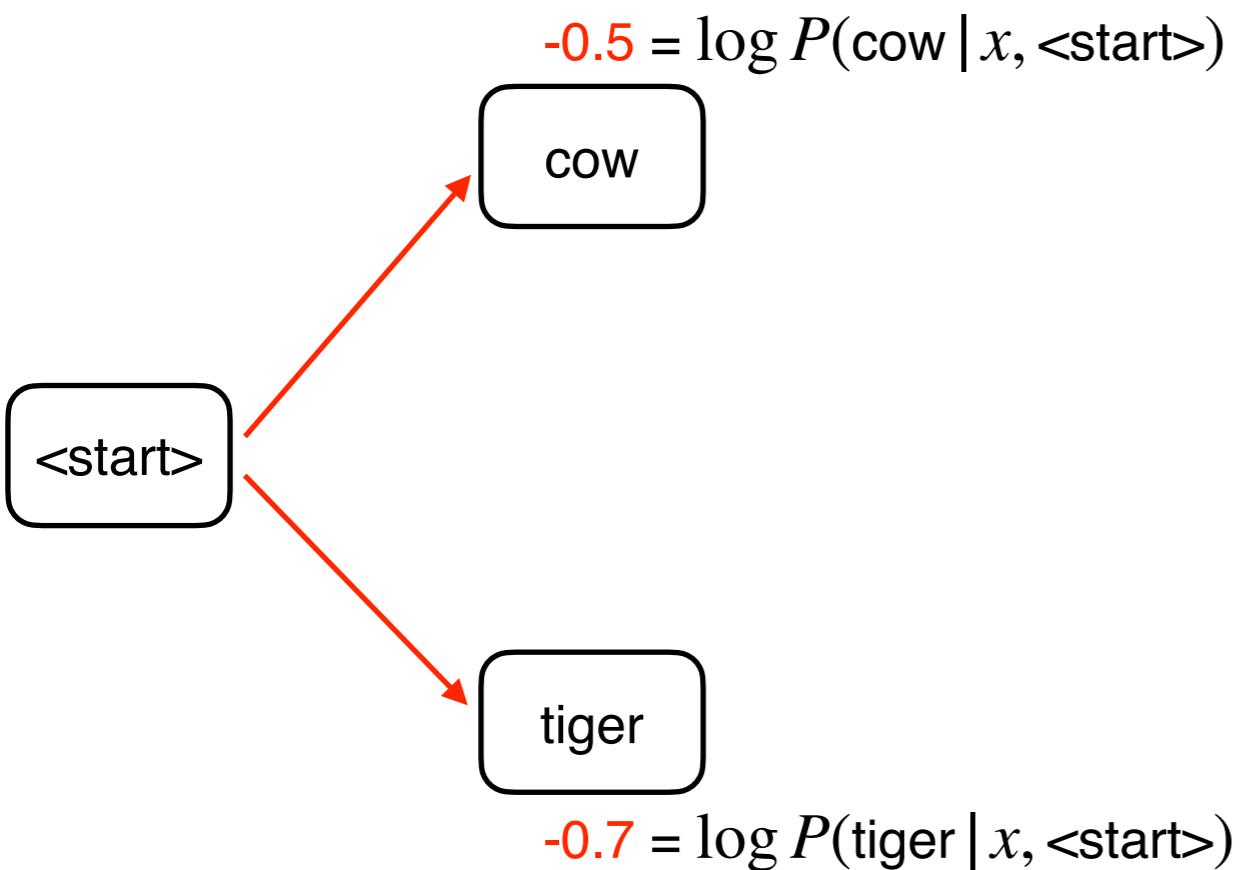
Exhaustive Search Decoding

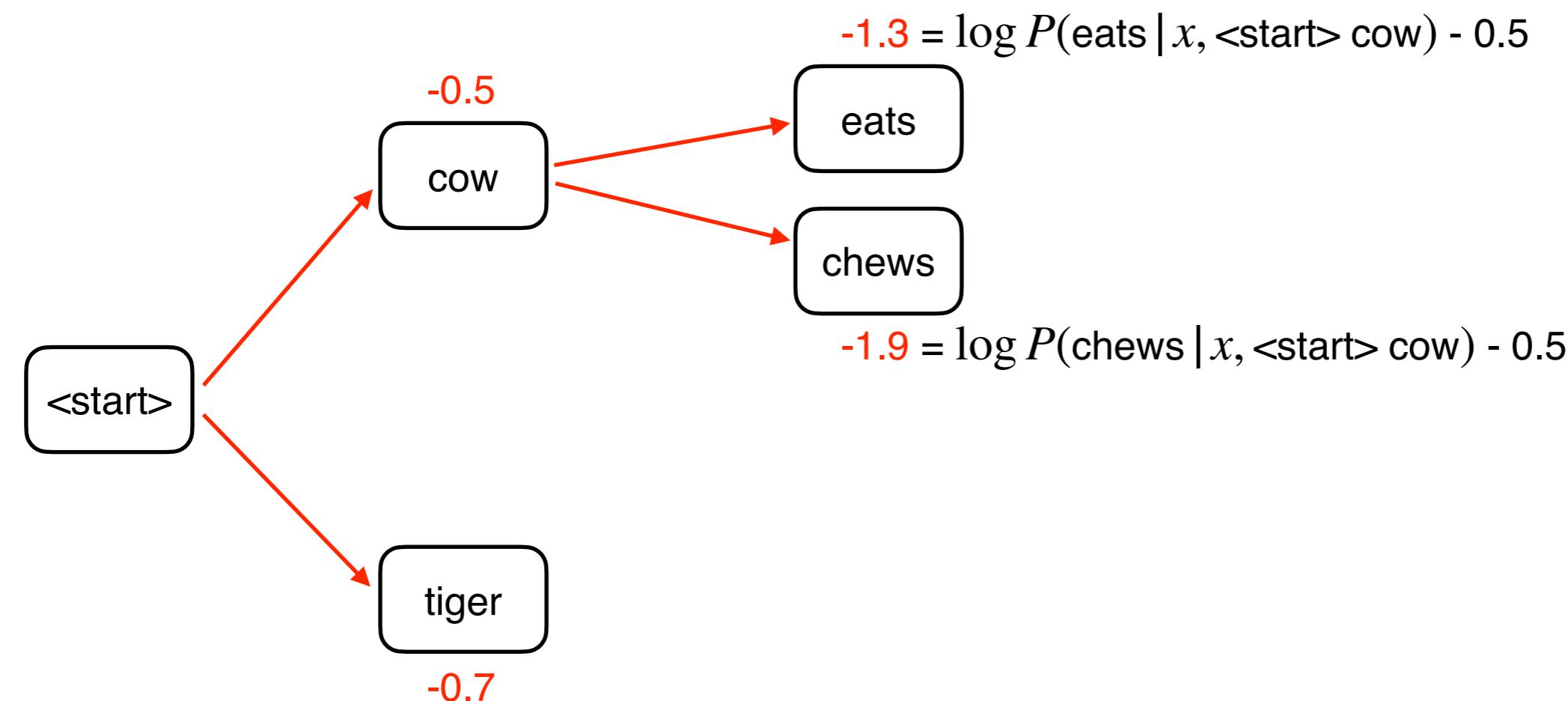
- To find optimal $P(y | x)$, we need to consider every word at every step to compute the probability of all possible sequences
- $O(V^n)$; $V = \text{vocab size}$; $n = \text{sentence length}$
- Far too expensive to be feasible

