Machine Translation

COMP90042

Natural Language Processing

Lecture 17

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

MT is Difficult

Awkward.

- Not just simple word for word translation
- Structural changes, e.g., syntax and semantics
- Multiple word translations, idioms

- Inflections for gender, case etc European languages.
- Missing information (e.g., determiners)

Japanese.

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
  - Word-based approach
- Neural Machine Translation
  - Encoder-decoder architecture
- Attention Mechanism
  - Improvement to Neural MT
- Machine Translation 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

probabilistic Model

- Given French sentence *f*, aim is to find the best English sentence *e*
  - $\text{argmax}_e P(e|f)$ → french sentence → english sentence
- Use Baye's rule to decompose into two components
  - $\text{argmax}_e P(f|e)P(e)$

TM language M

BR.

$$P(e|f) = \frac{P(f|e)P(e)}{P(f)}$$

known.

Language vs Translation Model

- $\text{argmax}_e P(f|e)P(e)$
- $P(e)$: **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:
  - Based on text frequency in large **monolingual corpora**

English.

- Translation model:
  - Based on word co-occurrences in **parallel corpora**
  - i.e. English-French sentence pairs

Parallel Corpora

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

The Rosetta Stone

Models of Translation

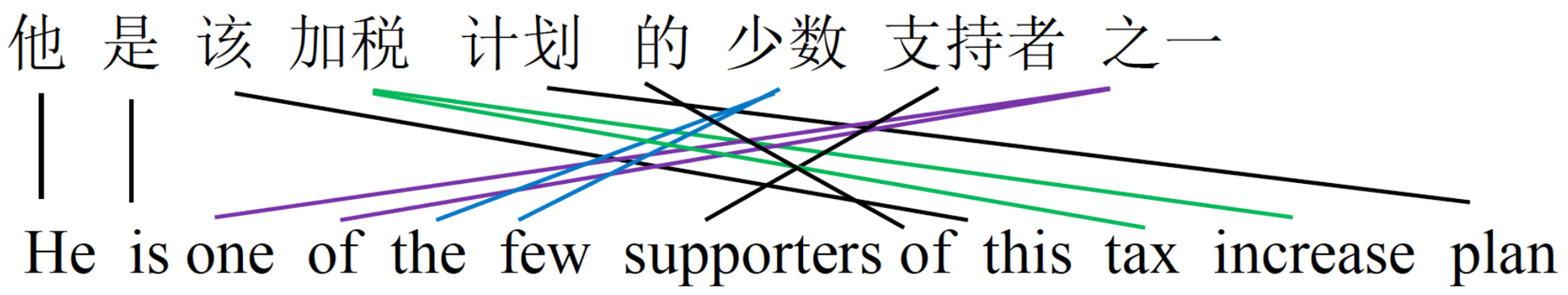
- How to learn $P(f \mid e)$ from parallel text?

Sentence pairs

- We only have sentence pairs; words are not aligned in the parallel text

no words alignment

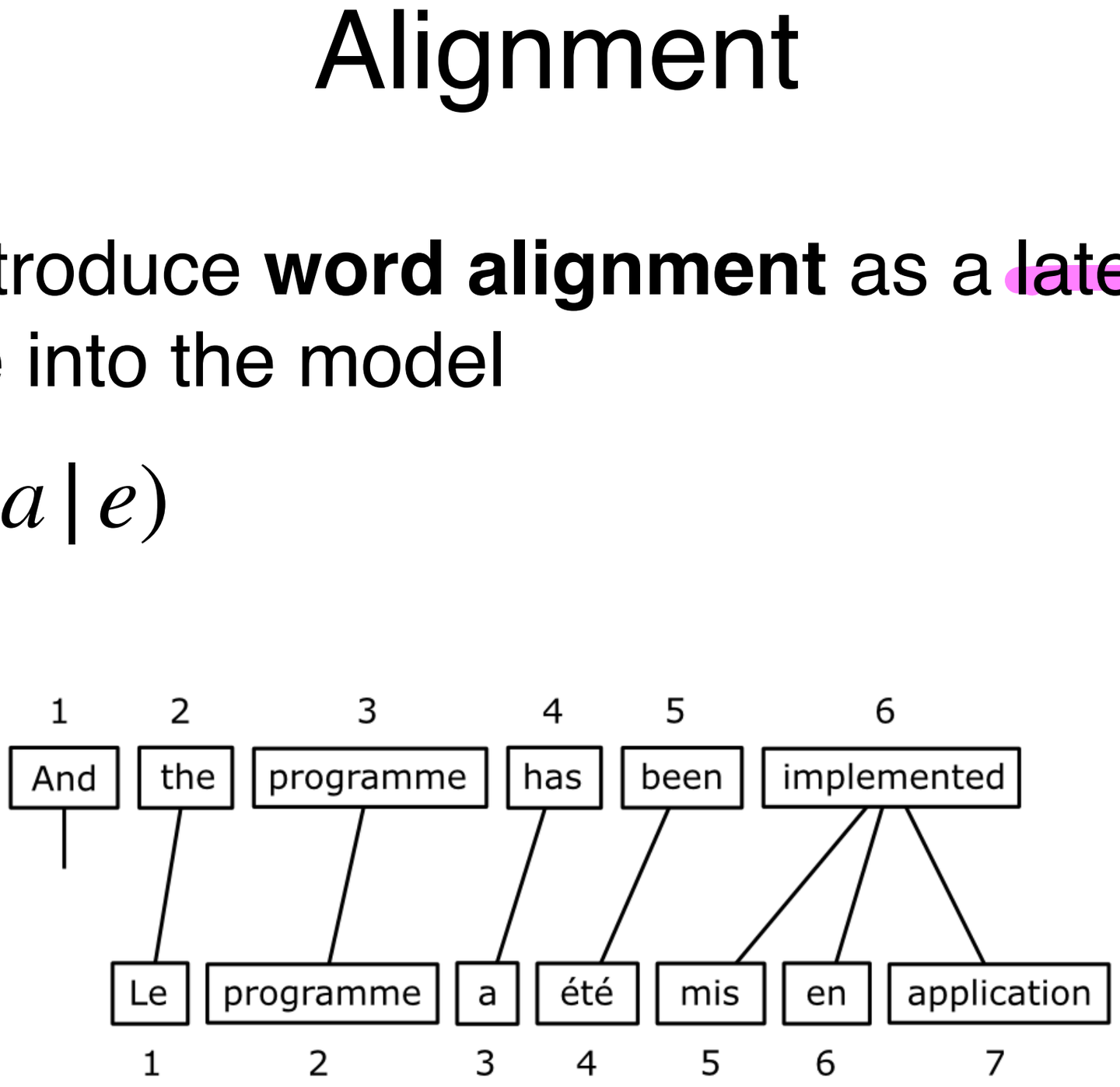
- Not word to word translation:
  - rearrangement of words



Zhang, Liu, Li, Zhou, Zong (2016)

Alignment

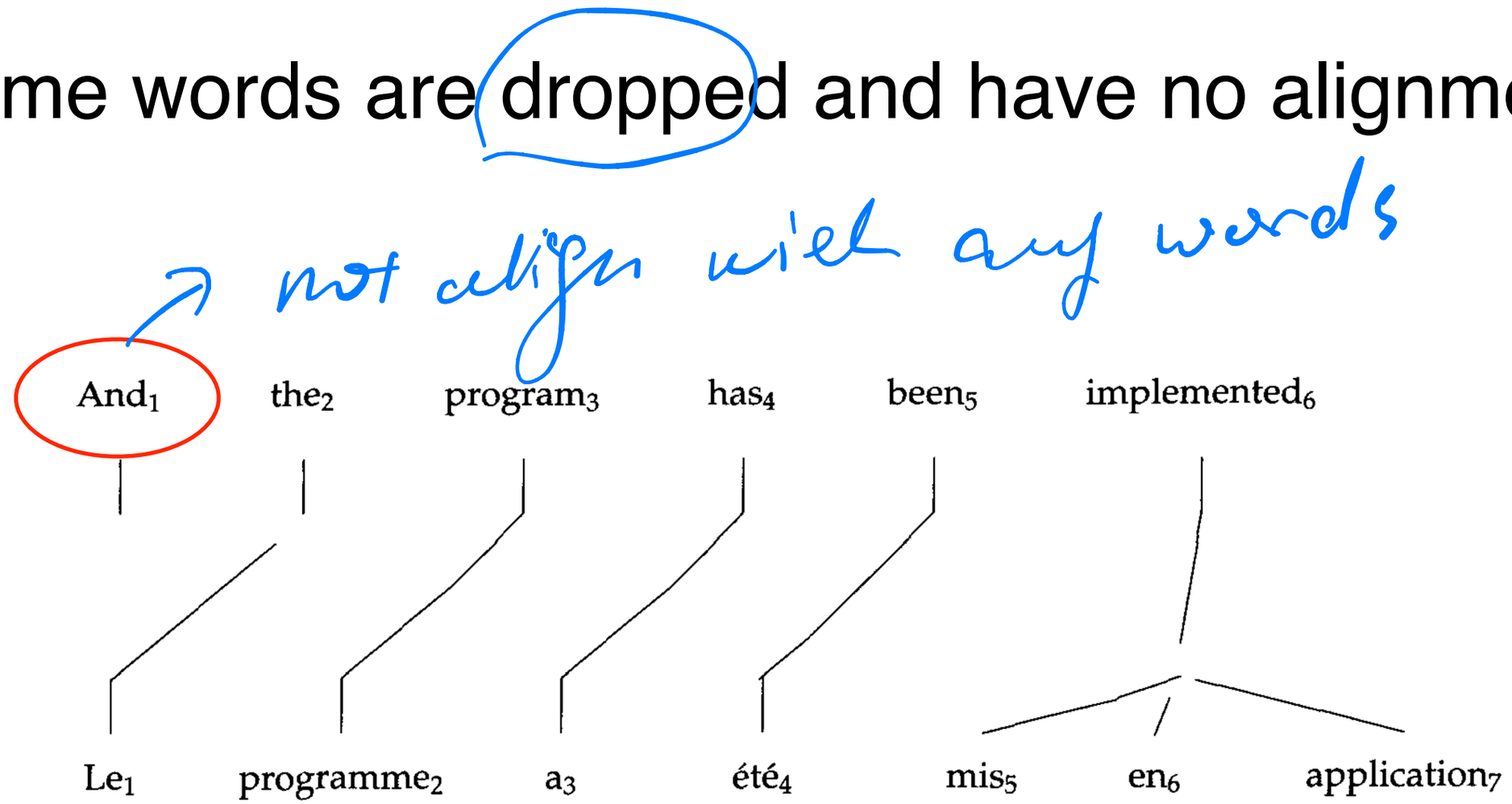
- Idea: introduce **word alignment** as a latent variable into the model
  - $P(f, a \mid e)$



https://en.wikipedia.org/wiki/Bitext_word_alignment#/media/File:Word_alignment.svg

