Machine Translation

Session 8: Sequence-to-sequence models

Sharid Loáiciga — June 10th, 2020

Slides credits: Alessandro Raganato
(who credits Abigail See & Rico Sennrich)

Modelling Translation

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network*

Suppose we are translating English to Finnish

- We want to find best Finnish sentence $T\,(x_1, \ldots, x_m)$, given a English sentence $S\ (y_1, \ldots, y_n)$

- We can express translation as a probabilistic model:

$$T^* = \arg\max_T p(T|S)$$

- Expanding using the chain rule gives:

$$\begin{aligned} p(T|S) &= p(y_1, \ldots, y_n | x_1, \ldots, x_m) \\ &= \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1}, x_1, \ldots, x_m) \end{aligned}$$

Differences Between Translation and Language Model

- Target-side language model:

$$p(T) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1})$$

- Translation model:

$$p(T|S) = \prod_{i=1}^{n} p(y_i|y_1, \ldots, y_{i-1}, x_1, \ldots, x_m)$$

- We could just treat sentence pair as one long sequence, but:
  - We do not care about $p(S)$
  - We may want different vocabulary, network architecture for source text

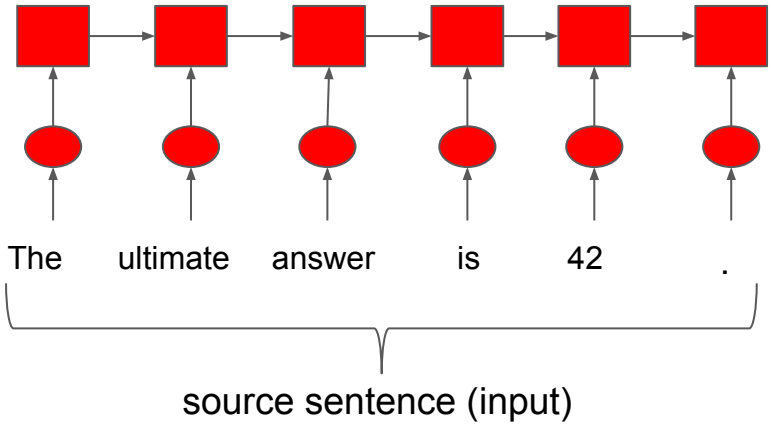
⇨ Use separate RNNs for source and target.

Neural Machine Translation (NMT)

- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.

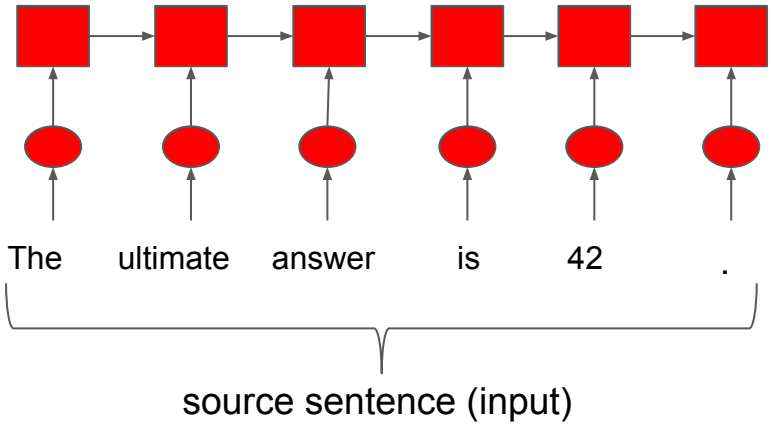
Neural Machine Translation (NMT)

- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



Neural Machine Translation (NMT)

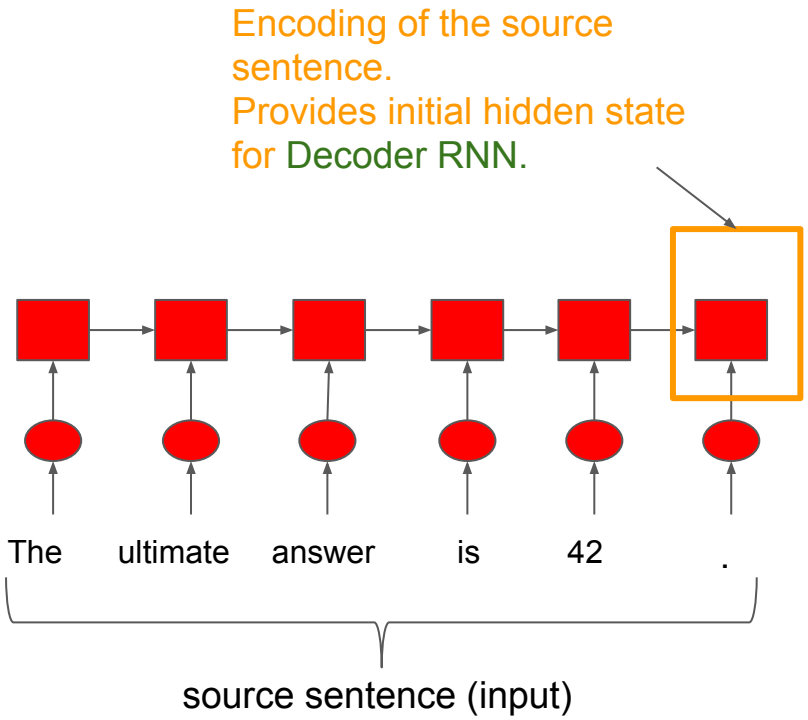
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

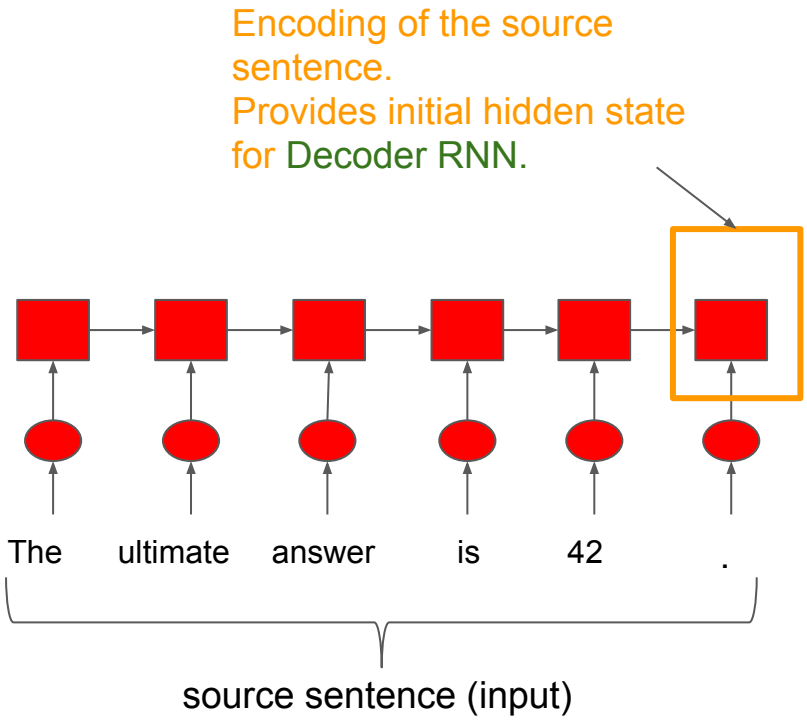
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.