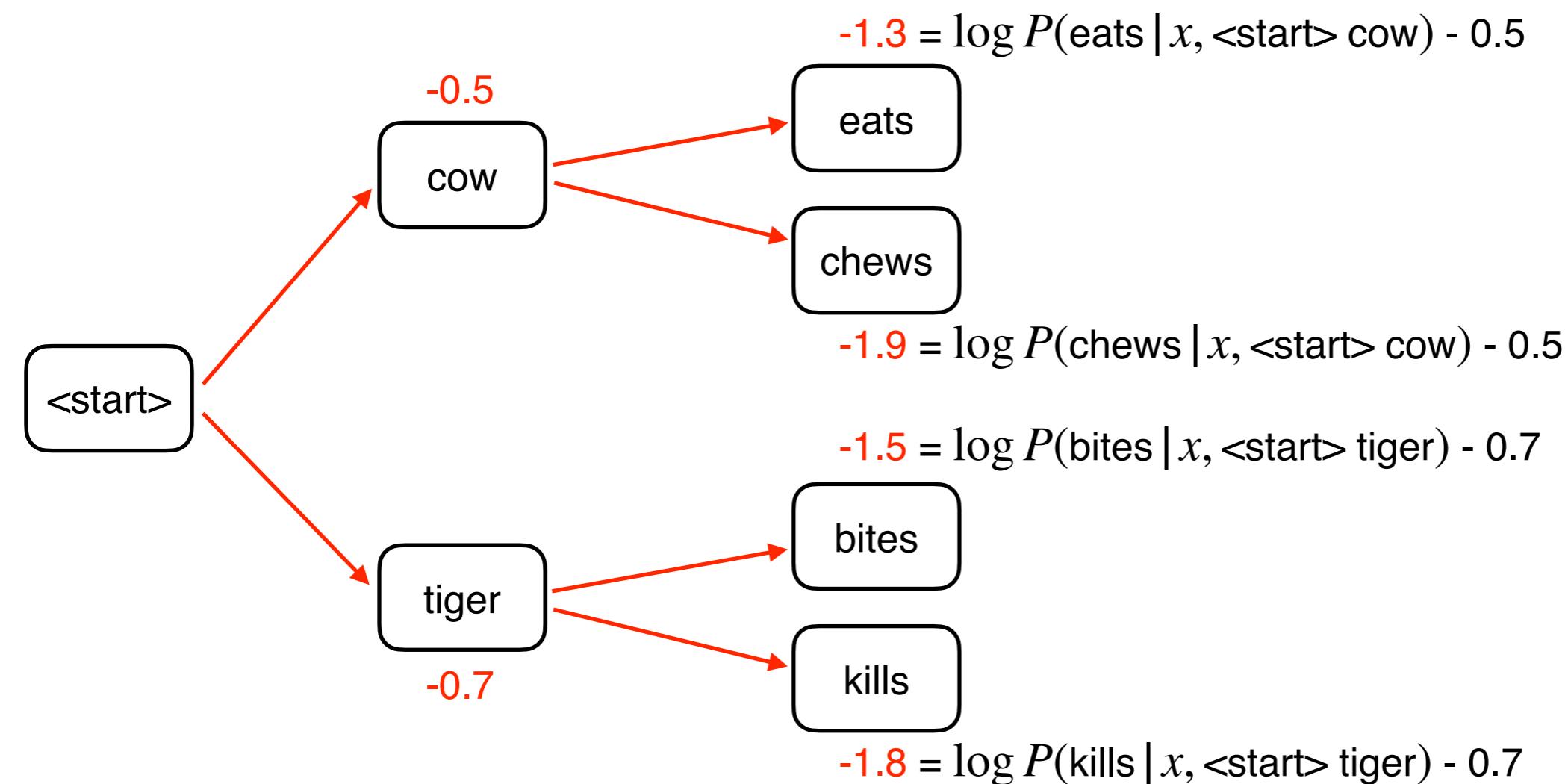
Beam Search Decoding

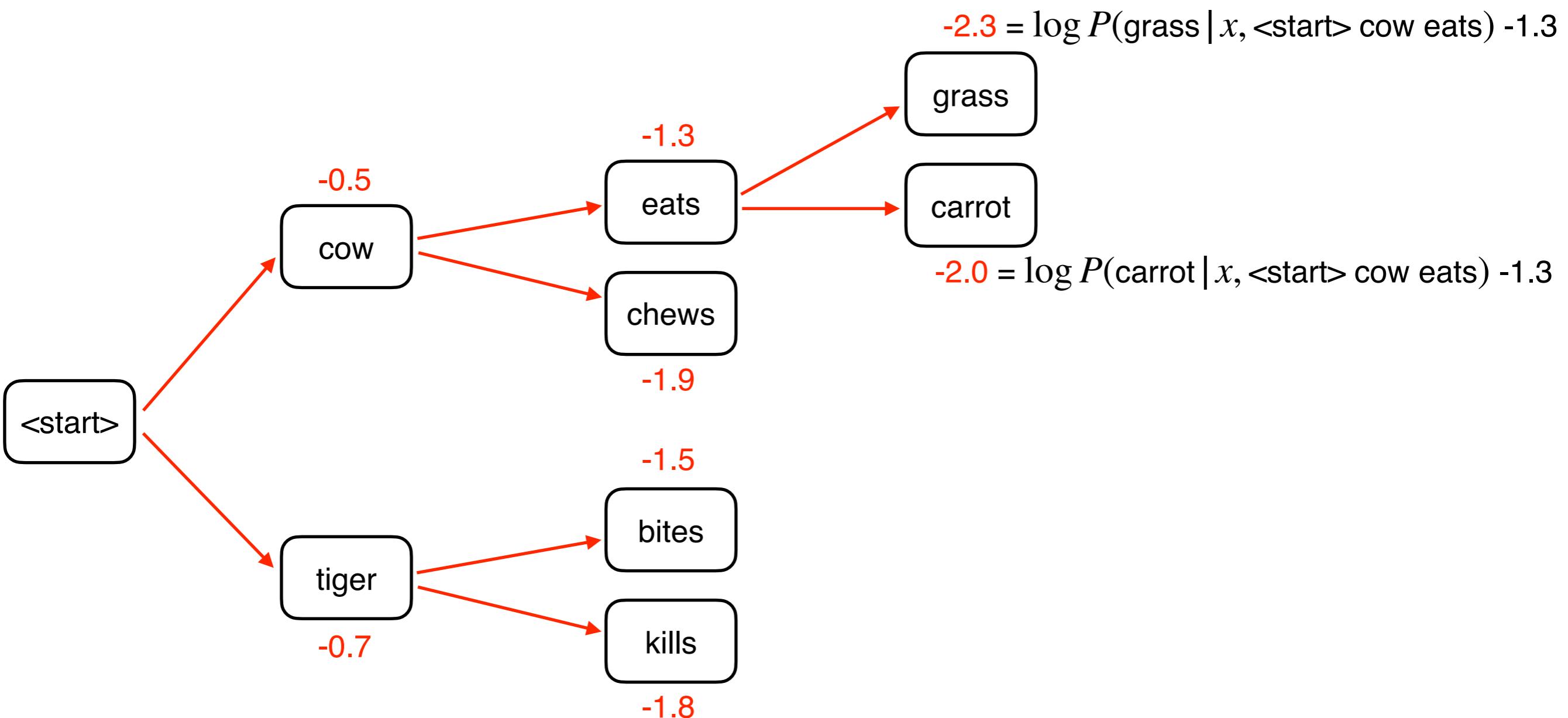
- Instead of considering all possible words at every step, consider **k best words**
- That is, we keep track of the top-k words that produce the best partial translations (**hypotheses**) thus far
- $k = \text{beam width}$ (typically 5 to 10)
- $k = 1 = \text{greedy decoding}$
- $k = V = \text{exhaustive search decoding}$