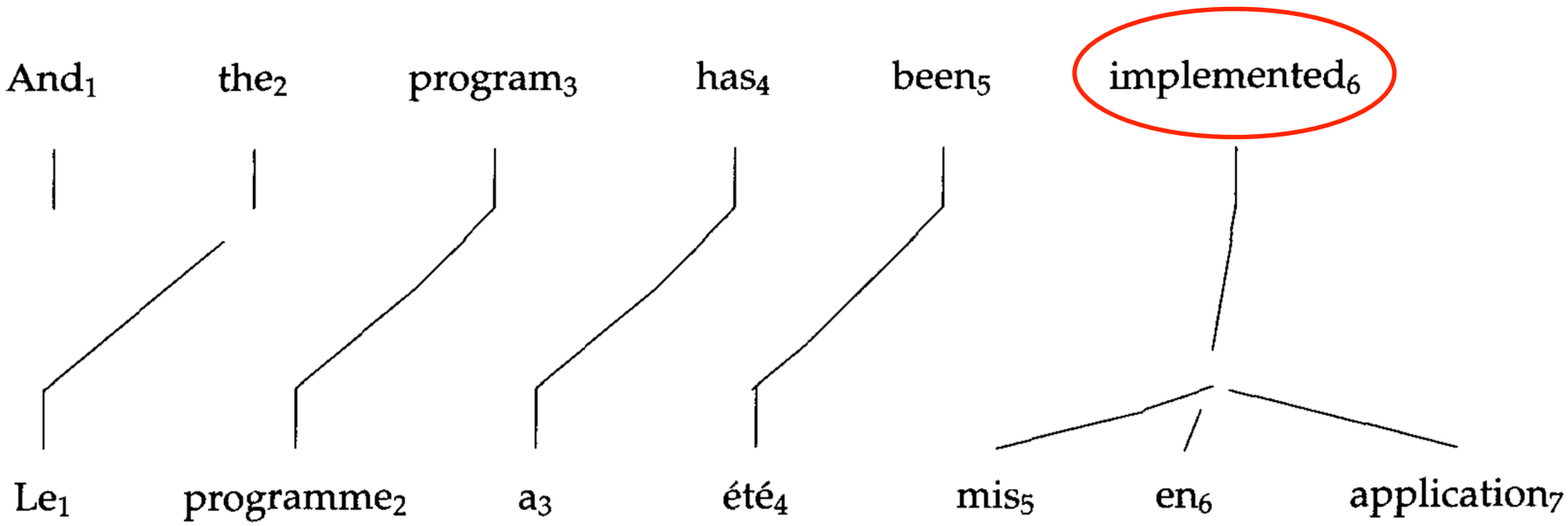
Complexity of Alignment

- Some words are dropped and have no alignment



Complexity of Alignment

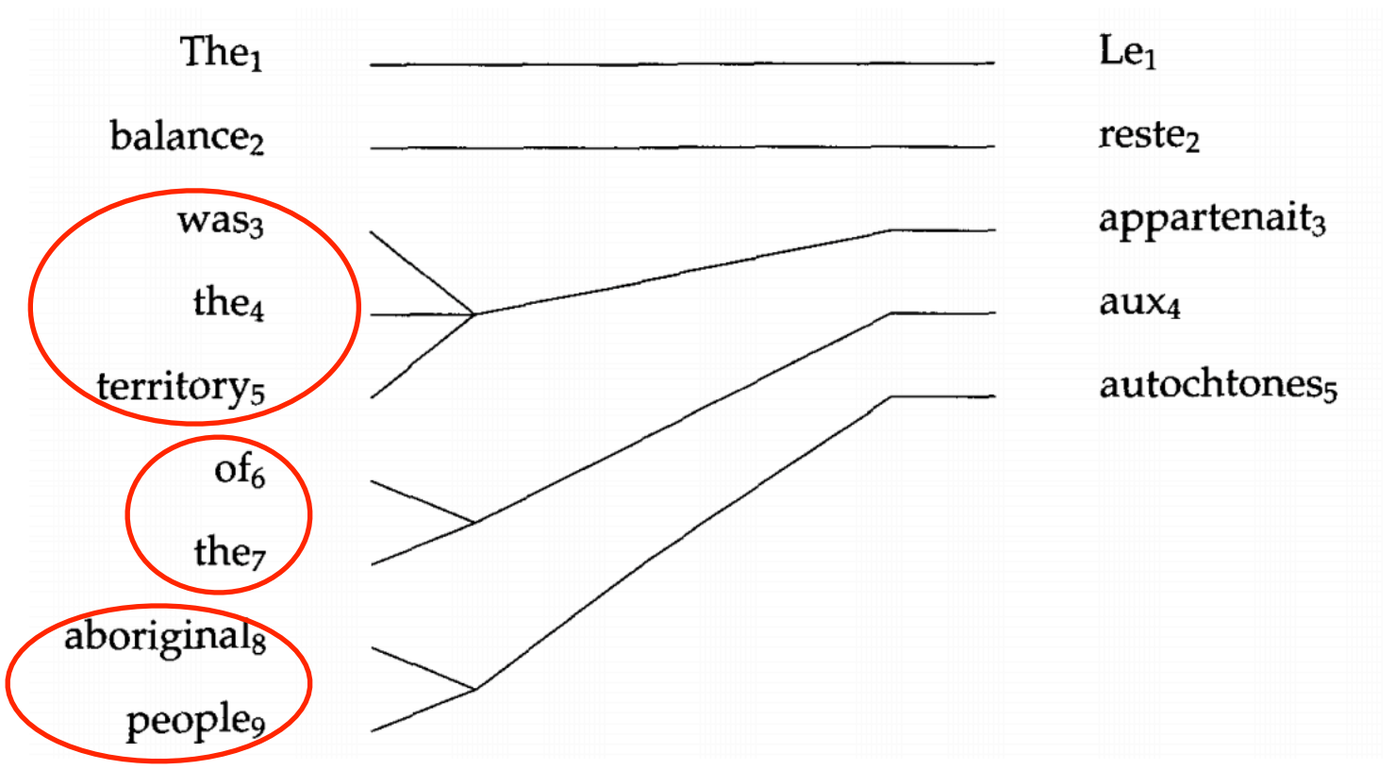
- One-to-many alignment



Brown, Della Pietra, Della Pietra, Mercer, 1993

Complexity of Alignment

- Many-to-one alignment

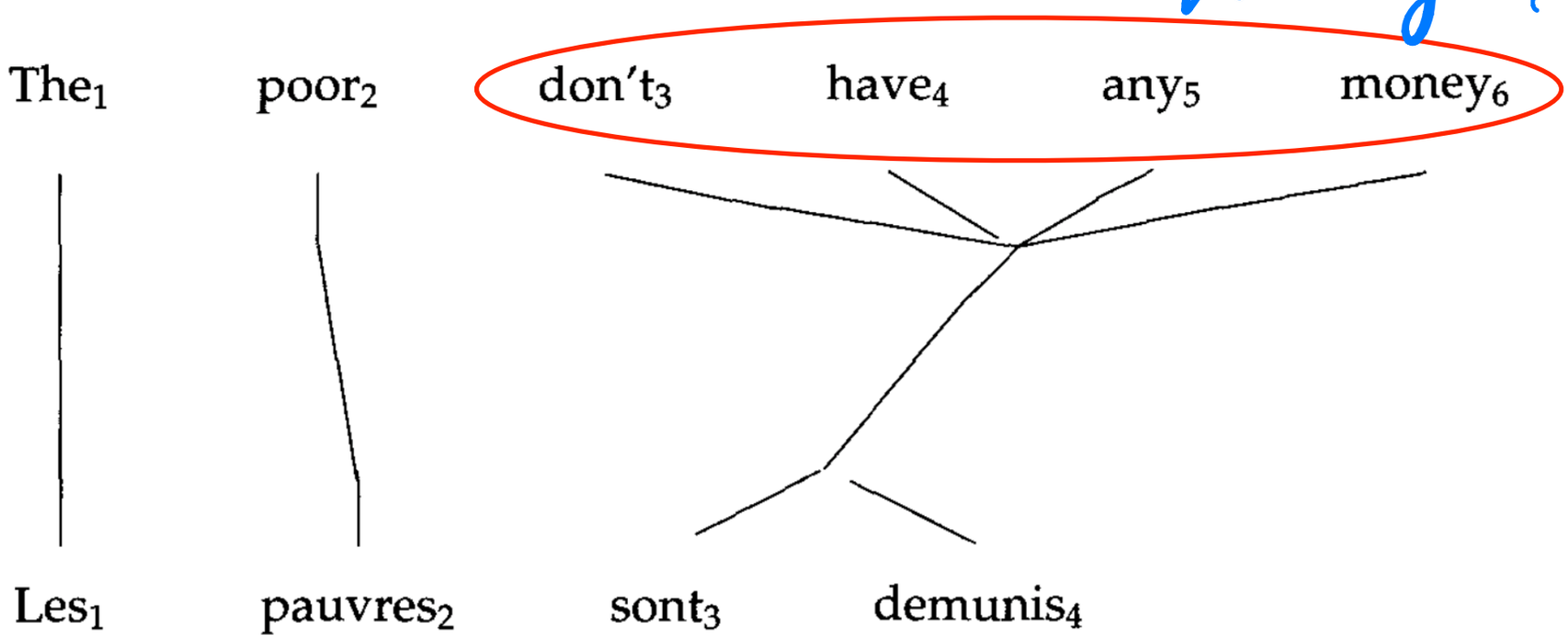


Brown, Della Pietra, Della Pietra, Mercer, 1993

Complexity of Alignment

- Many-to-many alignment

alignment types: drop, one-many, many-to-many, many-to-many



Learning the Alignment

- Word alignments are rarely provided in parallel corpora
  - As such alignment *a* introduced as a latent variable
- Use algorithms such as expectation maximisation (EM) to learn