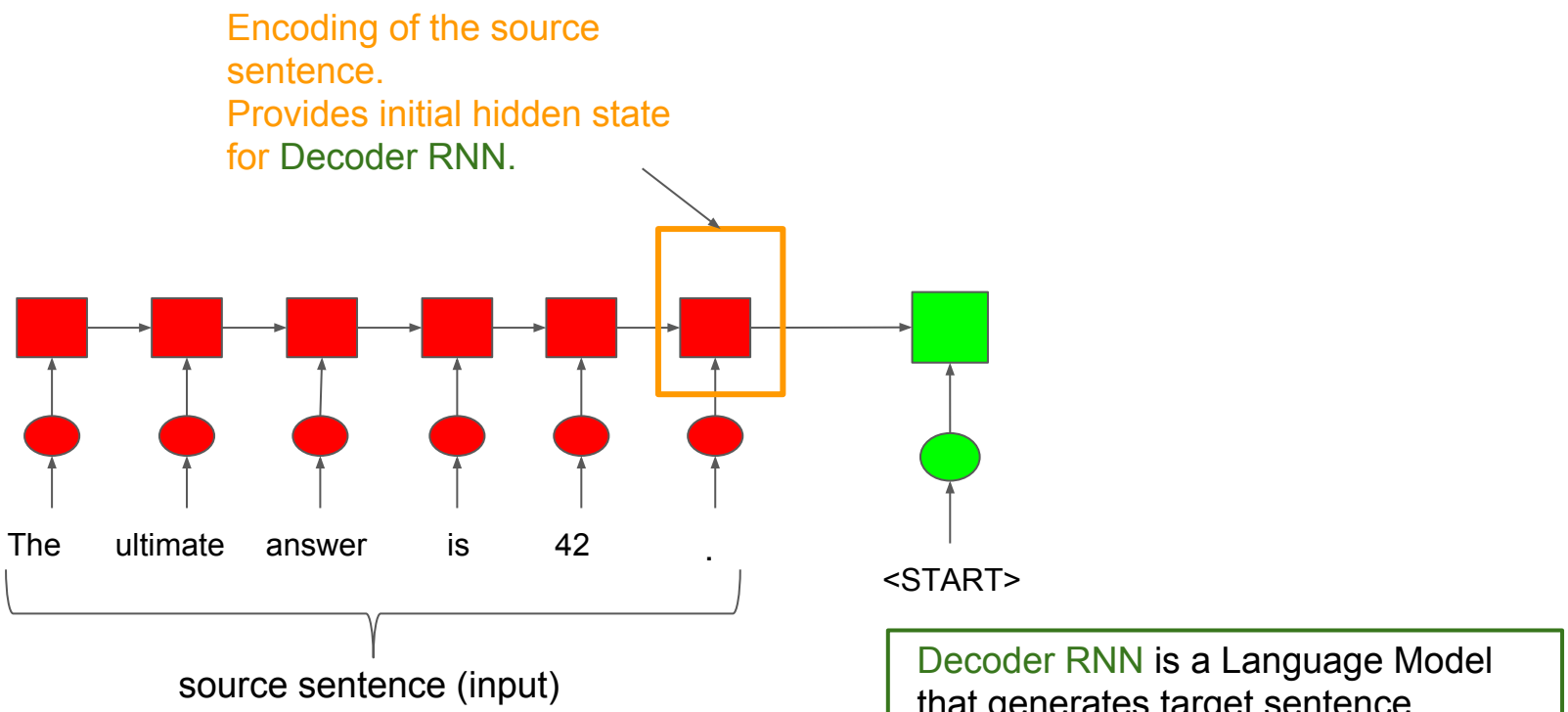


Decoder RNN is a Language Model that generates target sentence, *conditioned* on encoding.

The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

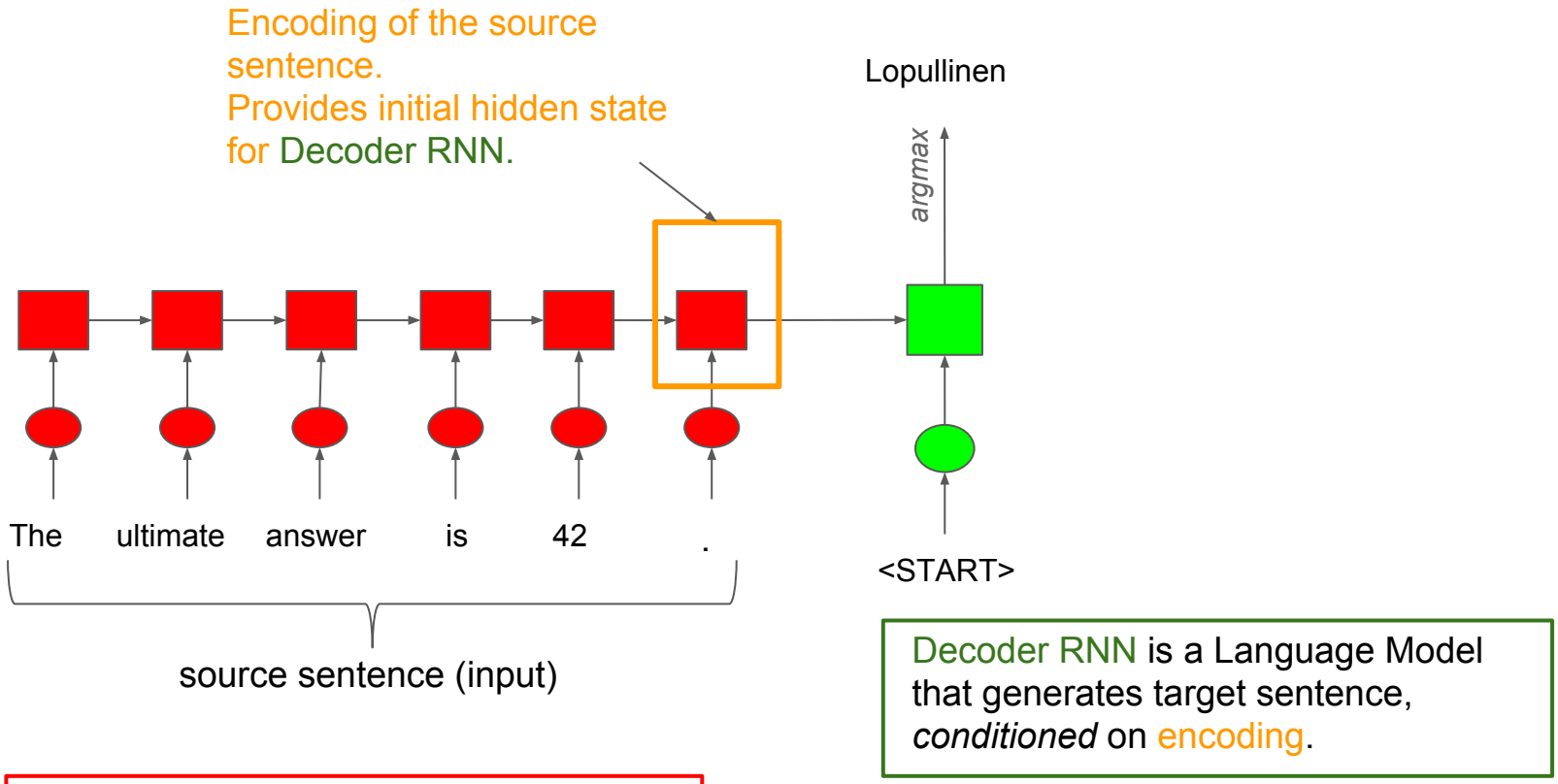
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