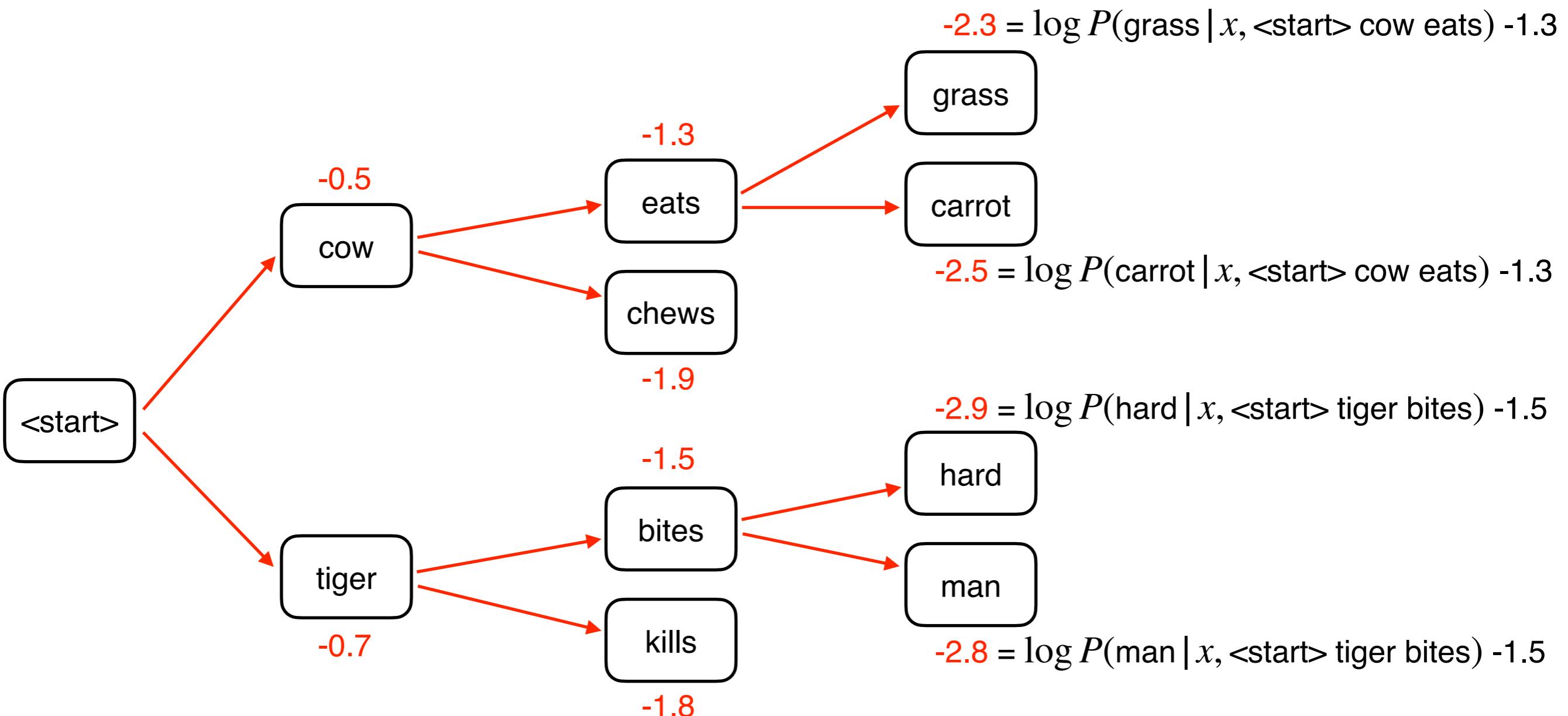
<start>



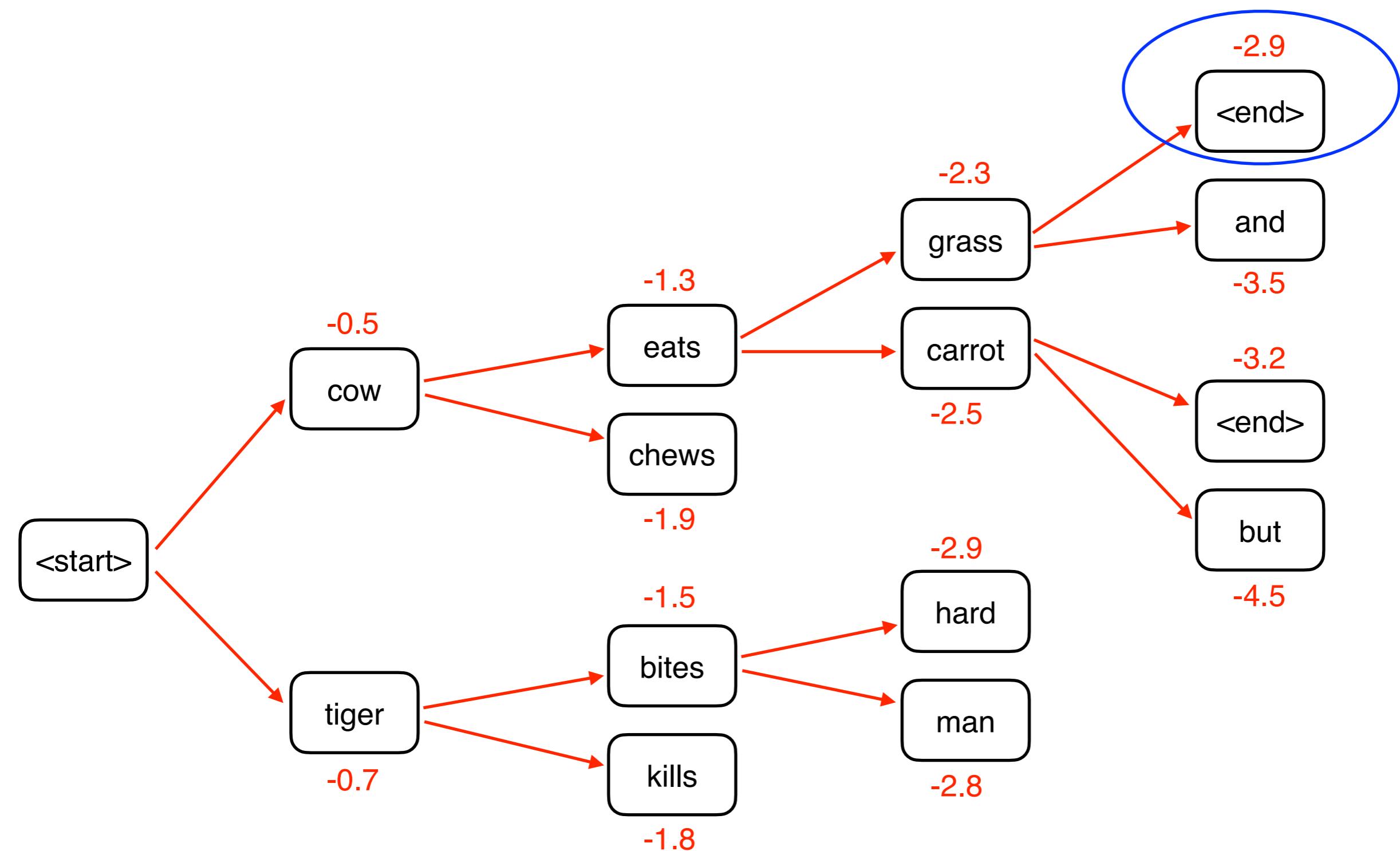






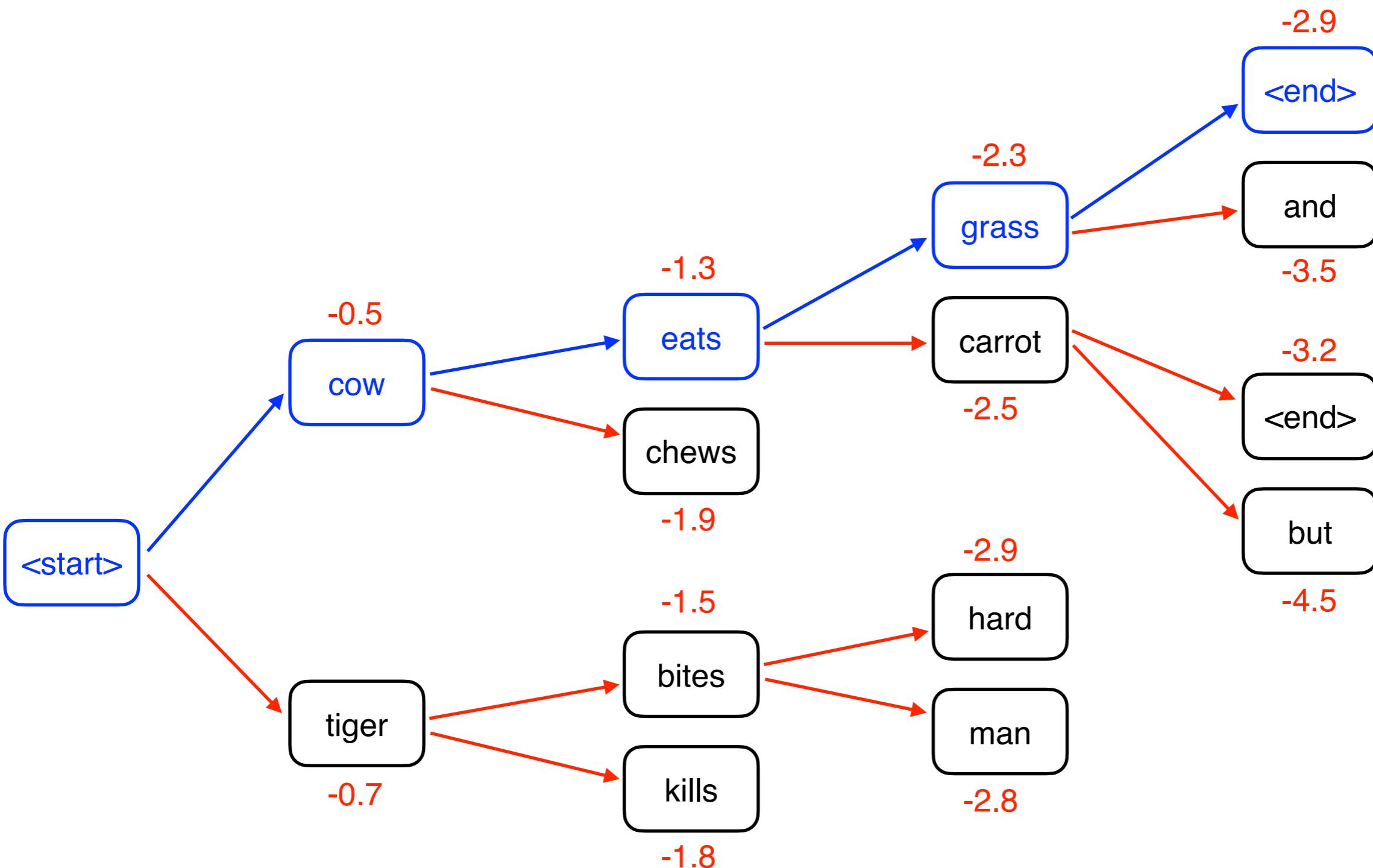


Best translation probability!