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

Google Translation from Static → NN

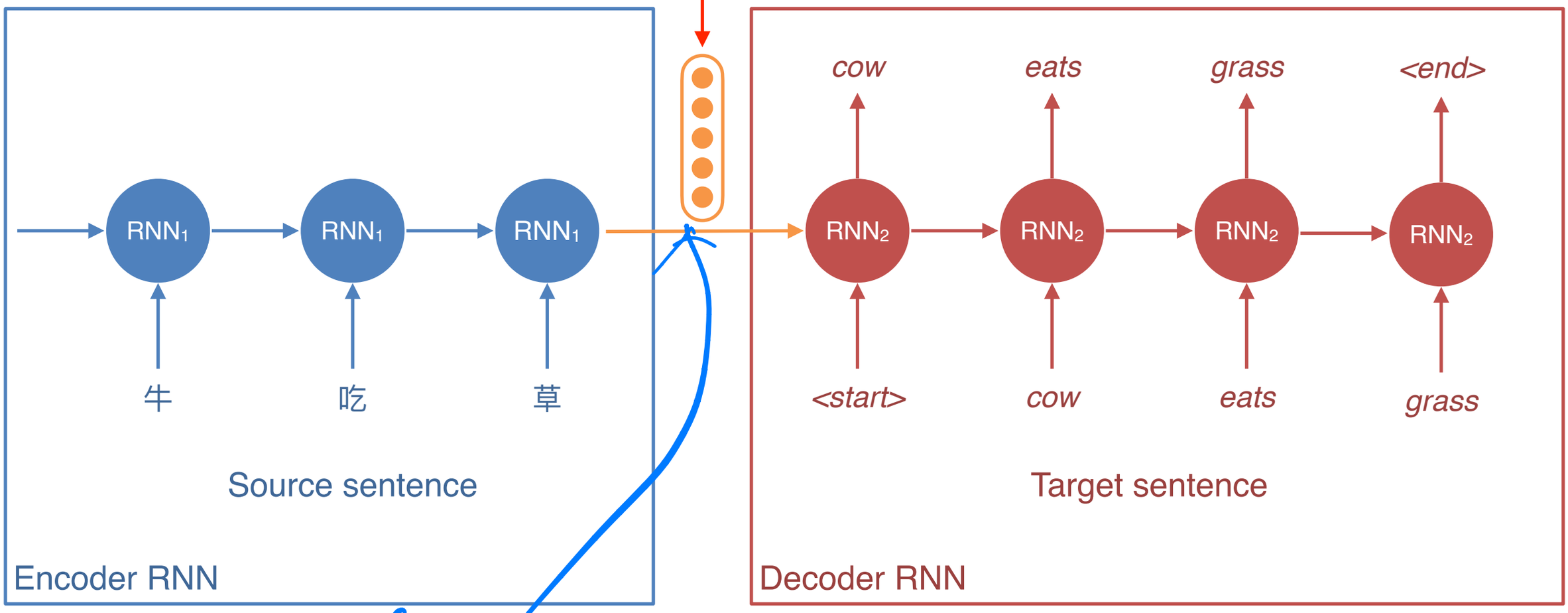
- Neural machine translation is a new approach to do machine translation
- Use a single neural model to directly translate from source to target

Seq → Seq Model

- Architecture: encoder-decoder model
  - 1st RNN to encode the source sentence
  - 2nd RNN to decode the target sentence

last hidden representation

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



$$h_i = RNN_1(h_{i-1}, x_i)$$
$$s_t = RNN_2(s_{t-1}, y_t)$$
$$s_1 = h_{|x|}$$

x denotes words in source sentences.

↓ previous state

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*
- $P(y|x) = P(y_1|x)P(y_2|y_1,x)\ldots P(y_t|y_1,\ldots,y_{t-1},x)$

Language Model with additional Condition

Conditioned on Source

Training Neural MT

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

train like LM

Language Model Training Loss

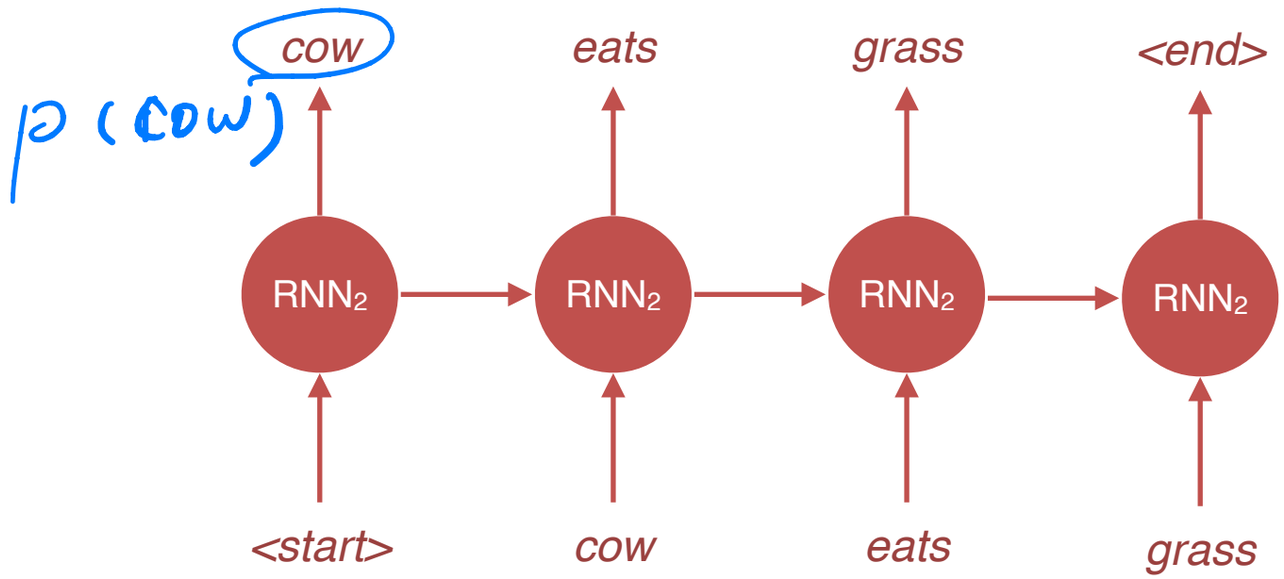
cross entropy

negative logprob of *cow* | negative logprob of *eats* | negative logprob of *grass* | negative logprob of *<end>*

total loss function

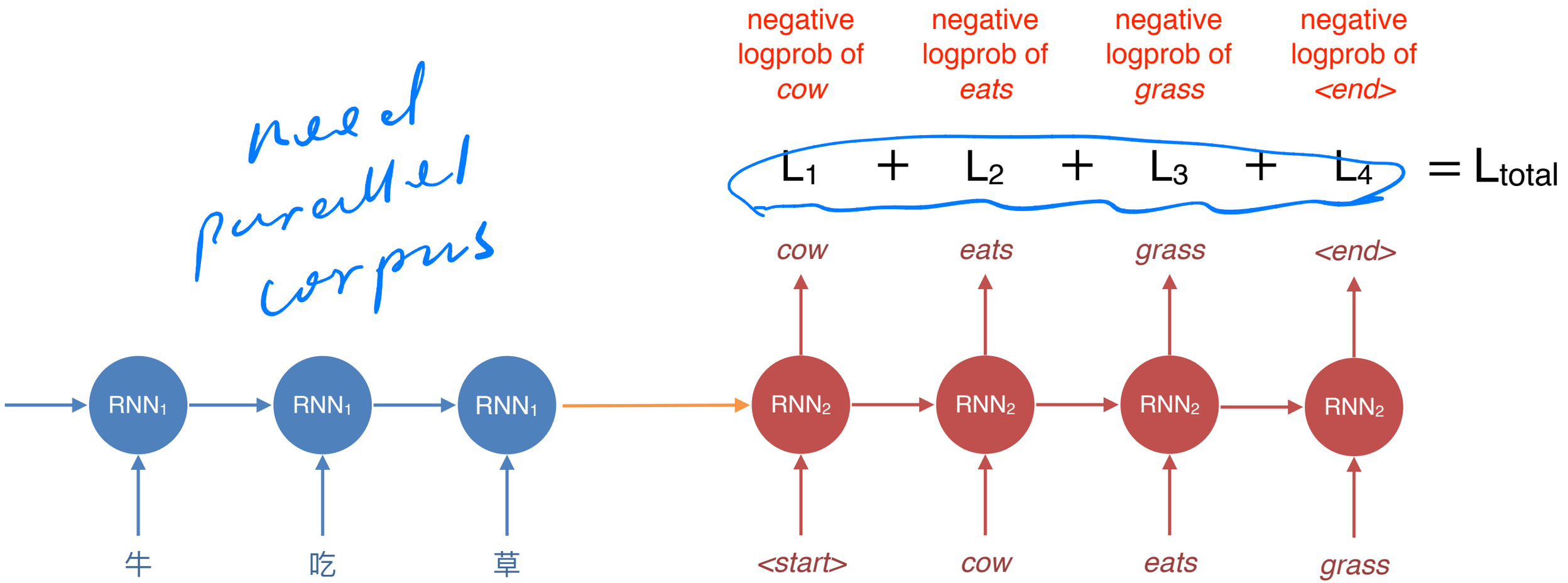
$$L_1 + L_2 + L_3 + L_4 = L_{total}$$

p(cow)