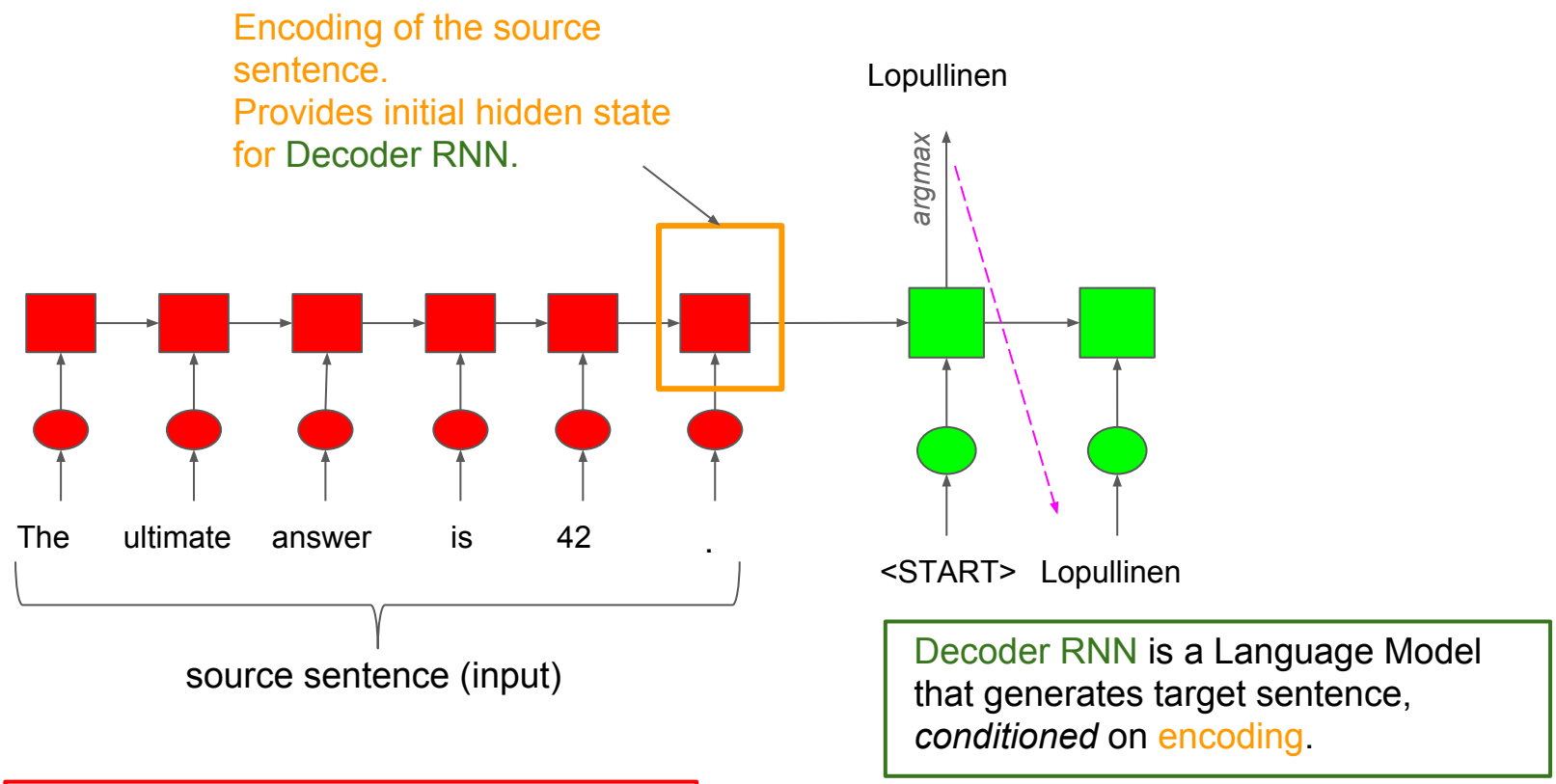
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

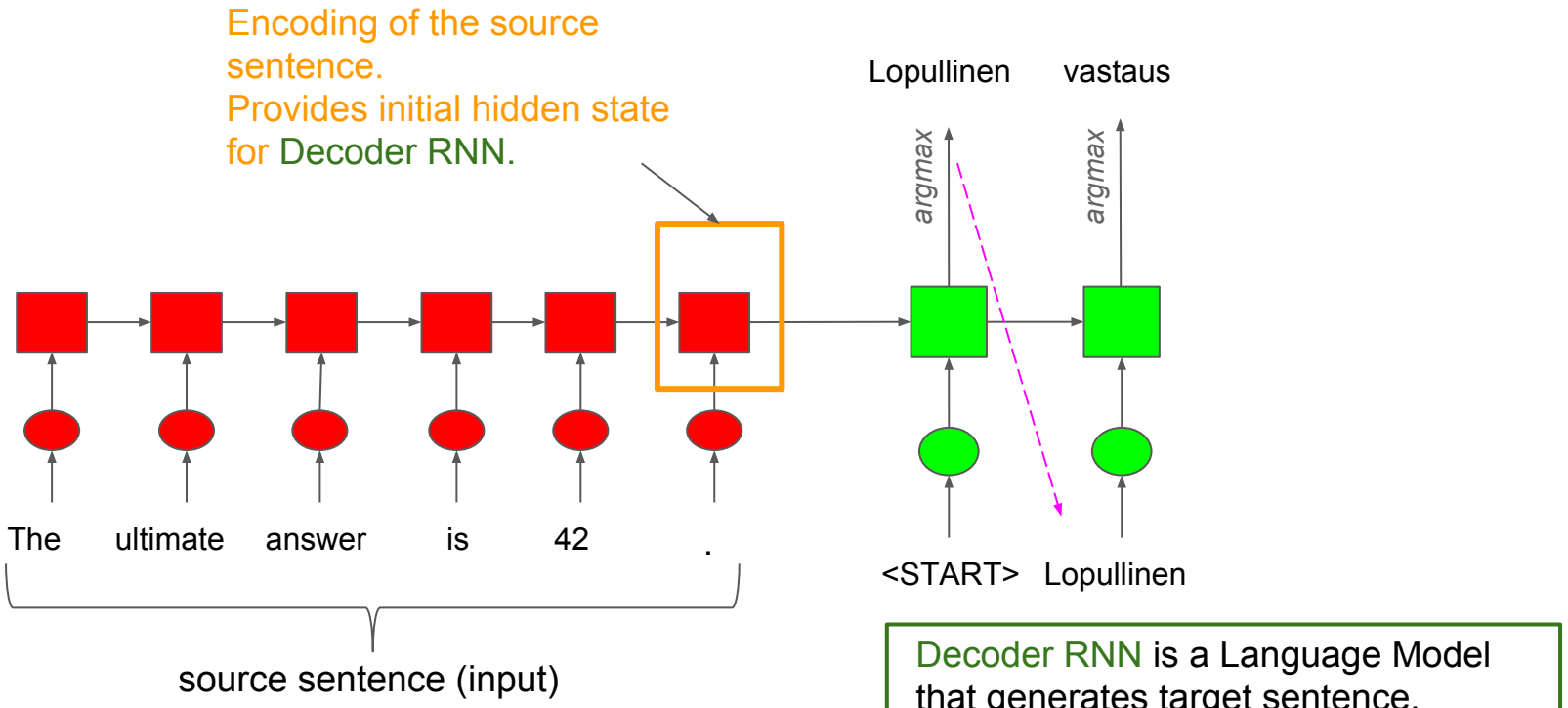
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