Beam search decoding with $k=2$

Follow pointers to get the target sentence



Beam search decoding with $k=2$

When to Stop?

- When decoding, we stop when we generate `<end>` token
- But multiple hypotheses may terminate their sentence at different time steps
- We store hypotheses that have terminated, and continue explore those that haven't
- Typically we also set a maximum sentence length that can be generated (e.g. 50 words)

What are some issues of NMT?

- Information of the whole source sentence is represented by a single vector
- NMT can generate new details not in source sentence
- NMT tend to generate not very fluent sentences
- Black-box model; difficult to explain when it doesn't work

PollEv.com/jeyhanlau569



What are some issues of NMT

Information of the whole source sentence
is represented by a single vector

27%

NMT can generate new details not in
source sentence

27%

NMT tend to generate not very fluent
sentences

12%

Black-box model; difficult to explain
when it doesn't work

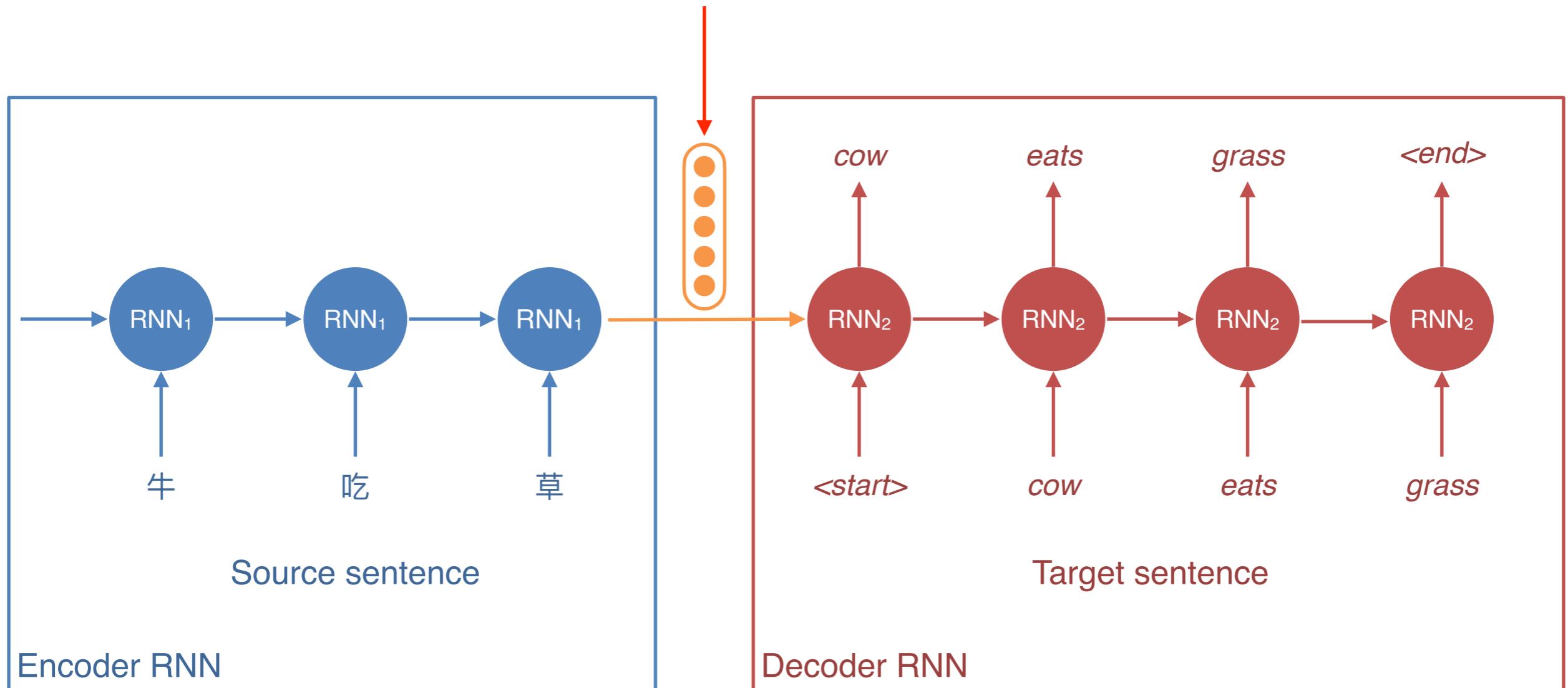
35%

Neural MT: Summary

- Single end-to-end model
 - ▶ Statistical MT systems have multiple sub-components
- Less feature engineering
- Can produce new details that are not in the source sentence (*hallucination*)