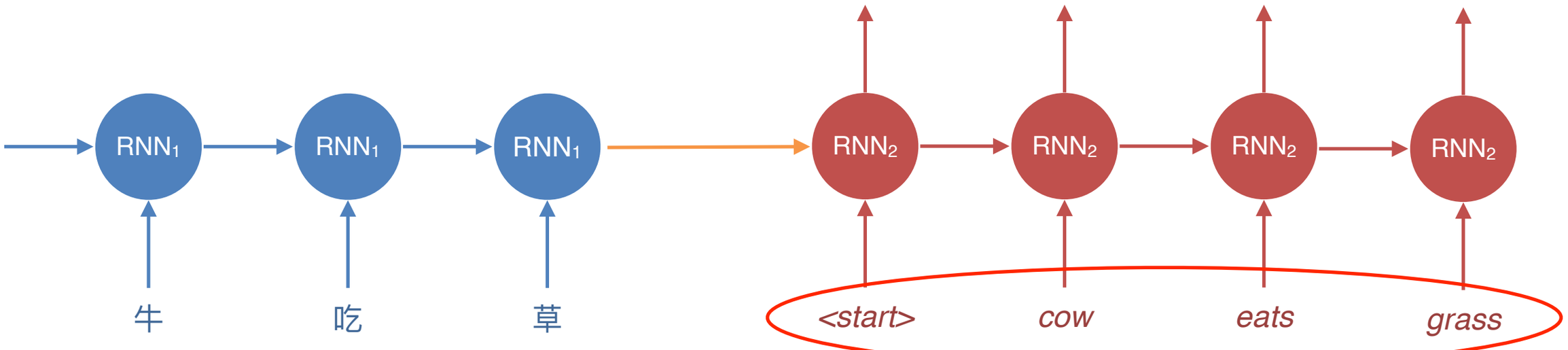


Neural MT Training Loss

need parallel corpus

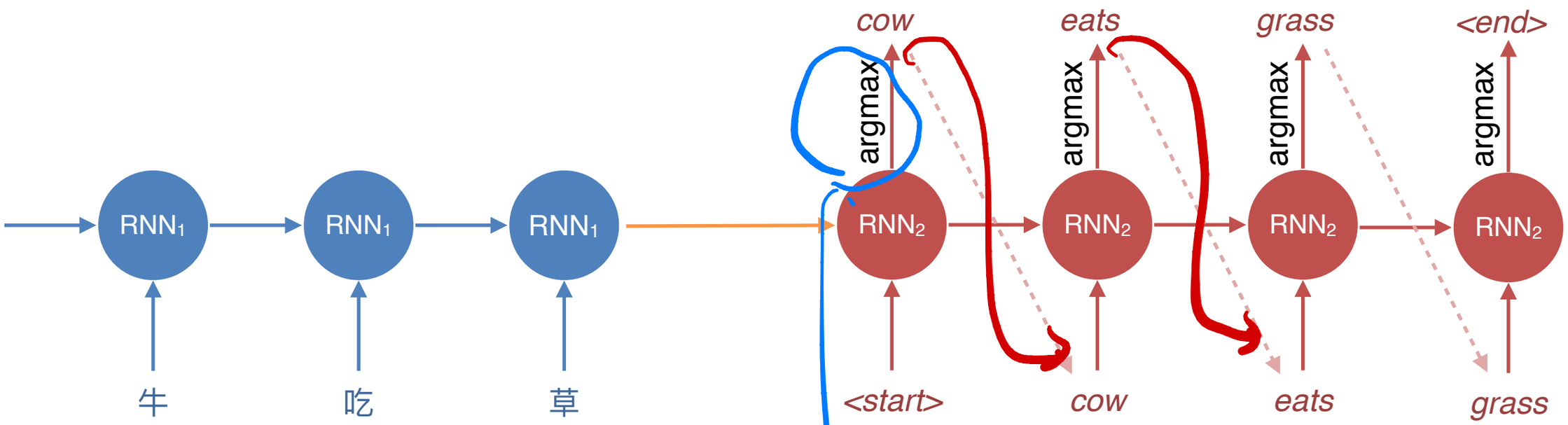


Training



- 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

Greedy Decoding

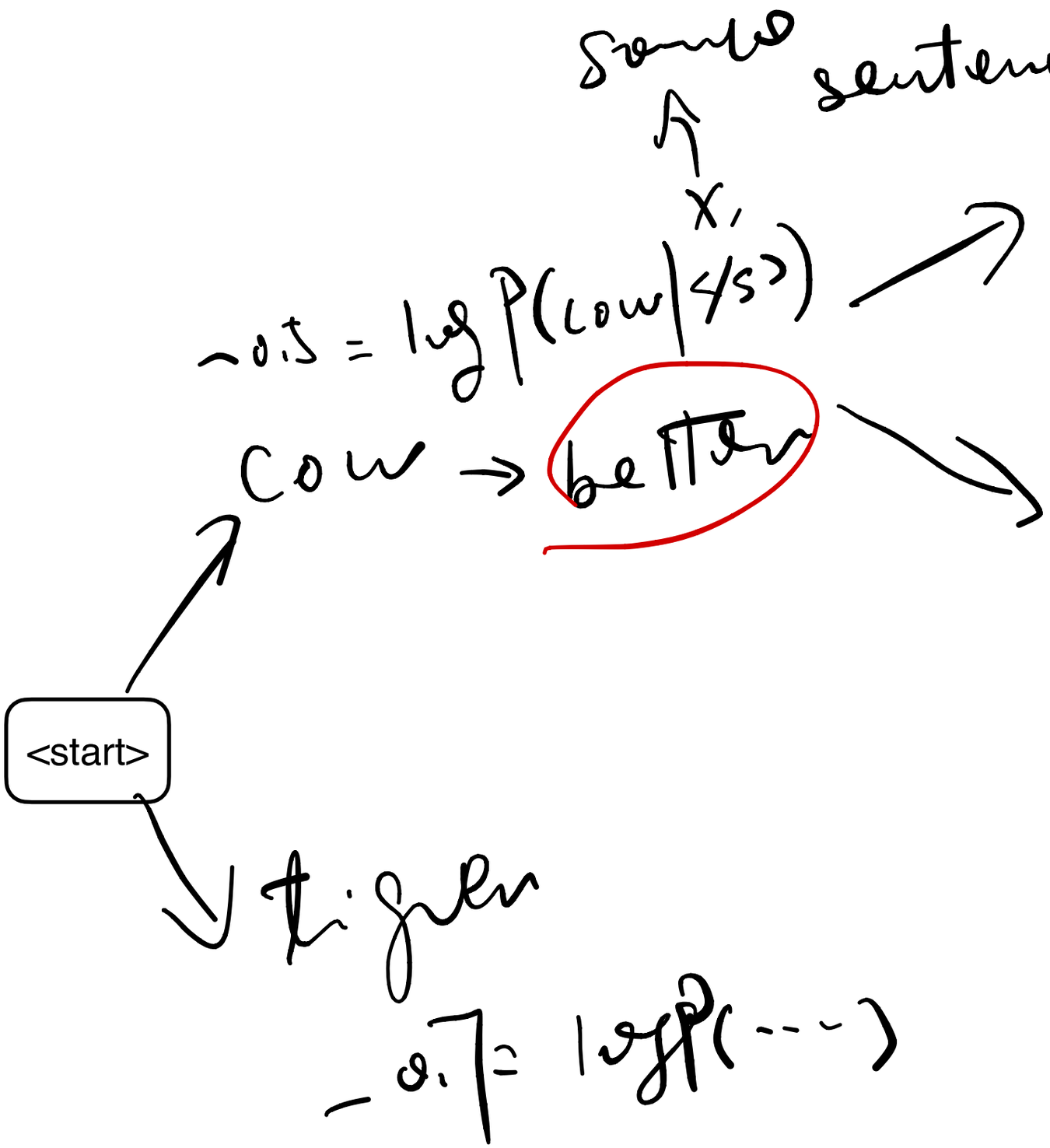
- argmax decoding is also called greedy decoding
- Issue: does not guarantee optimal probability $P(y \mid x)$

Exhaustive Search Decoding

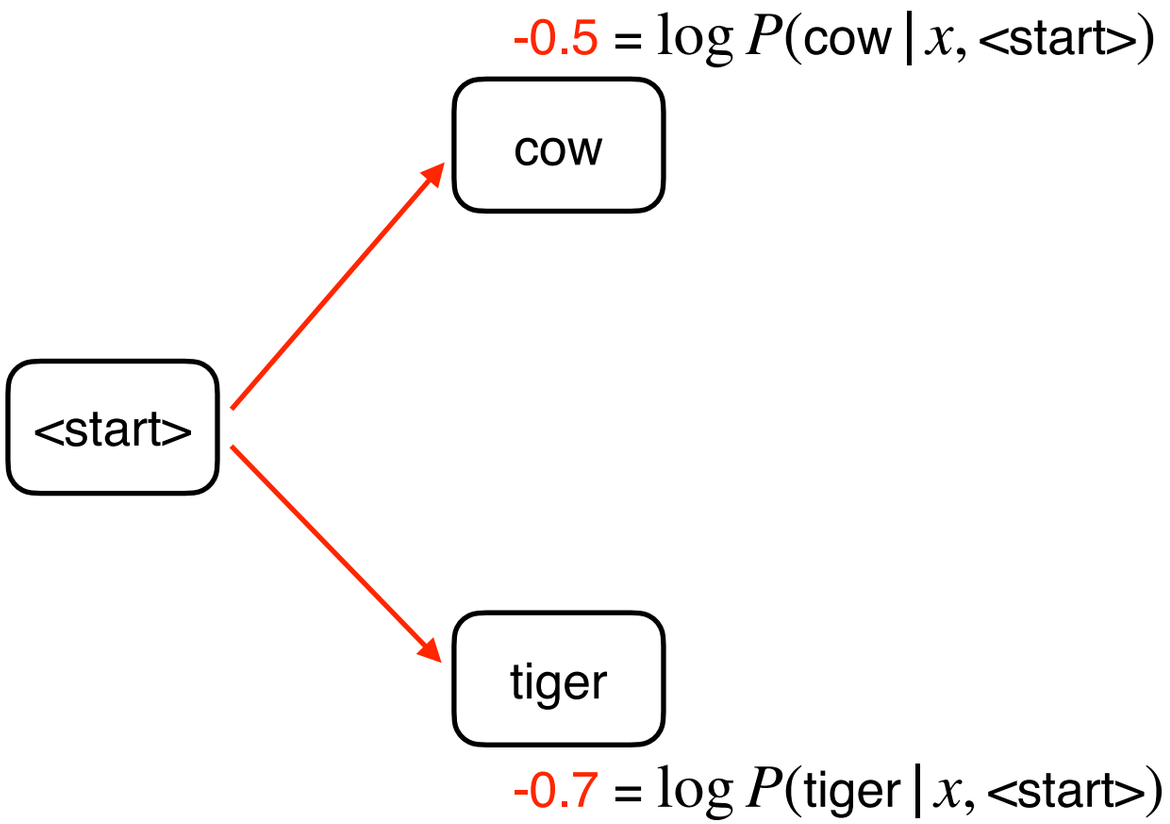
- To find optimal $P(y \mid x)$, we need to consider every word at every step to compute the probability of all possible sequences
- $O(V^n)$; V = vocab size; n = sentence length
- Far too expensive to be feasible

Beam Search Decoding

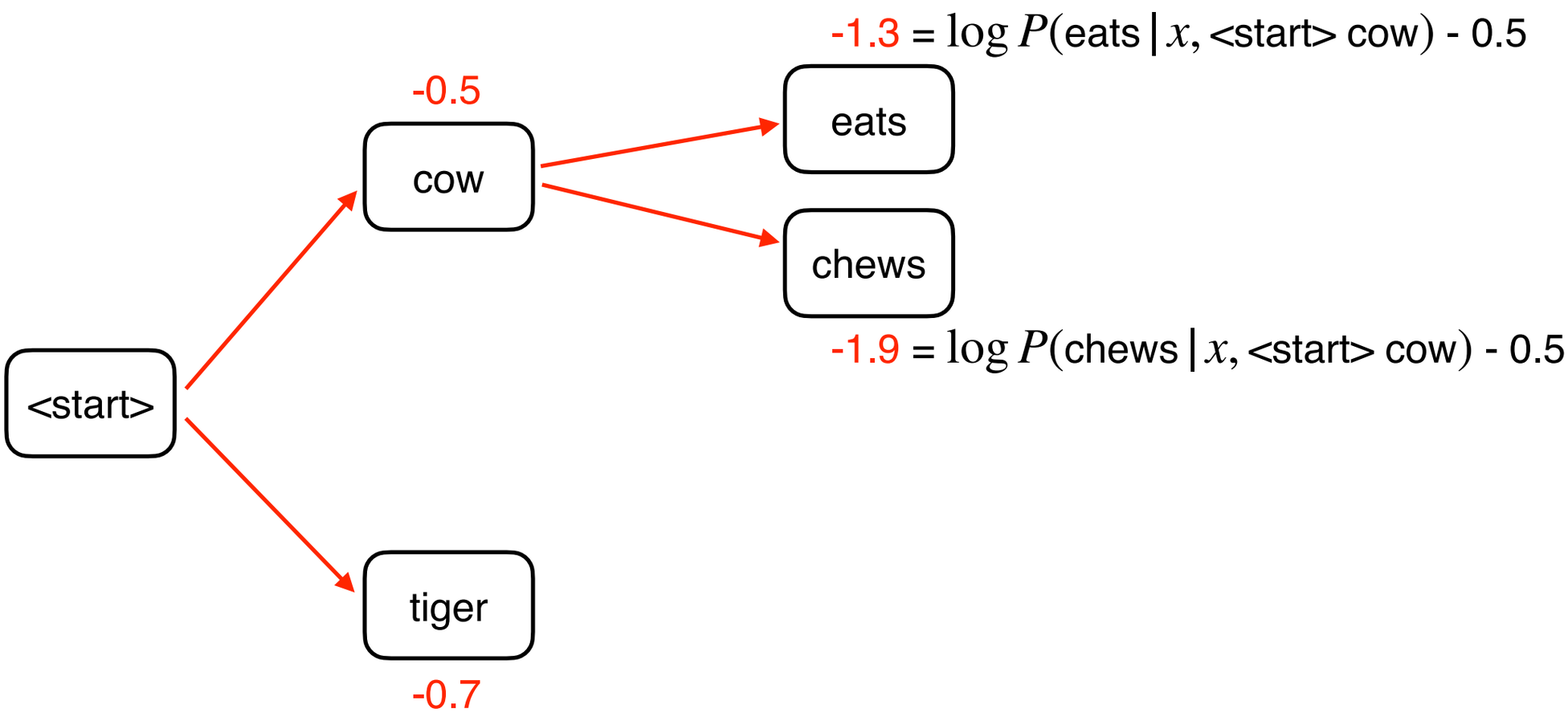
- Instead of considering all possible words at every step, consider **k best words**
- That is, we keep track of the top-k words that produce the best partial translations **(hypotheses)** thus far
- k = beam width (typically 5 to 10)
- k = 1 = greedy decoding
- k = V = exhaustive search decoding