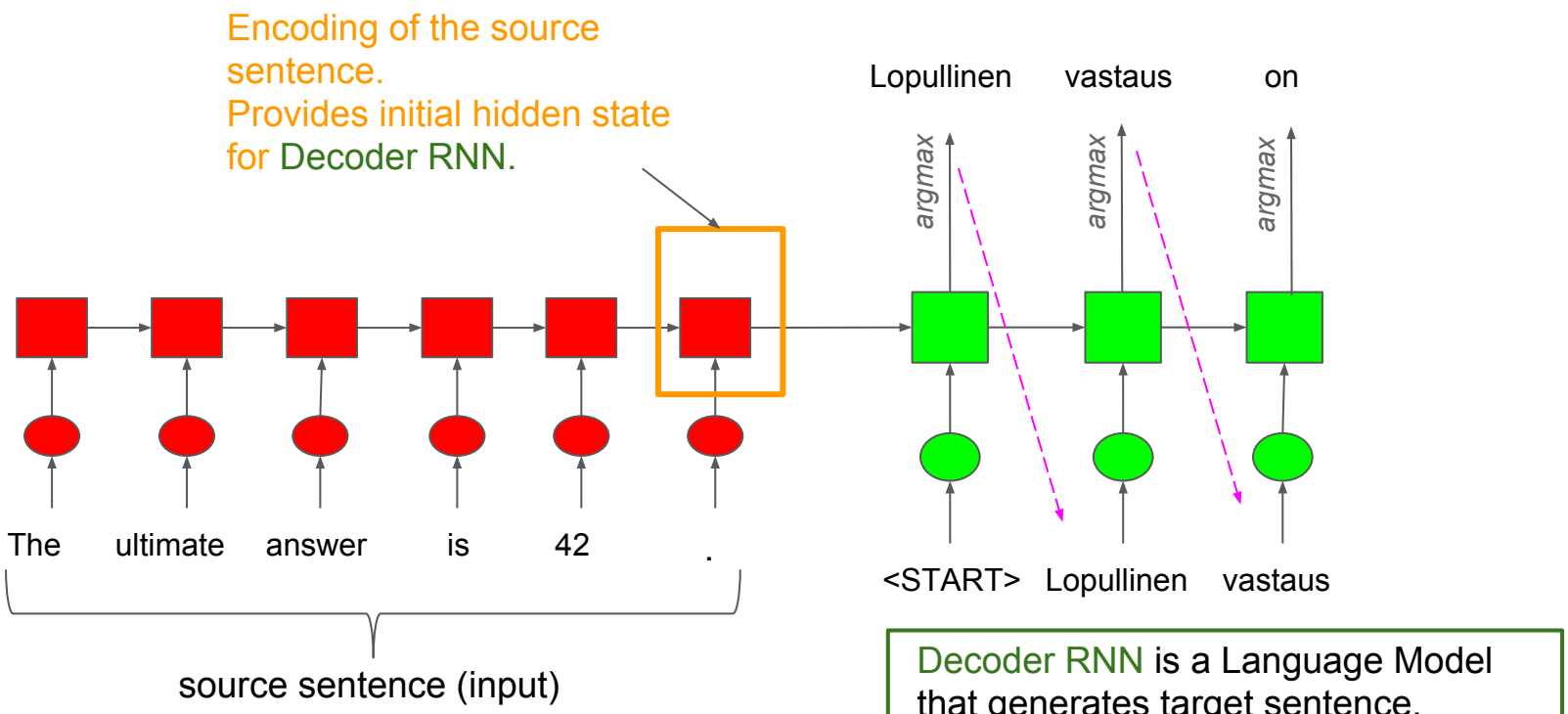
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

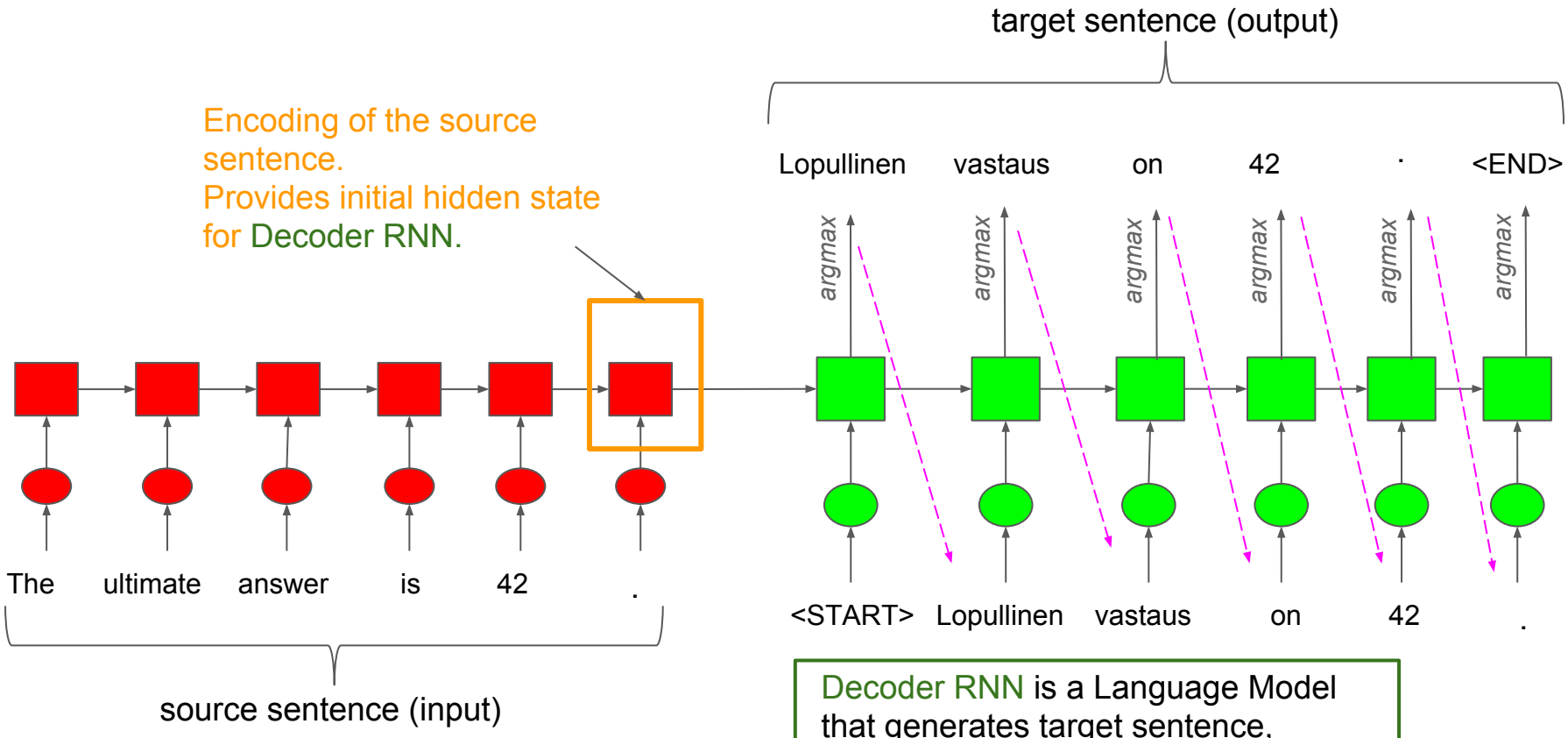
- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



The **encoder** creates a representation of the source sentence.

Neural Machine Translation (NMT)

- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.



Decoder RNN is a Language Model that generates target sentence, *conditioned* on encoding.

The **encoder** creates a representation of the source sentence.

Note: This diagram shows test time behavior: decoder output is fed in --> as next step's input

Neural Machine Translation (NMT)

- The neural network architecture is called sequence-to-sequence (aka seq2seq) or encoder-decoder and usually it involves *two* RNNs.

- Last encoder hidden-state "summarises" source sentence

- Sequence-to-sequence is versatile! It is useful for more than just MT

➢ Many NLP tasks can be phrased as sequence-to-sequence:

  - Summarization (long text → short text)
  - Dialogue (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)

Neural Machine Translation (NMT)

- The sequence-to-sequence model is an example of a **Conditional Language Model**:
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

- **Question**: How to train a NMT system?
- **Answer**: Get a big parallel corpus...

Training a Neural Machine Translation system

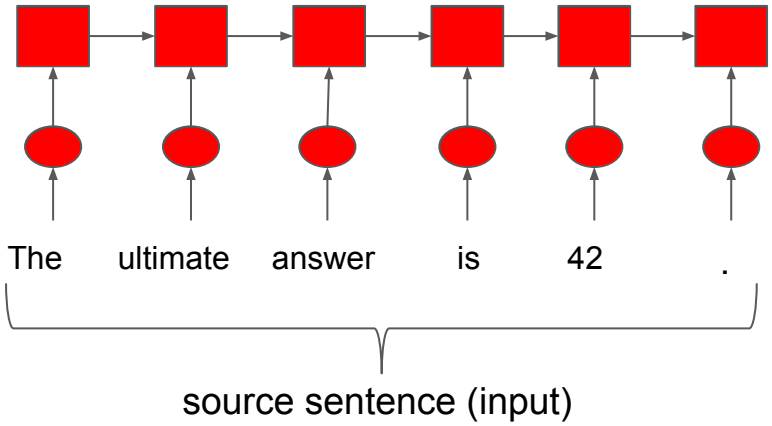
The ultimate answer is 42 .