Attention Mechanism

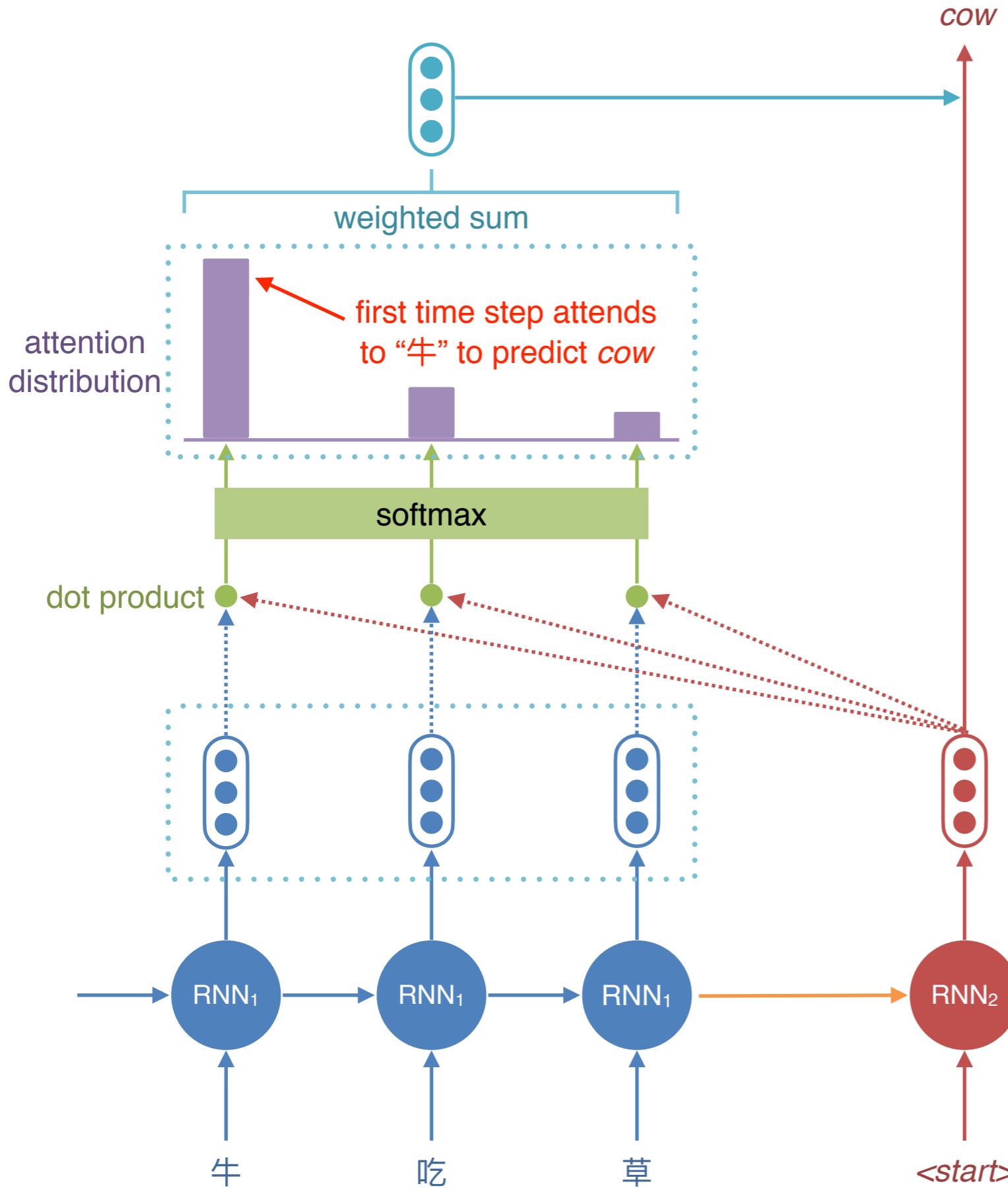
This vector encodes the whole source sentence!