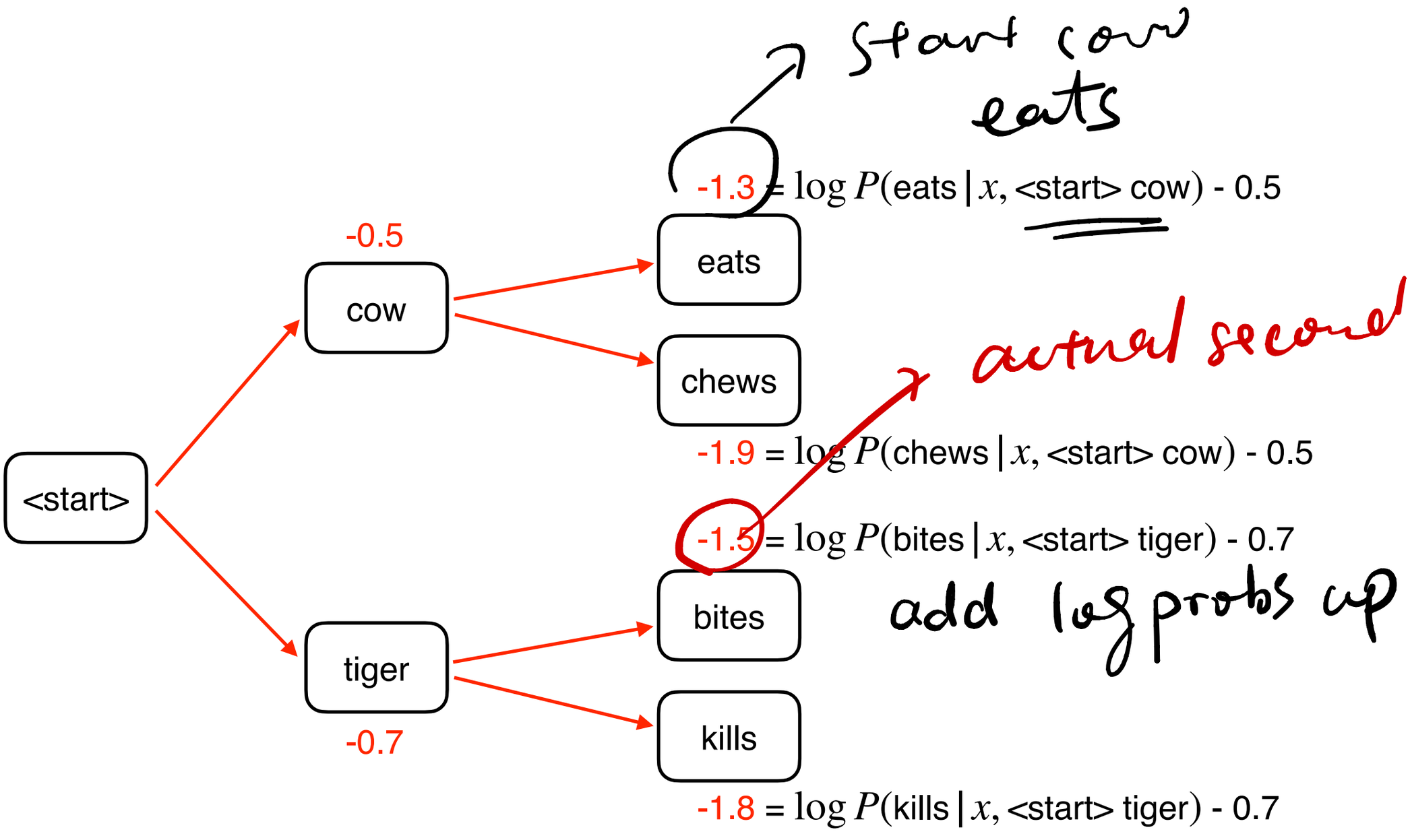
Beam search decoding with k=2



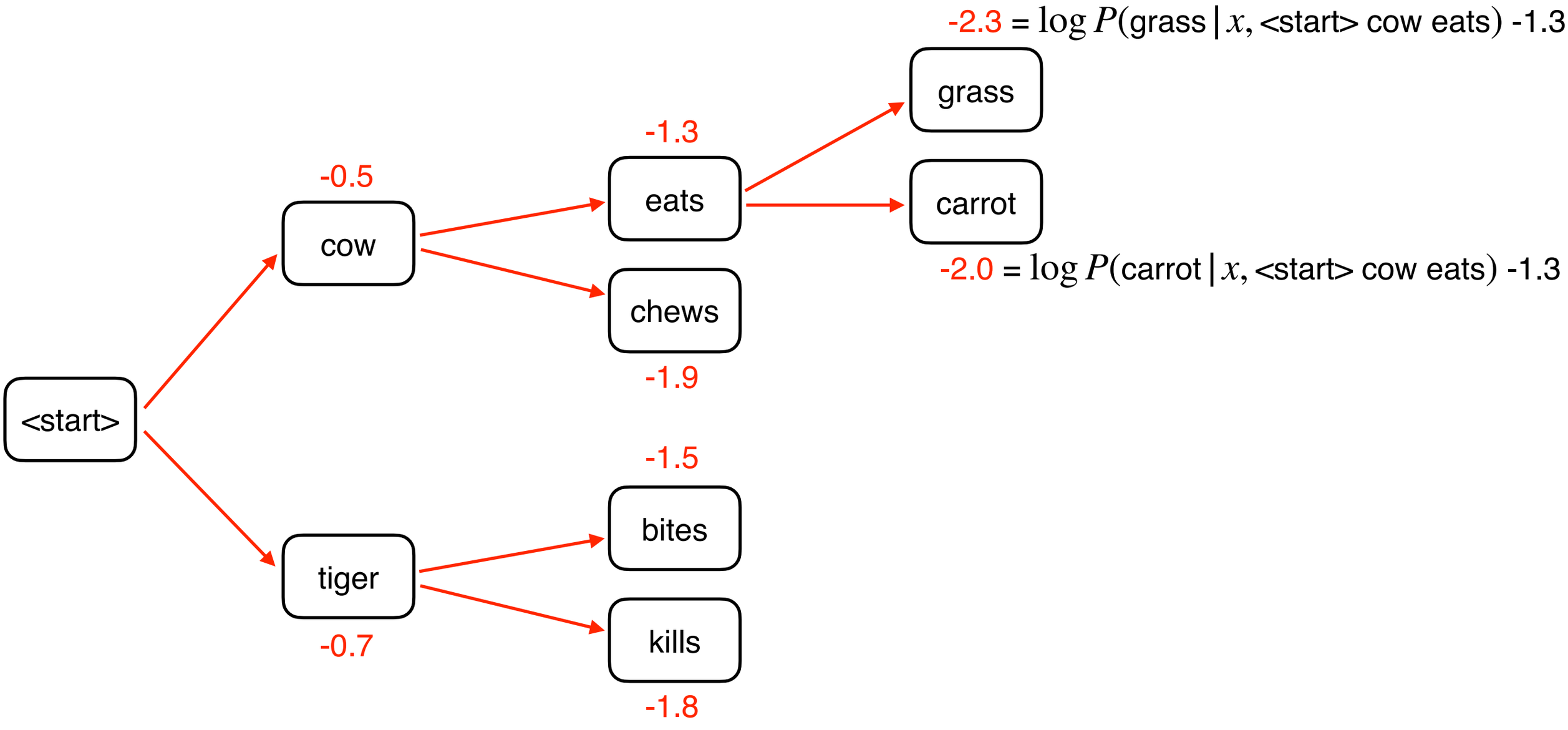
Beam search decoding with k=2



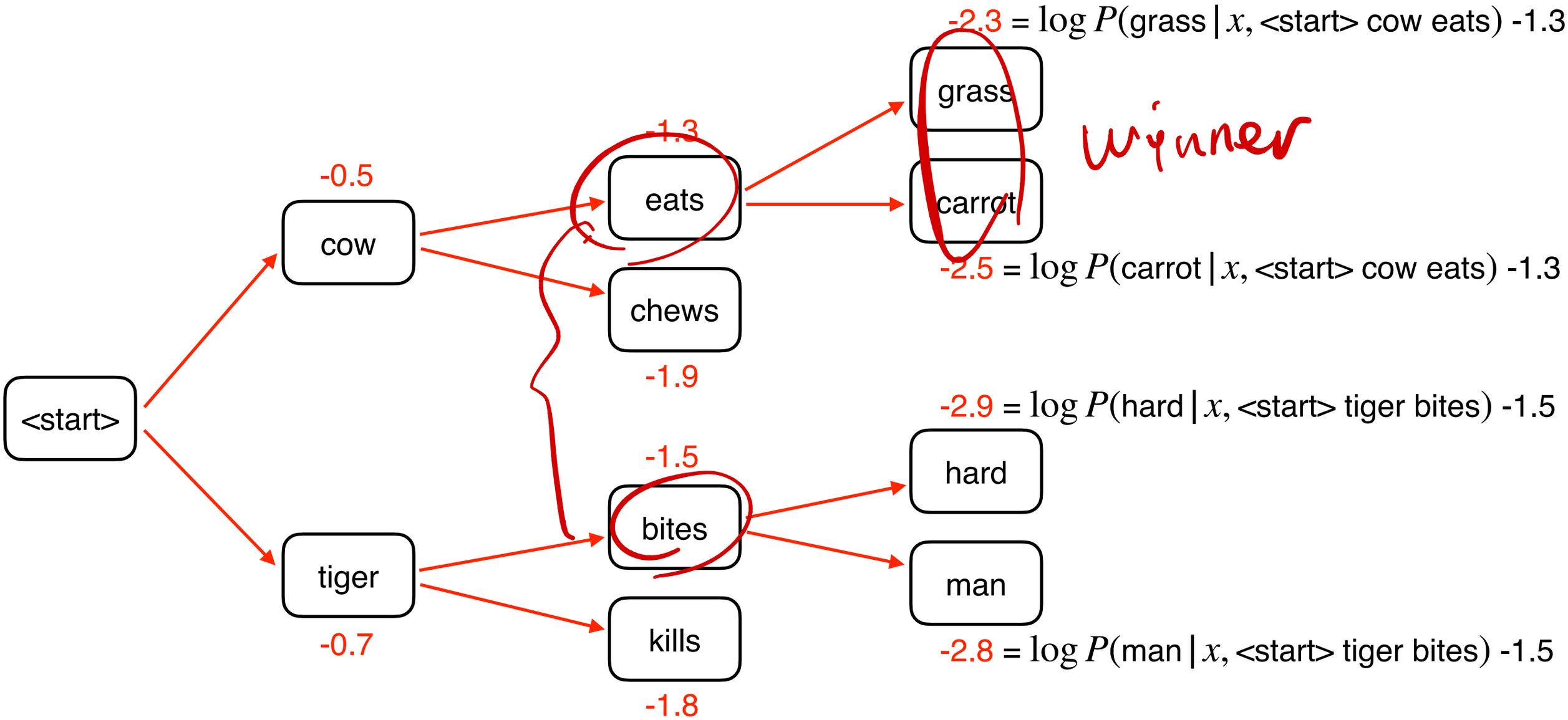
Beam search decoding with k=2



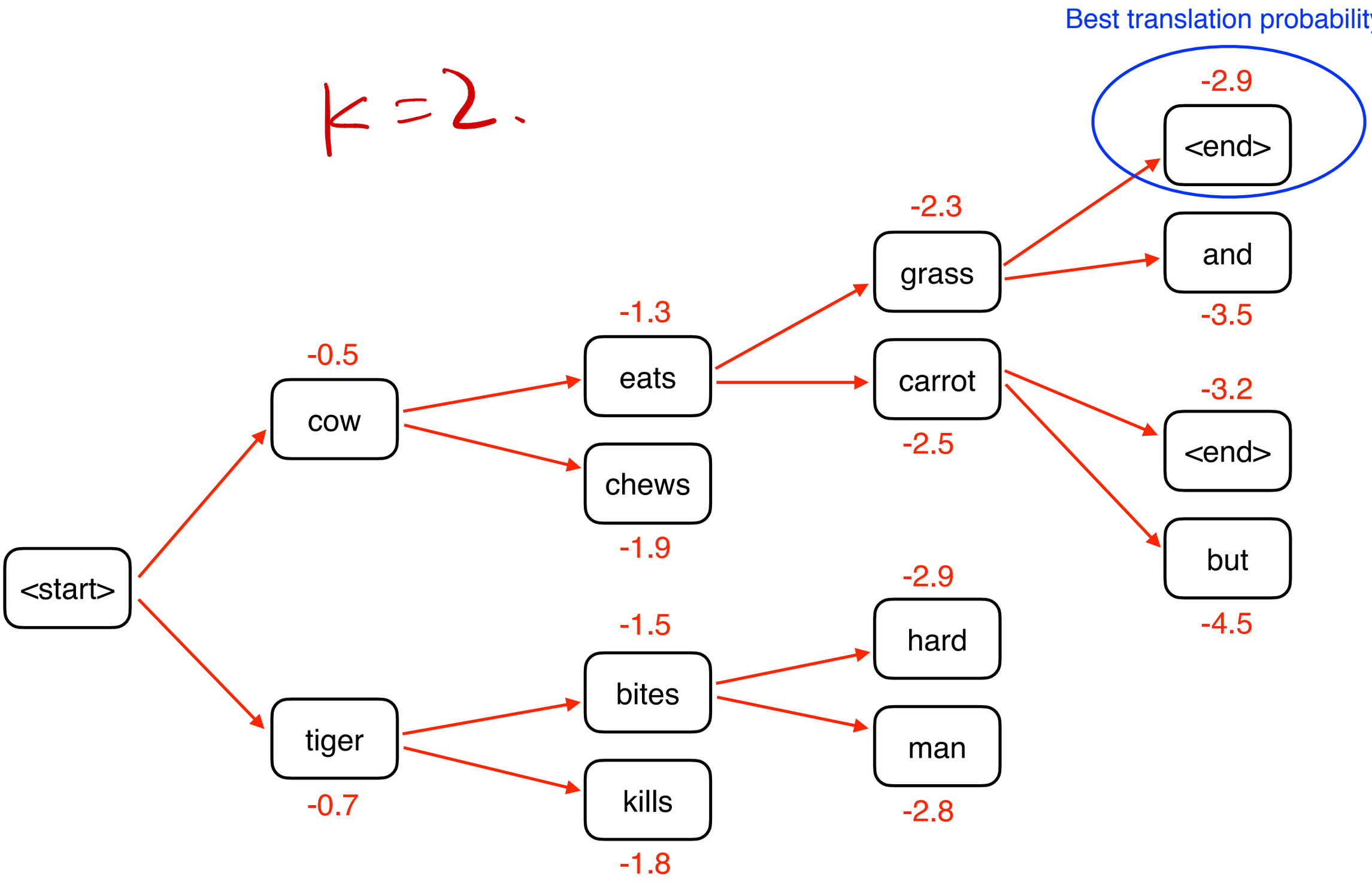
Beam search decoding with k=2



Beam search decoding with k=2

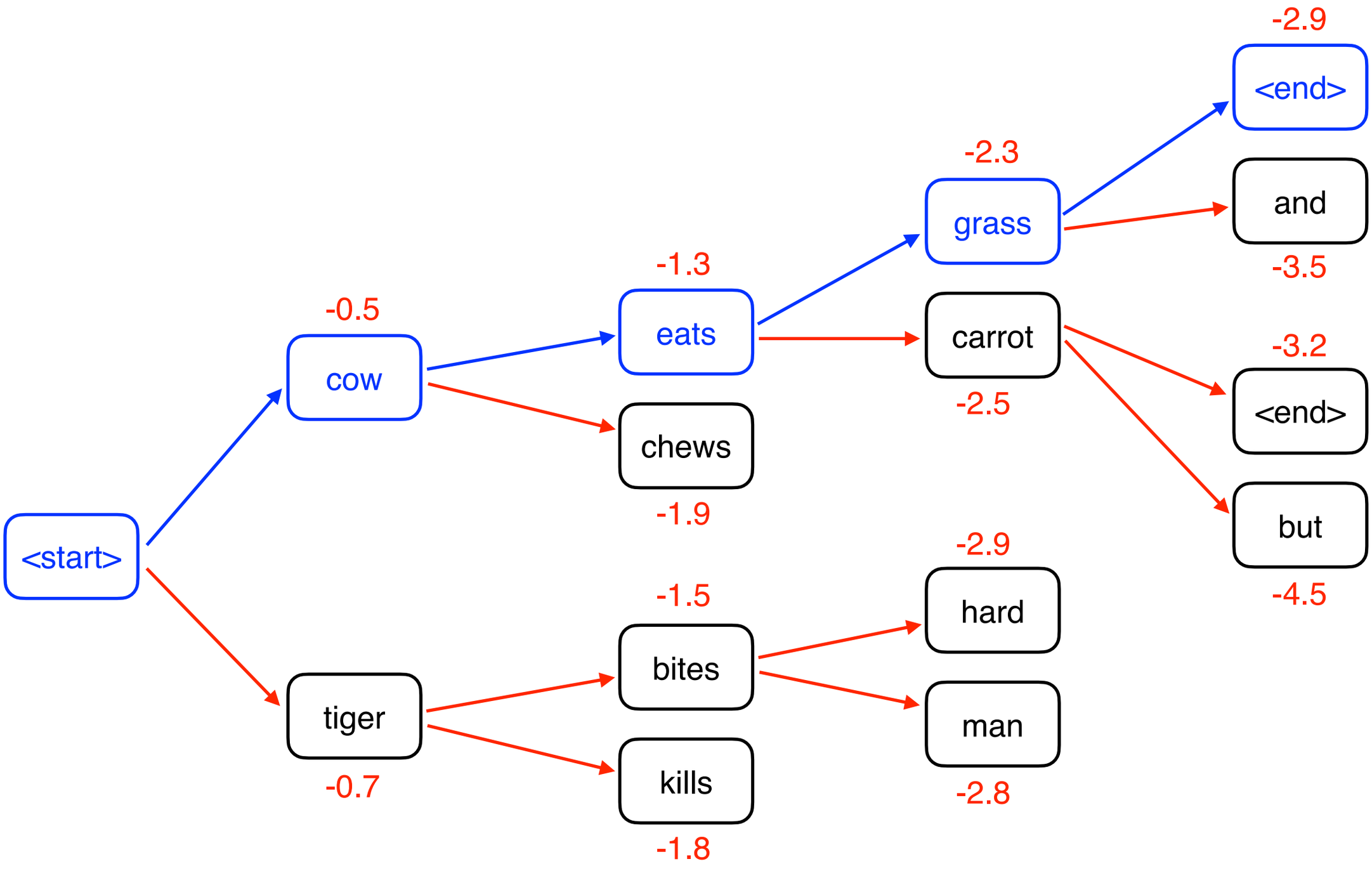


Beam search decoding with k=2