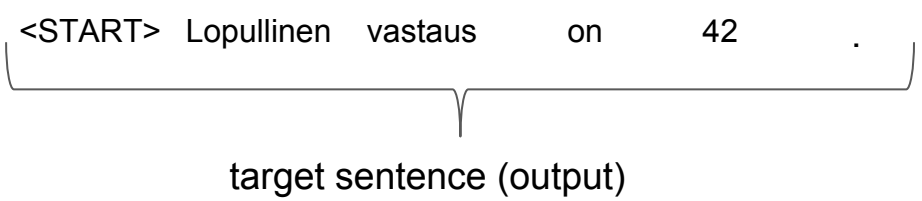
source sentence (input)

<START> Lopullinen vastaus on 42 .

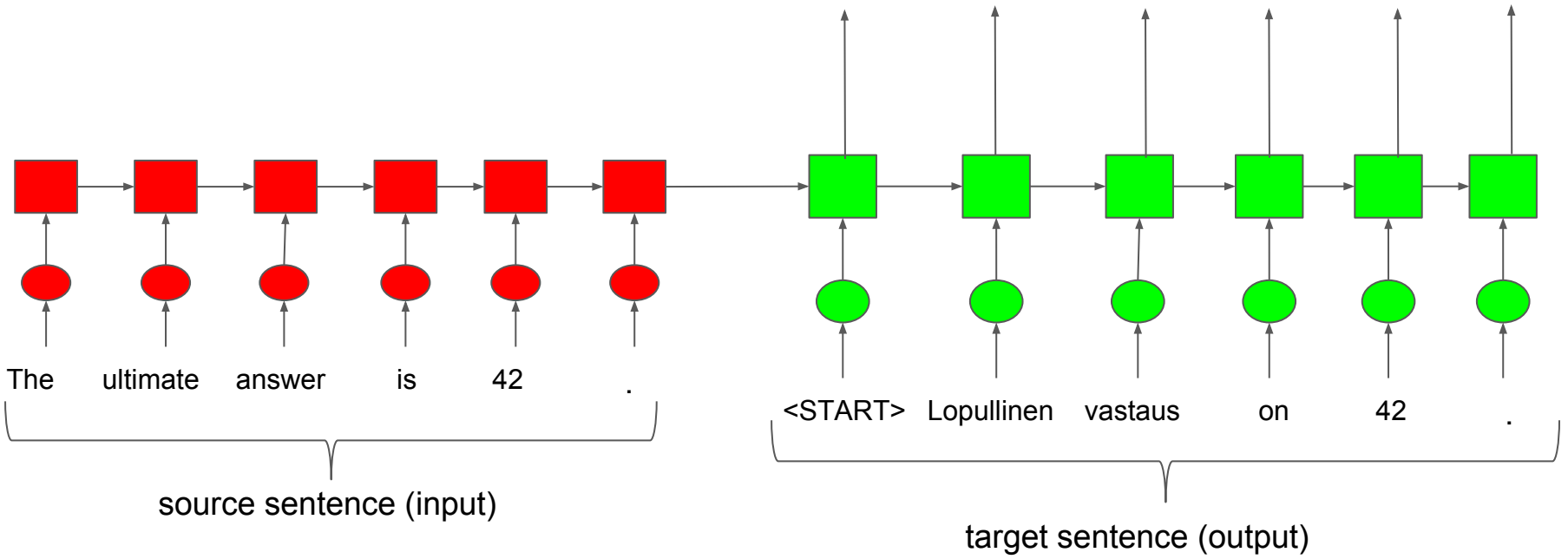
target sentence (output)