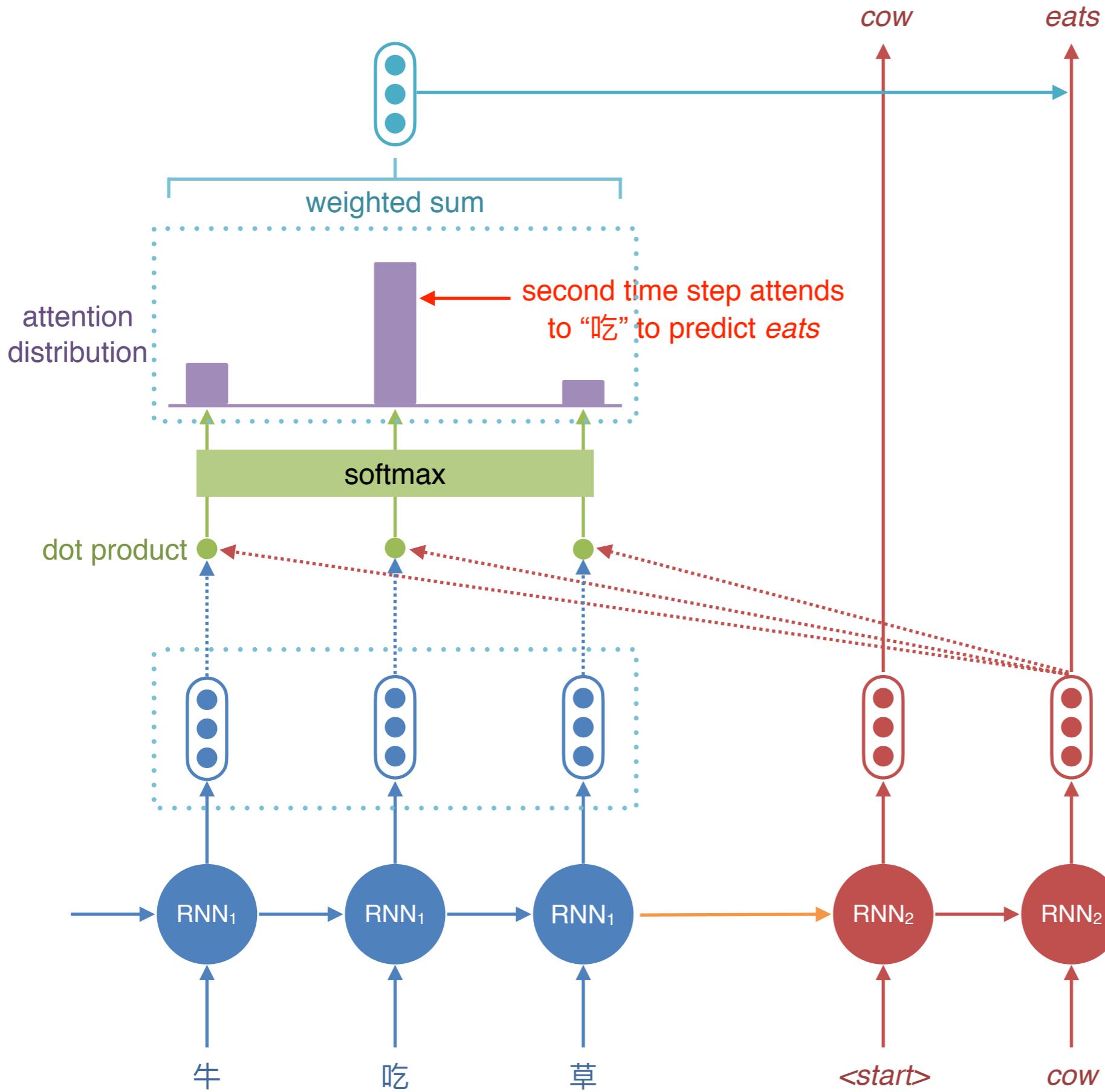


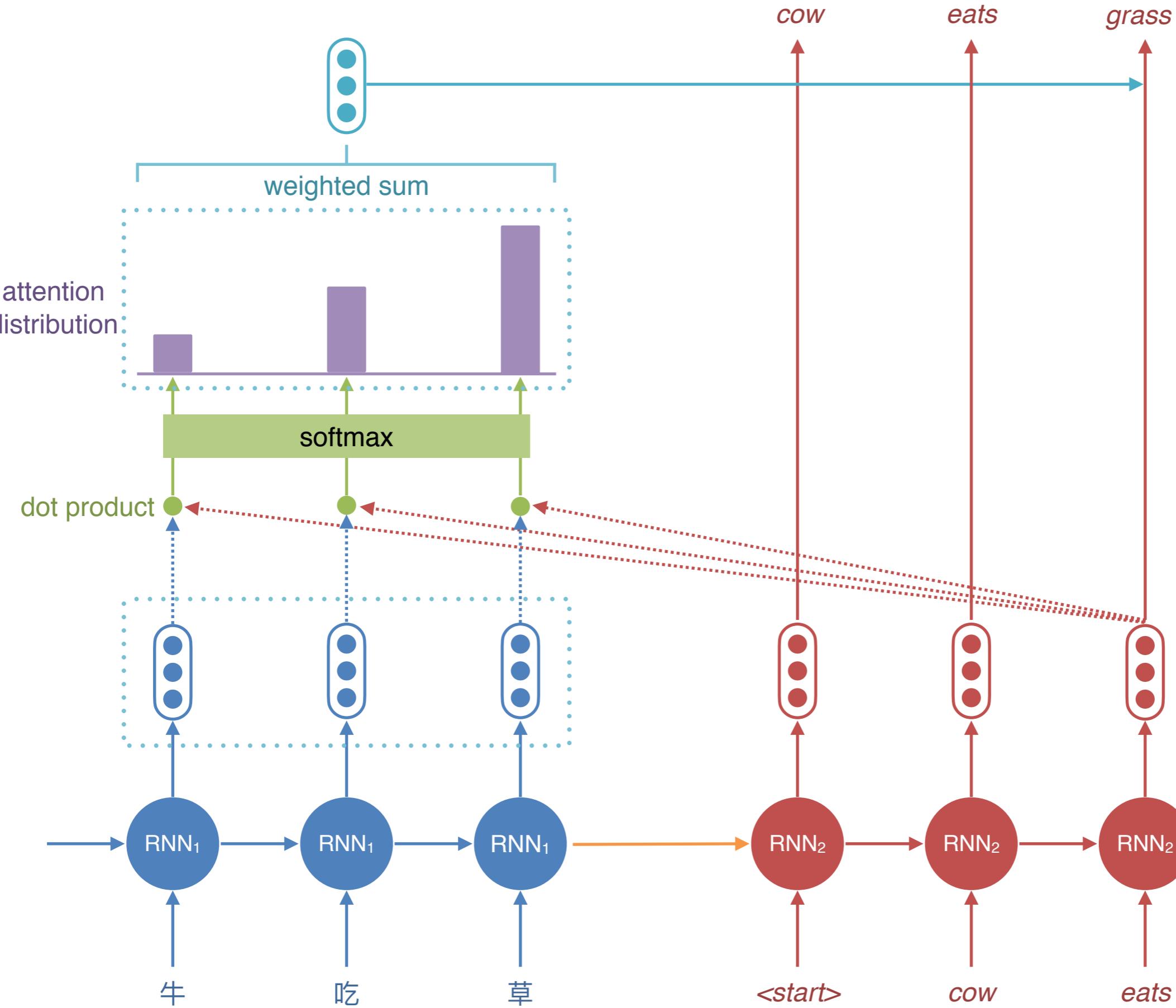
- With a long source sentence, the **encoded vector** is unlikely to capture all the information in the sentence