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)

stop until all sentences terminated

Neural MT: Summary

- Single end-to-end model
  - Statistical MT systems have multiple sub-components
- Less feature engineering
- Generated translation is more fluent
- But can produce new details that are not in the source sentence (*hallucination*)

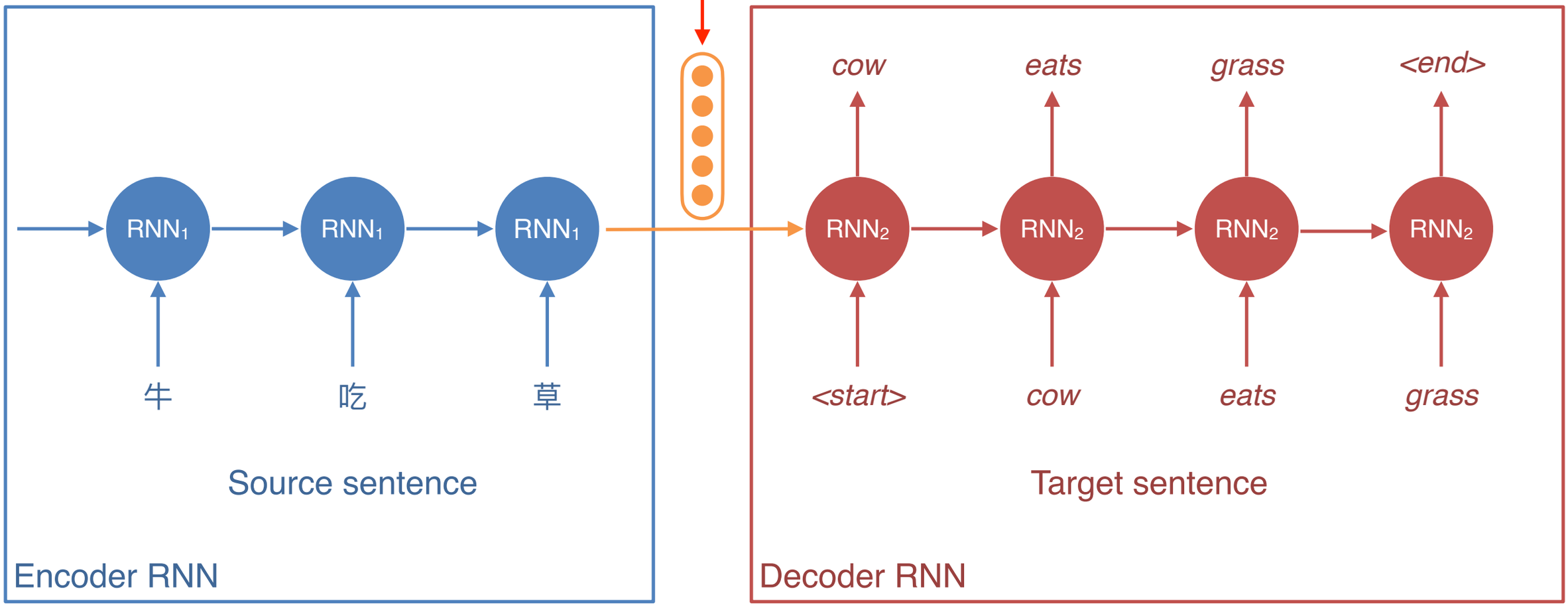
details not in source.