Training a Neural Machine Translation system

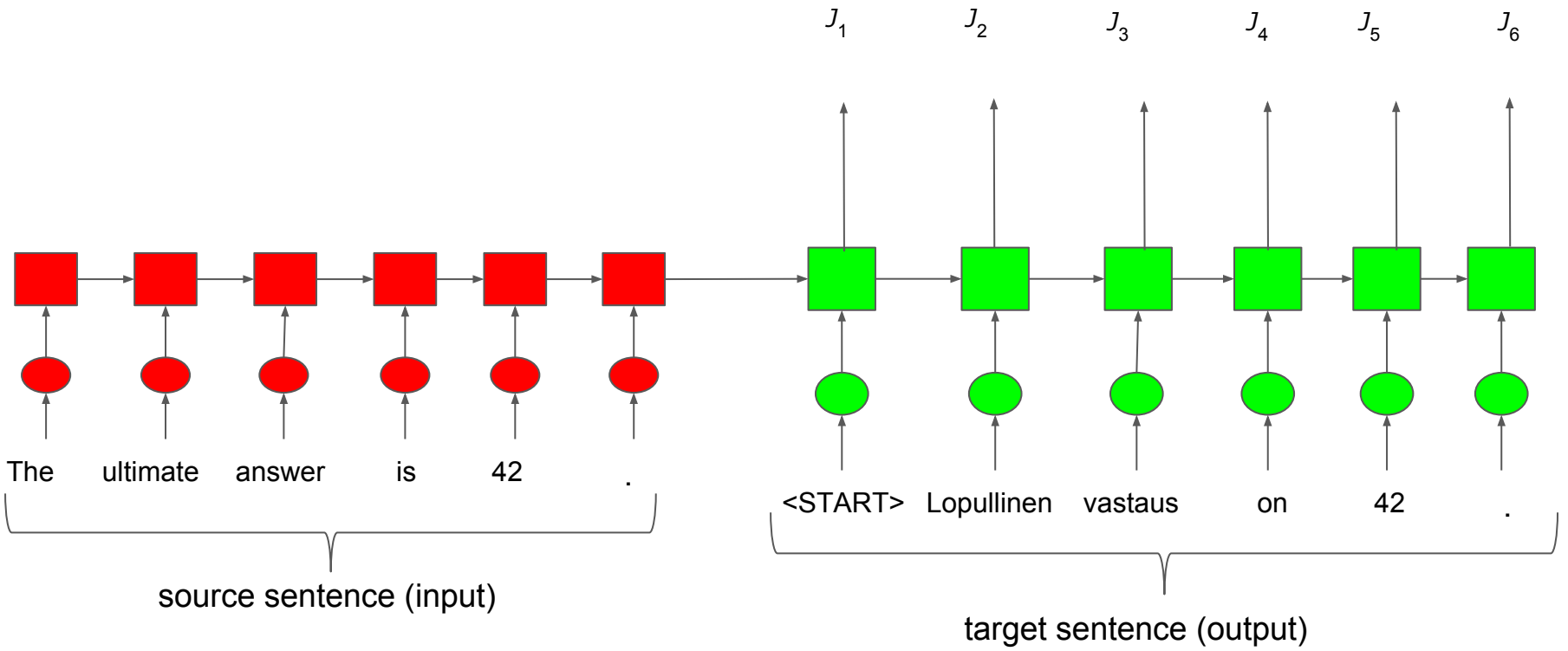




Training a Neural Machine Translation system



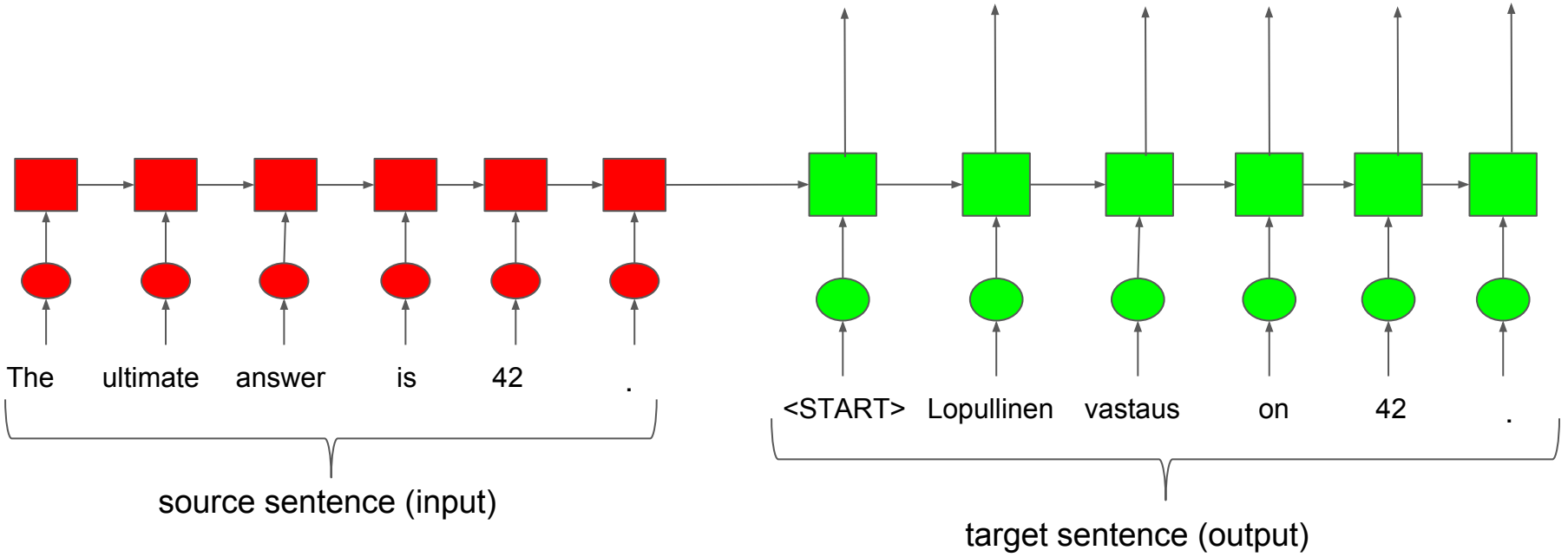
Training a Neural Machine Translation system



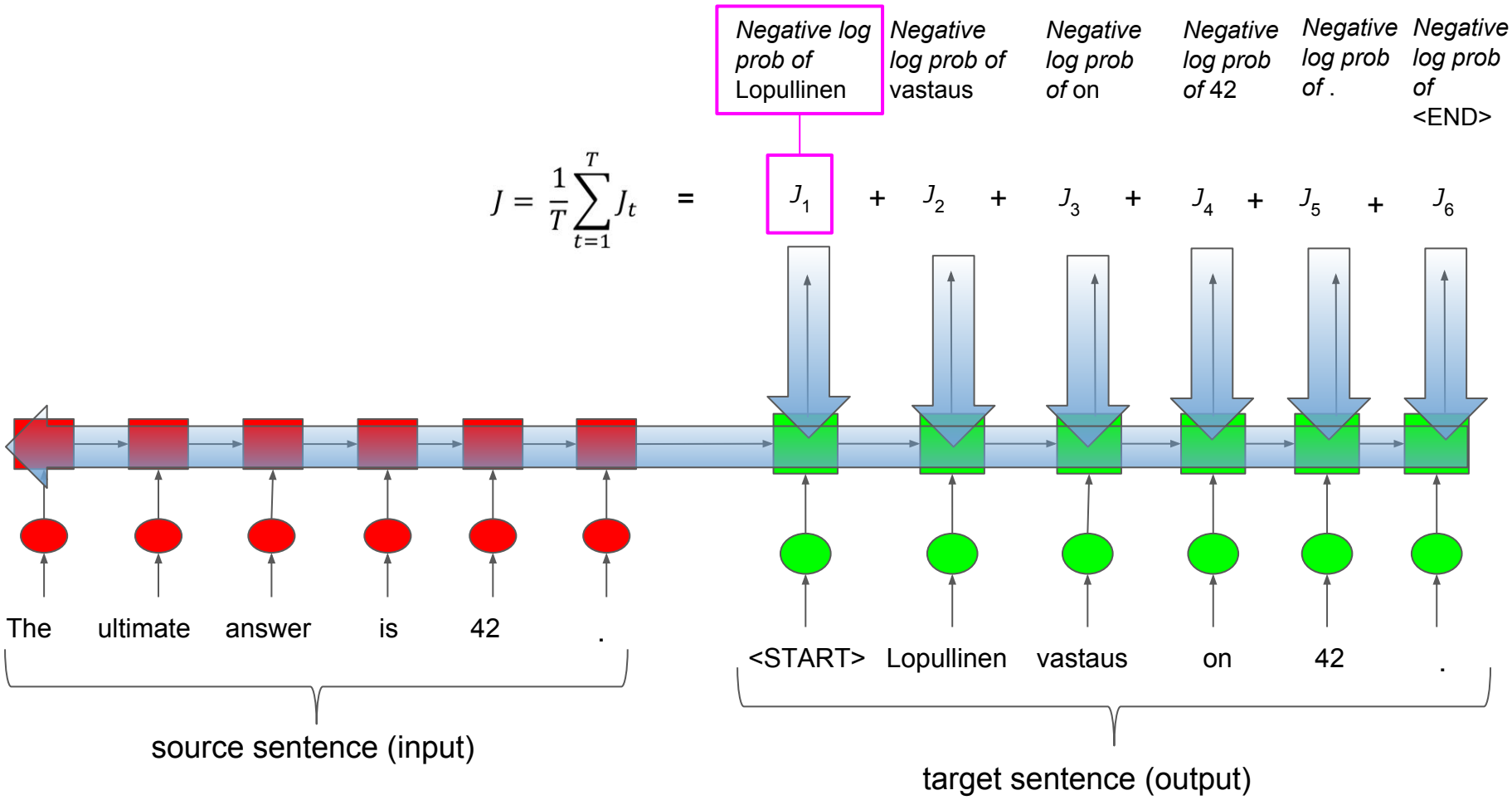
Training a Neural Machine Translation system

Negative log prob of Lopullinen | *Negative log prob of* vastaus | *Negative log prob of* on | *Negative log prob of* 42 | *Negative log prob of* . | *Negative log prob of* <END>

$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = J_1 + J_2 + J_3 + J_4 + J_5 + J_6$$



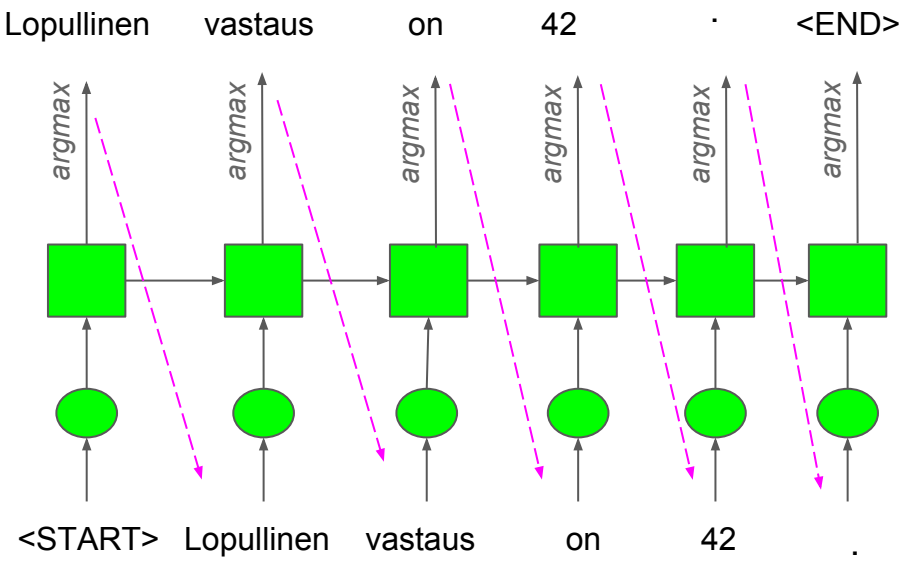
Training a Neural Machine Translation system



Seq2seq is optimized as a **single system**, Backpropagation operates "**end-to-end**".

Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)
- **Problems with this method?**

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
  - ➢ Input: The ultimate answer is 42 . (Lopullinen vastaus on 42 .)
    - ⇨ Lopullinen _
    - ⇨ Lopullinen lähtölaskenta _ (no going back now...)
- How to fix this?

Exhaustive search decoding

- Ideally we want to find a (length *T*) translation *y* that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

- We could try computing all possible sequences *y*
  - This means that on each step *t* of the decoder, we're tracking V^t possible partial translations, where *V* is vocab size
  - This $O(V^T)$ complexity is far too expensive!

Beam search decoding

- Core idea: On each step of decoder, keep track of the *k* most probable partial translations (which we call *hypotheses*)
  - *k* is the beam size (in practice around 5 to 10)

- A hypothesis $y_1,...,y_t$ has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top *k* on each step

- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam search decoding: example

- Beam size = k = 2. Blue numbers = $$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

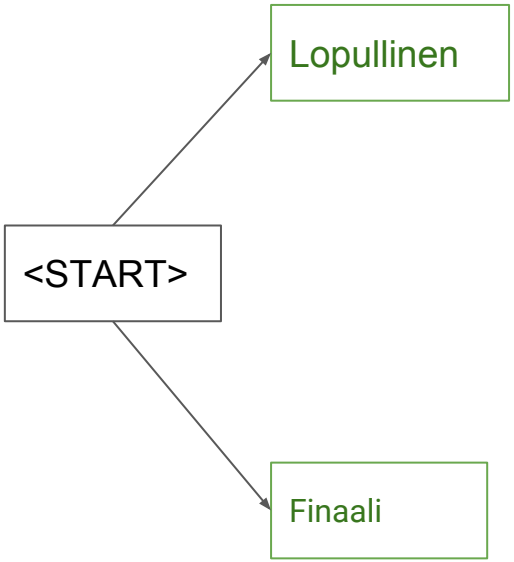
<START>

Calculate prob dist of next word

Beam search decoding: example

- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

-0.7 = log P_{LM} (Lopullinen|<START>)

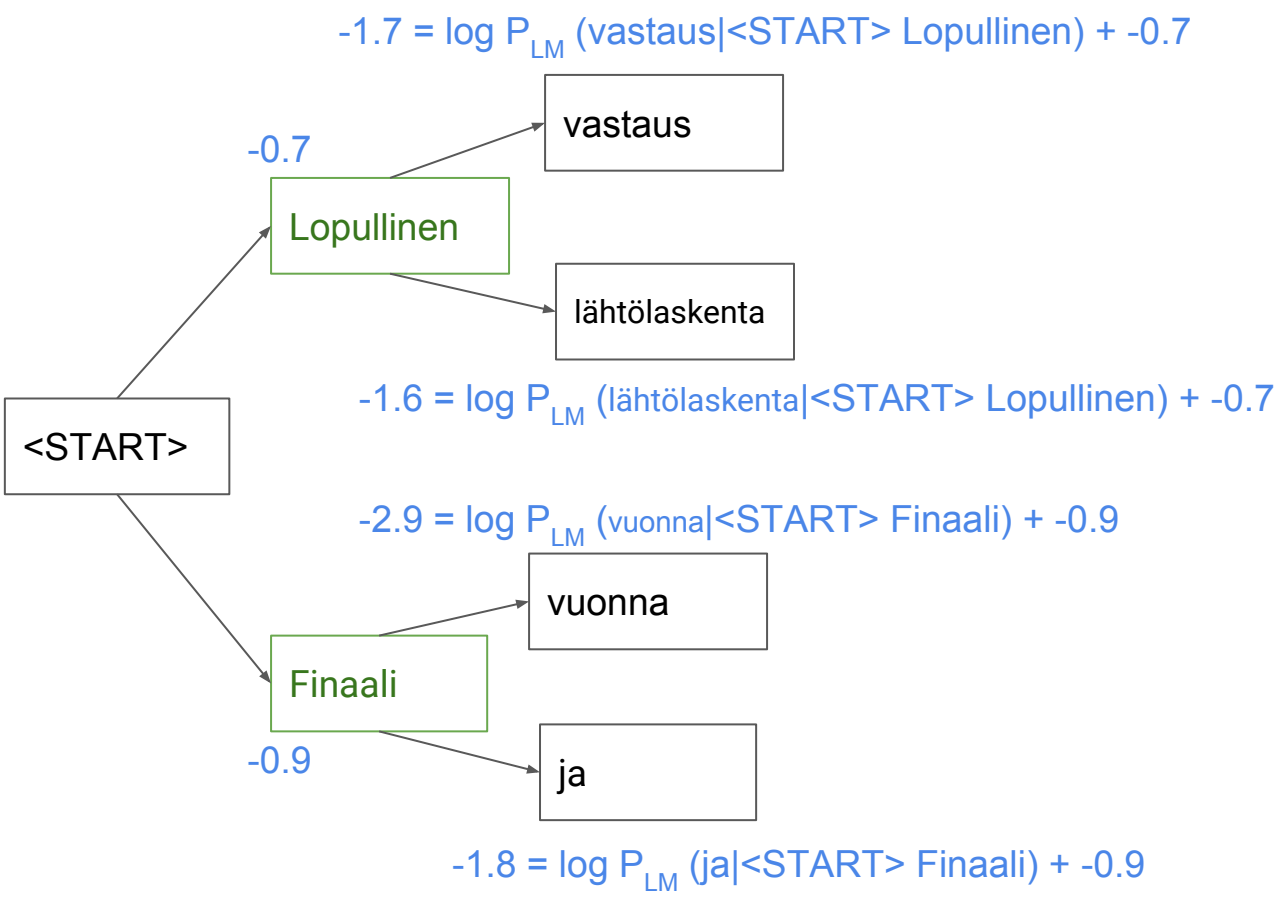


-0.9 = log P_{LM} (Finaali|<START>)

Take top *k* words and compute scores

Beam search decoding: example

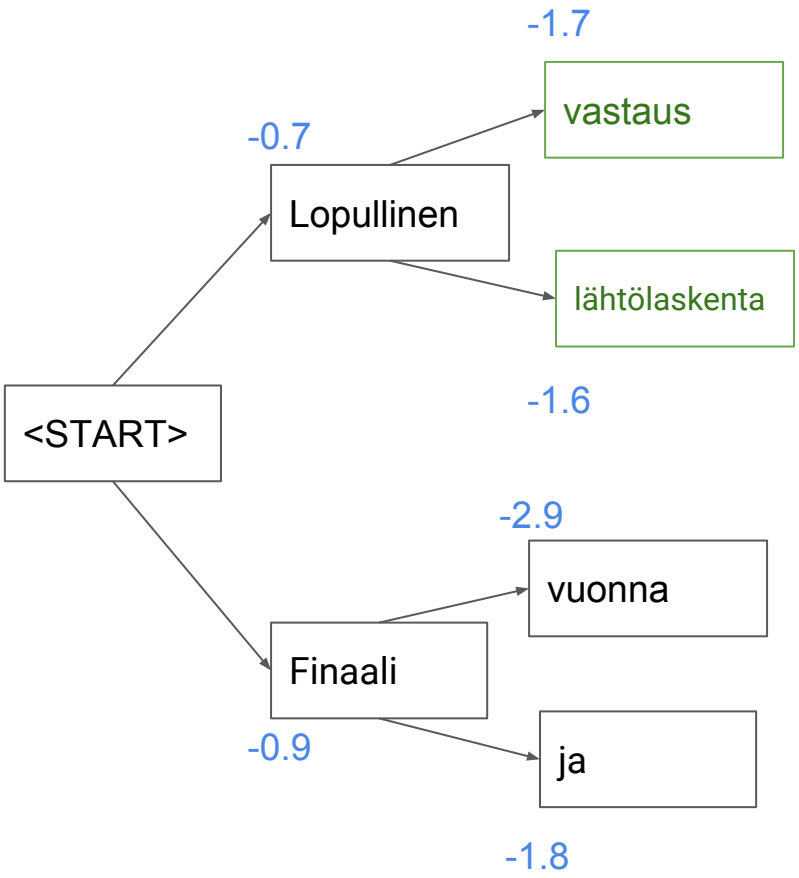
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