- This creates an **information bottleneck**

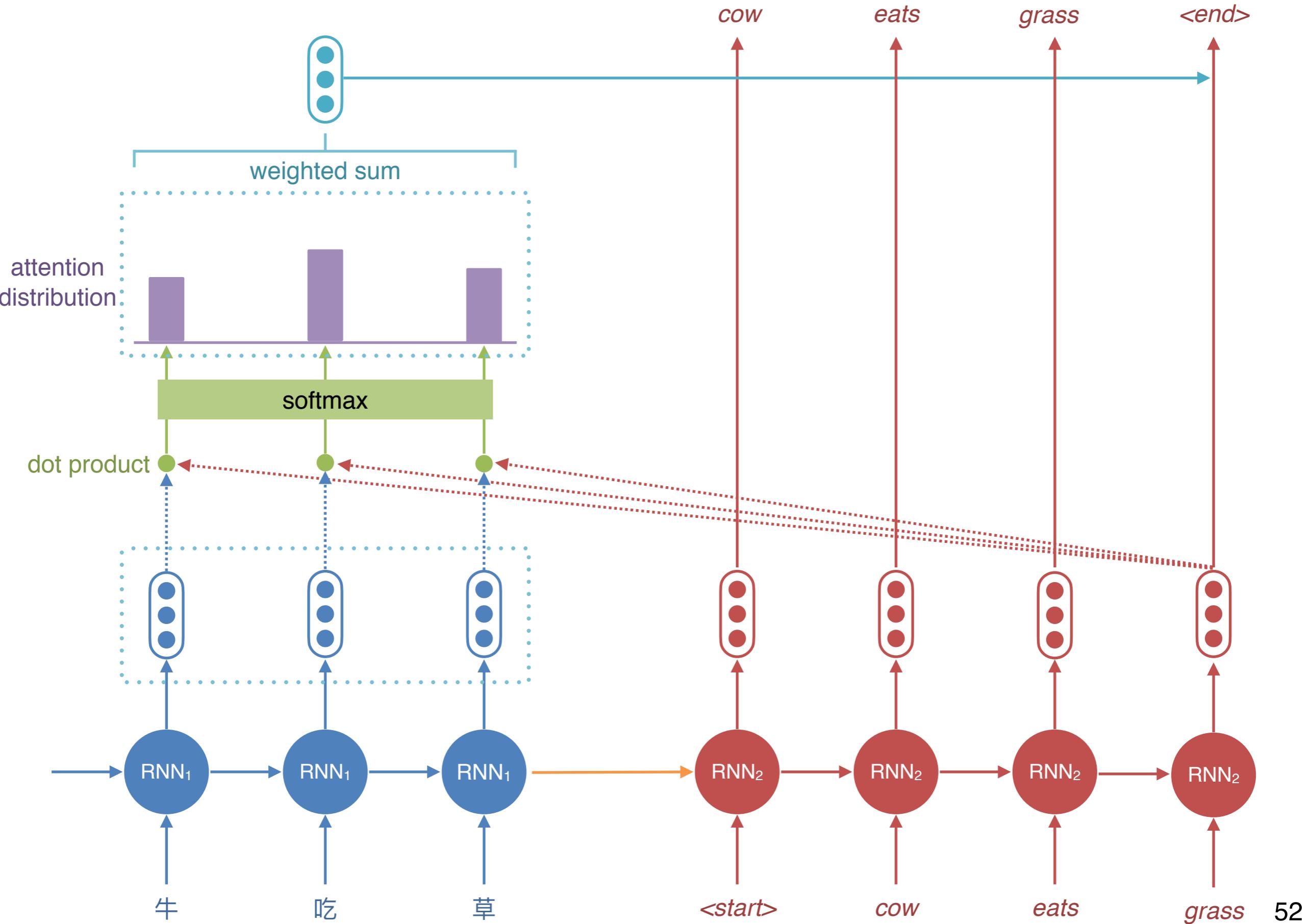
Attention

- For the decoder, at every time step allow it to ‘attend’ to words in the source sentence