Source: Caldiero sprach mit E! Nachrichten nach dem hart erkämpften Sieg, noch immer unter dem Schock über den Gewinn des Großen Preises von 1 Million $.
Reference: Caldiero spoke with E! News after the hard-fought victory, still in shock about winning the $1 million grand prize.
NMT Translation: Caldiero spoke with E, after the hard won victory, still under the shock of the winning of the Grand Prix of 1 million $.

Lee, Firat, Agarwal, Fannjiang, Sussillo, 2018

Attention (!)

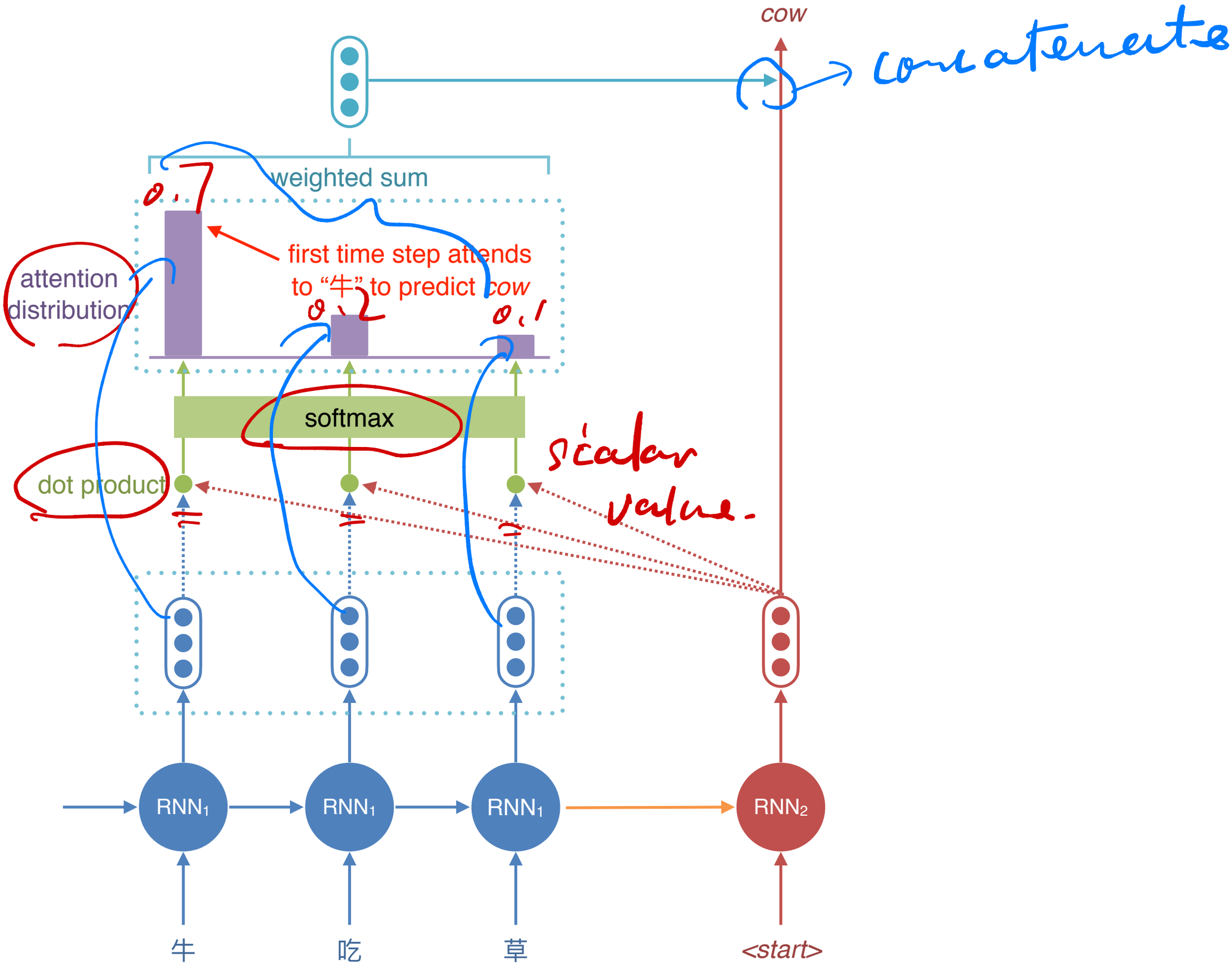
This vector encodes the whole source sentence

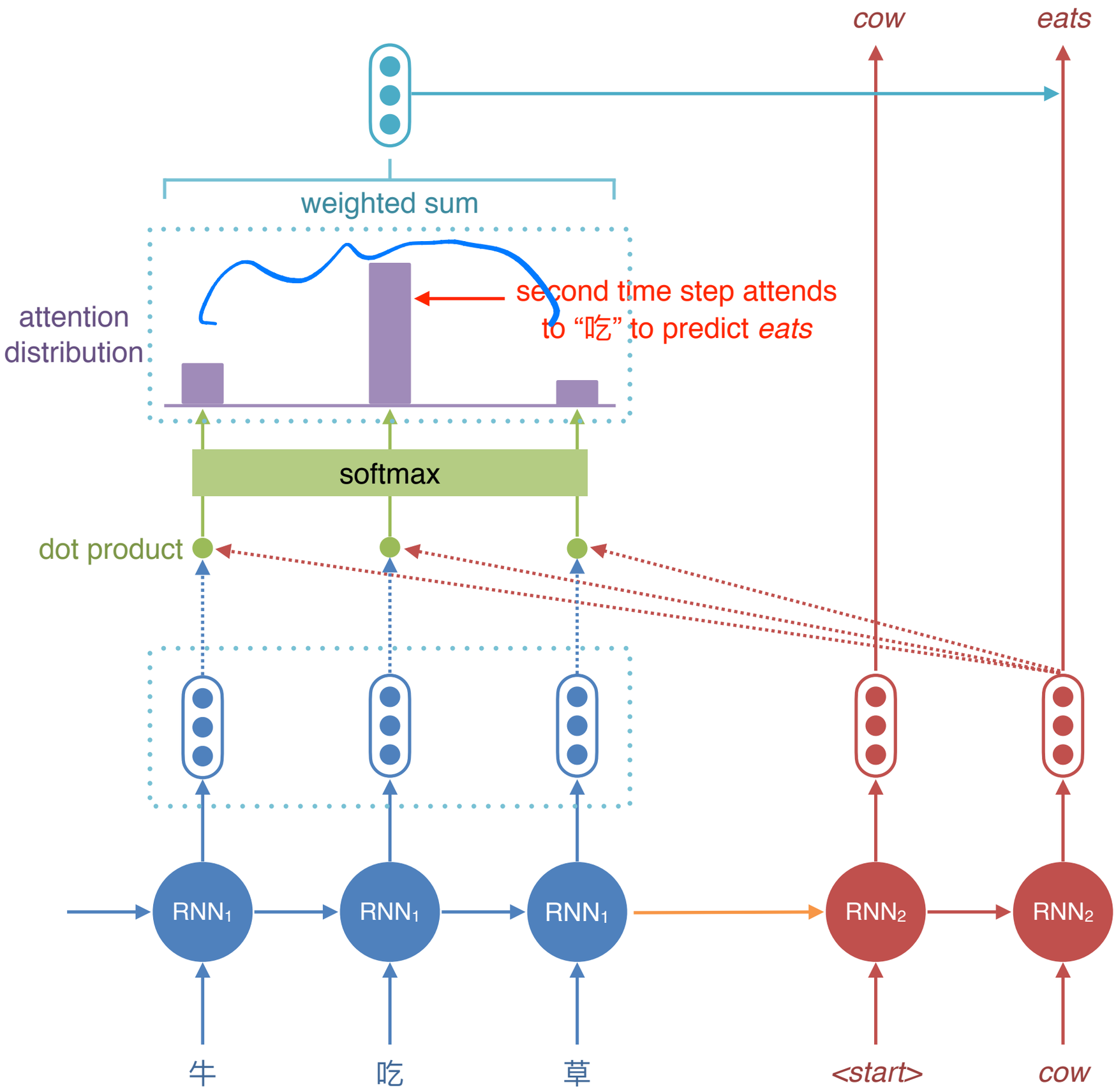


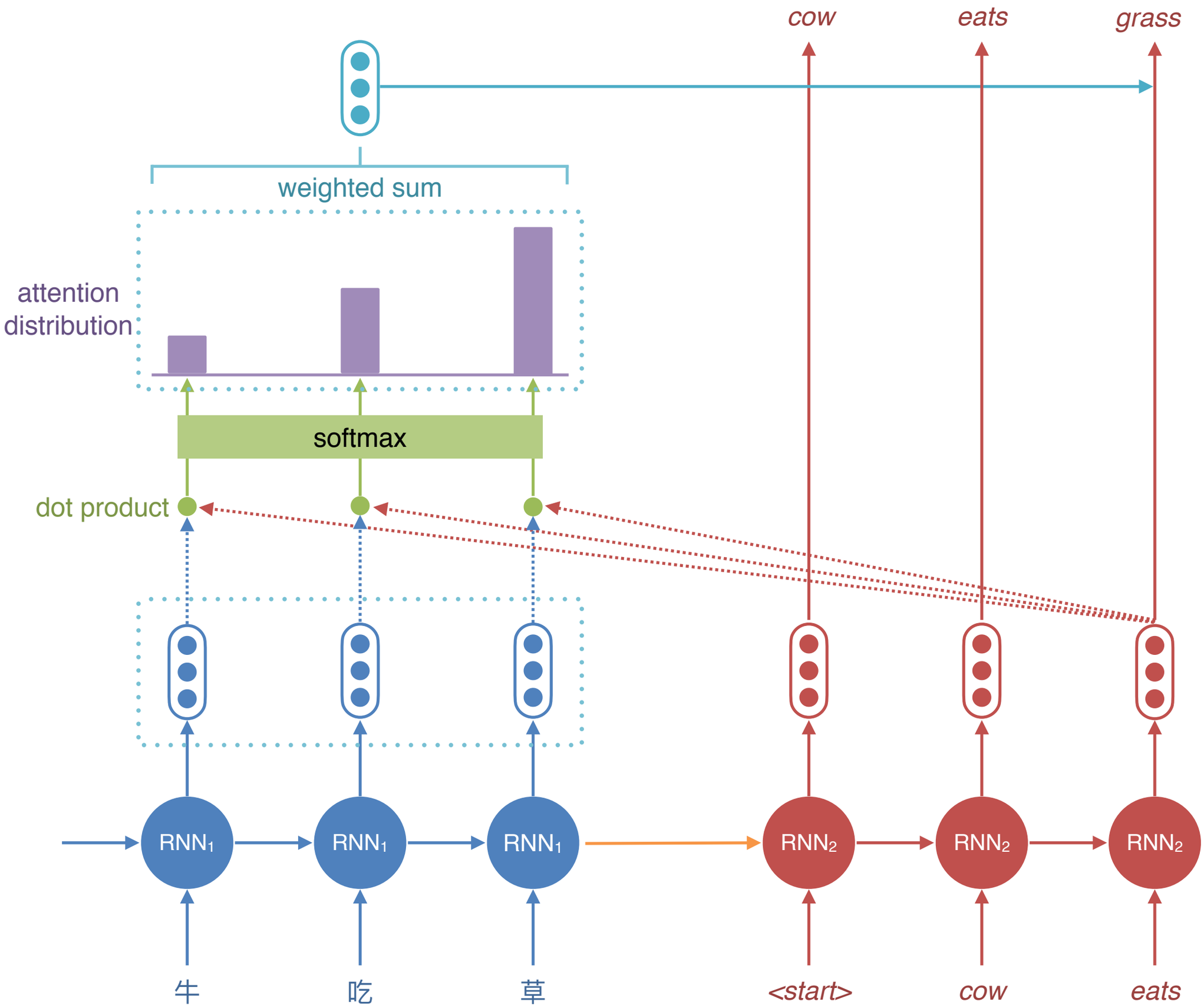
- With a long source sentence, the encoded vector is unlikely to capture all the information in the sentence
- This creates an **information bottleneck**

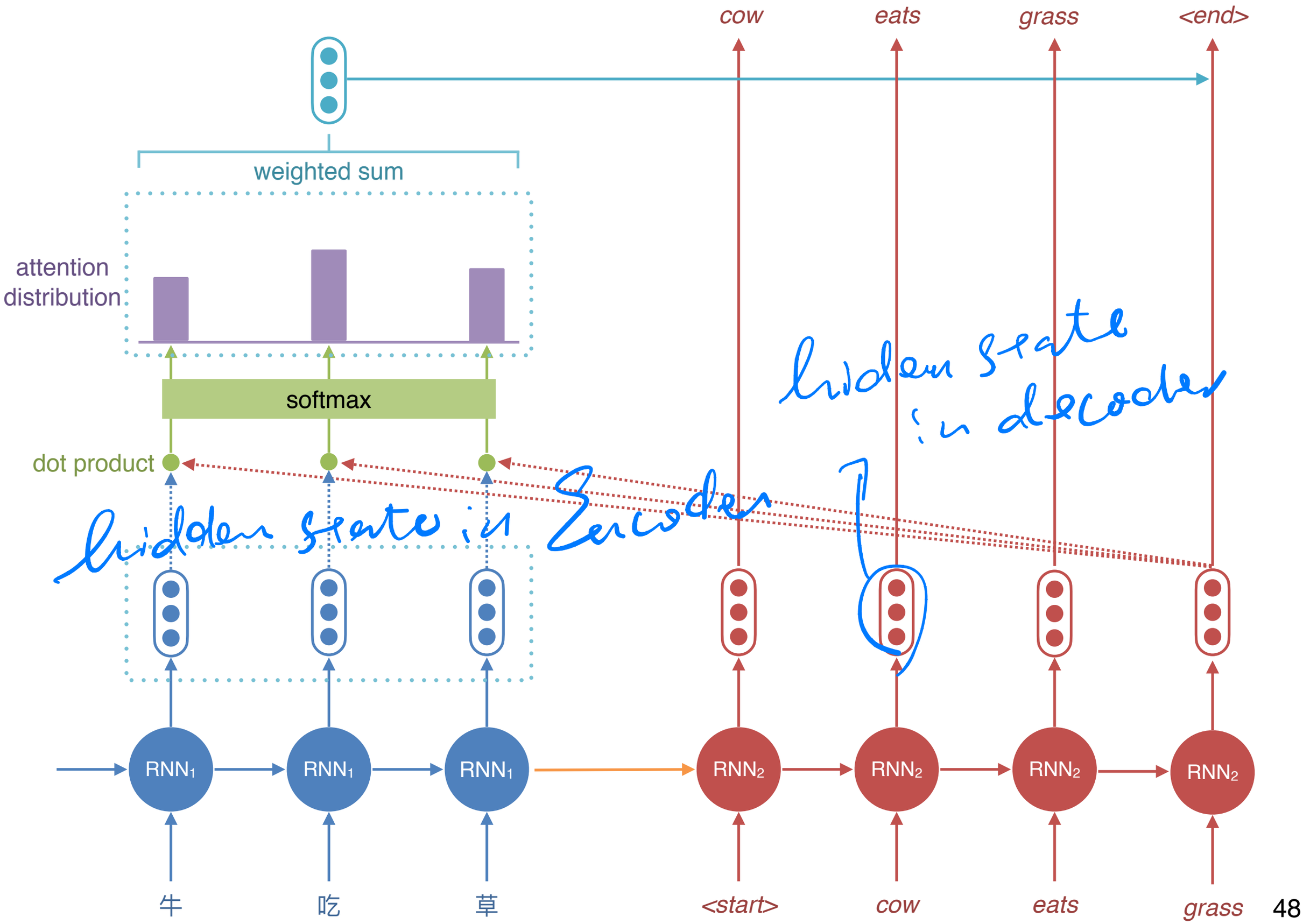
Attention

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


cow
concatenate
weighted sum
0.7
attention distribution
first time step attends to "牛" to predict cow
0.2
0.1
softmax
dot product
scalar value.
RNN1
RNN1
RNN1
RNN2
牛
吃
草
<start>


cow
eats
weighted sum
attention distribution
second time step attends to “吃” to predict eats
softmax
dot product
RNN1
RNN1
RNN1
RNN2
RNN2
牛
吃
草
<start>
cow

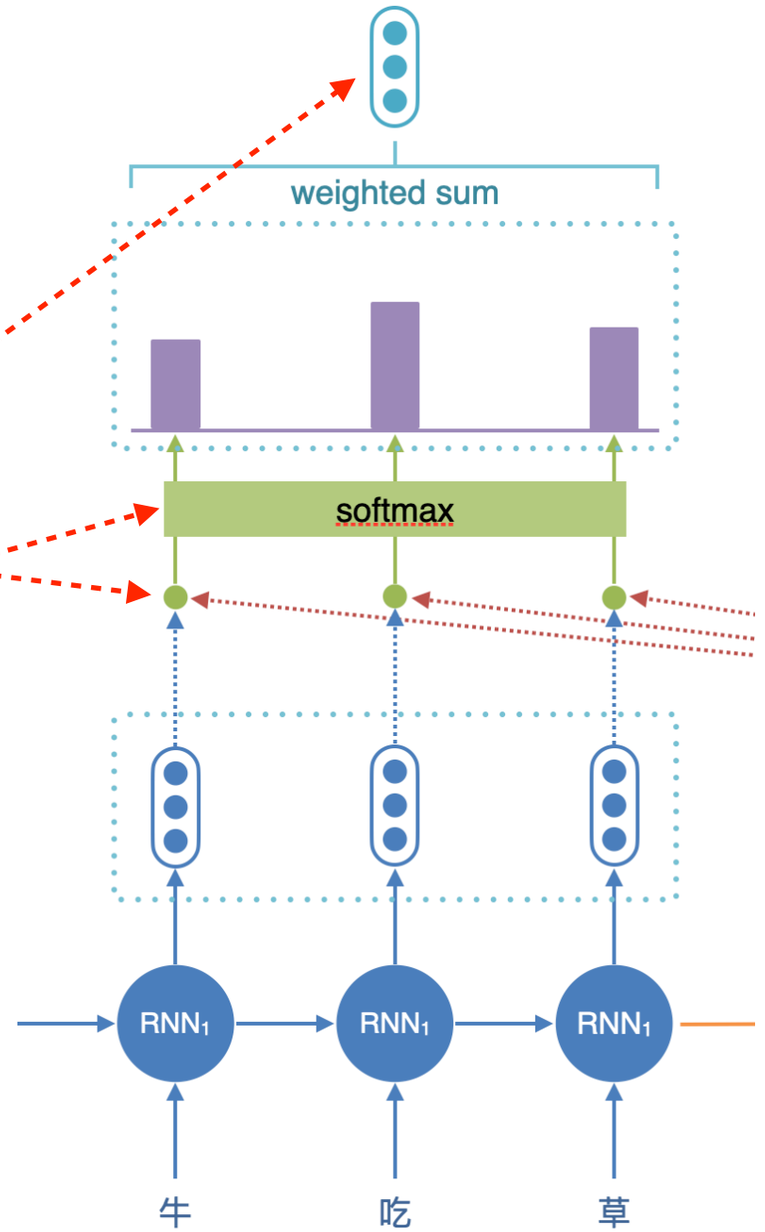

cow
eats
grass
weighted sum
attention distribution
softmax
dot product
RNN1
RNN1
RNN1
RNN2
RNN2
RNN2
牛
吃
草
<start>
cow
eats


cow
eats
grass
<end>
weighted sum
attention distribution
softmax
dot product
hidden state in decoder
hidden state in Encoder
RNN1
RNN1
RNN1
RNN2
RNN2
RNN2
RNN2
牛
吃
草
<start>
cow
eats
grass

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:
  - $e_t = [s_t^\intercal h_1, s_t^\intercal h_2, \ldots, s_t^\intercal h_{|x|}]$
- 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: $s_t^\intercal h_i$
  - Bilinear: $s_t^\intercal W h_i$
  - Additive: $v^\intercal \tanh(W_s s_t + W_h h_i)$
- 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

Avoid Bottleneck.

- Provides some form of interpretability
  - Attention weights can be seen as word alignments
- Most state-of-the-art MT systems are based on this architecture
  - 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

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

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

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

"Brevity Penalty" to penalise short outputs

A Final Word

- Statistical MT
- Neural MT
- Encoder-decoder with attention architecture is a general architecture that can be used for other tasks
  - Summarisation (lecture 21)
  - Dialogue generation

Reading

- https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf (Section 1-5)