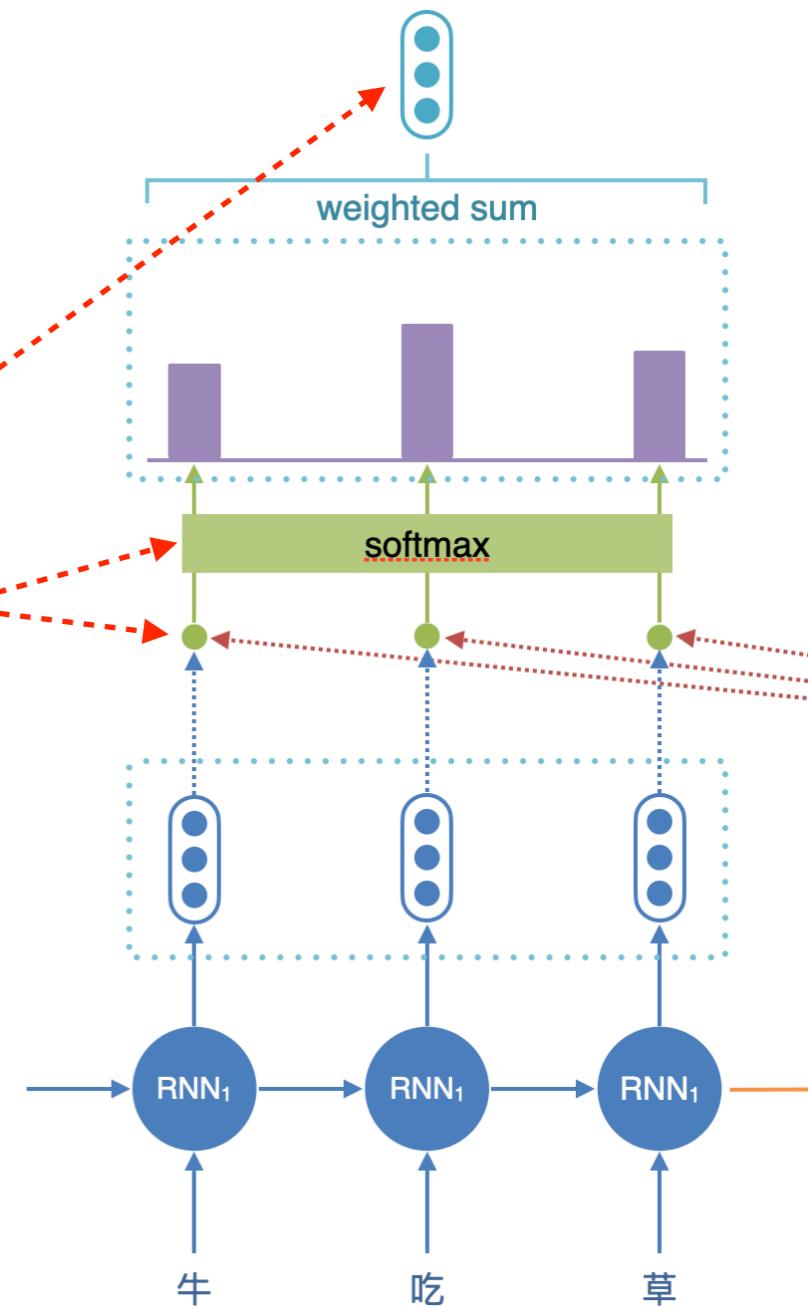


Encoder-Decoder with Attention

- Encoder hidden states: $h_i = RNN_1(h_{i-1}, x_i)$
- Decoder hidden states: $s_t = RNN_2(s_{t-1}, y_t)$
- For each time step in the decoder, attend to each of the hidden states to produce the attention weights:
 - $e_t = [s_t^\top h_1, s_t^\top h_2, \dots, s_t^\top h_{|x|}]$
- Apply softmax to the attention weights to get a valid probability distribution:
 - $\alpha_t = \text{softmax}(e_t)$
- Compute weighted sum of the encoder hidden states:

$$c_t = \sum_{i=1}^{|x|} \alpha_t^i h_i$$
- Concatenate c_t and s_t to predict the next word

context vector



Variants

- Attention:
 - ▶ Dot product: $s_t^\top h_i$
 - ▶ Bilinear: $s_t^\top Wh_i$
 - ▶ Additive: $v^\top \tanh(W_s s_t + W_h h_i)$
- c_t can be injected to the current state (s_t), or to the input word (y_t)

Attention: Summary

- Solves the information bottleneck issue by allowing decoder to have access to the source sentence words directly
- Provides some form of interpretability
 - ▶ Attention weights can be seen as word alignments
- Most state-of-the-art NMT systems use attention
 - ▶ Google Translate (<https://slator.com/technology/google-facebook-amazon-neural-machine-translation-just-had-its-busiest-month-ever/>)

Evaluation

MT Evaluation

- BLEU: compute n-gram overlap between “reference” translation and generated translation
- Typically computed for 1 to 4-gram

$$\text{BLEU} = \text{BP} \times \exp \left(\frac{1}{N} \sum_n^N \log p_n \right)$$

$$p_n = \frac{\# \text{ correct n-grams}}{\# \text{ predicted n-grams}}$$

“Brevity Penalty” to penalise short outputs

$$\text{BP} = \min \left(1, \frac{\text{output length}}{\text{reference length}} \right)$$

A Final Word

- Statistical MT
- Neural MT
 - ▶ Nowadays use Transformers rather than RNNs
- Encoder-decoder with attention architecture is a general architecture that can be used for other tasks
 - ▶ Summarisation (lecture 21)
 - ▶ Other generation tasks such as dialogue generation

Reading

- JM3 Ch. 11.2-11.5
- Optional: GIZA++ word alignment - <https://www.aclweb.org/anthology/W08-0509.pdf>