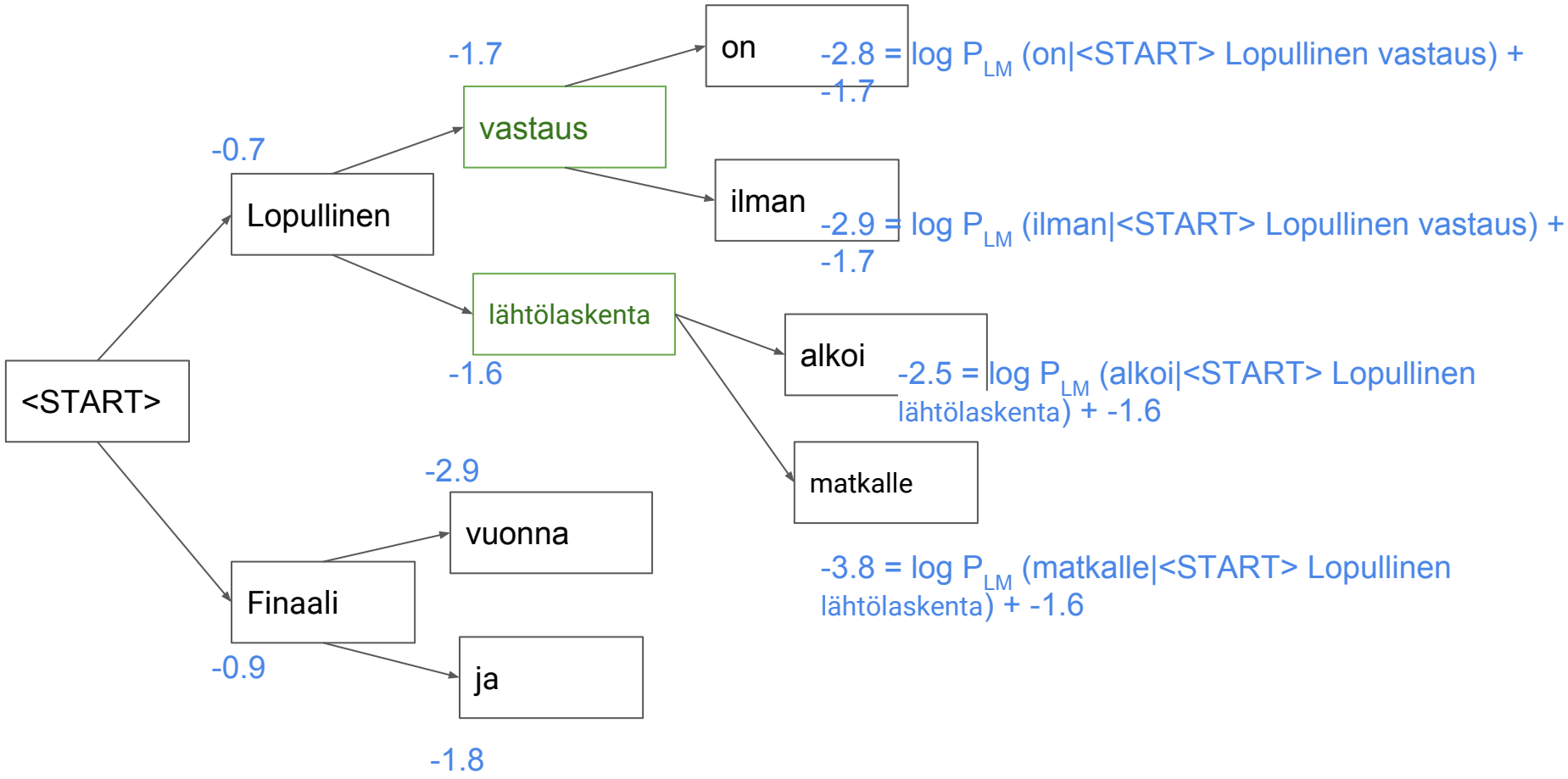
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 *hypotheses*, just keep k with highest scores

Beam search decoding: example

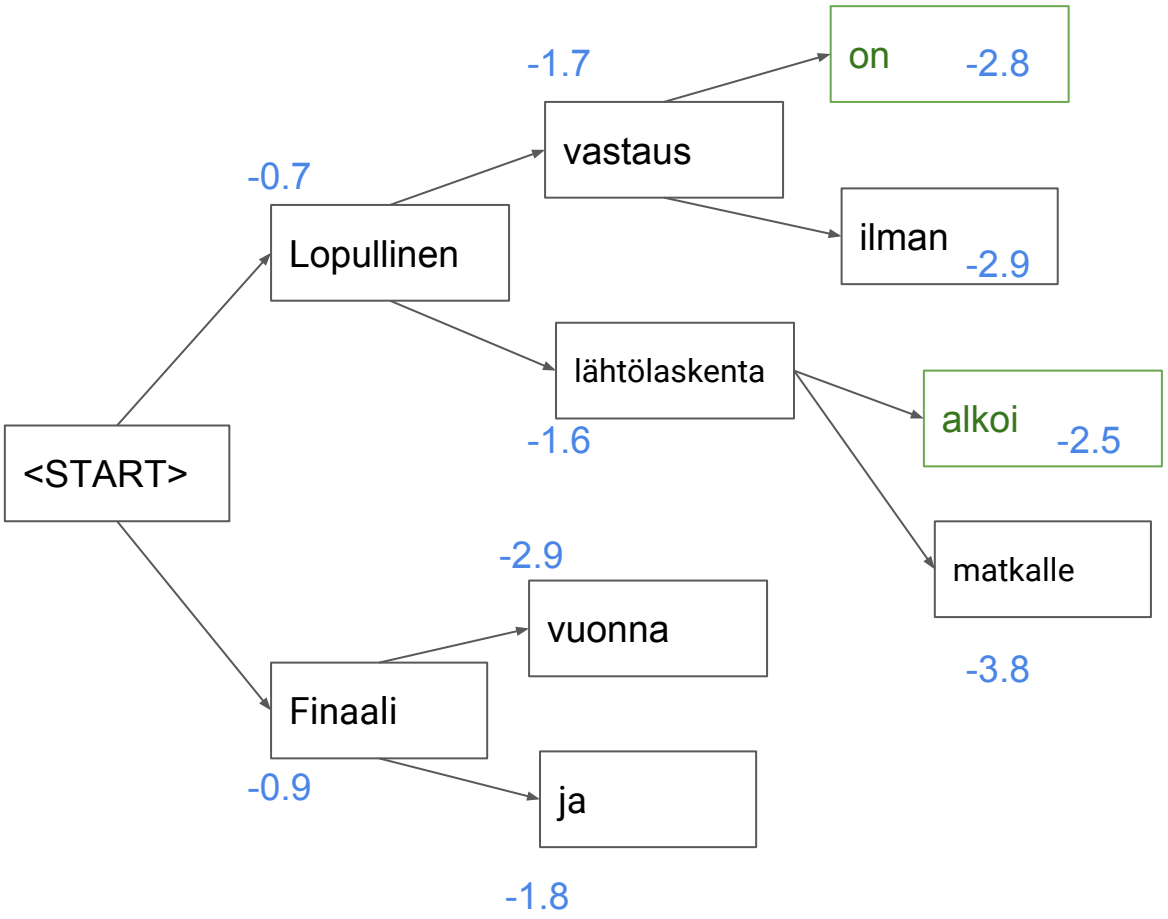
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k hypotheses*, find top *k* next words and calculate scores

Beam search decoding: example

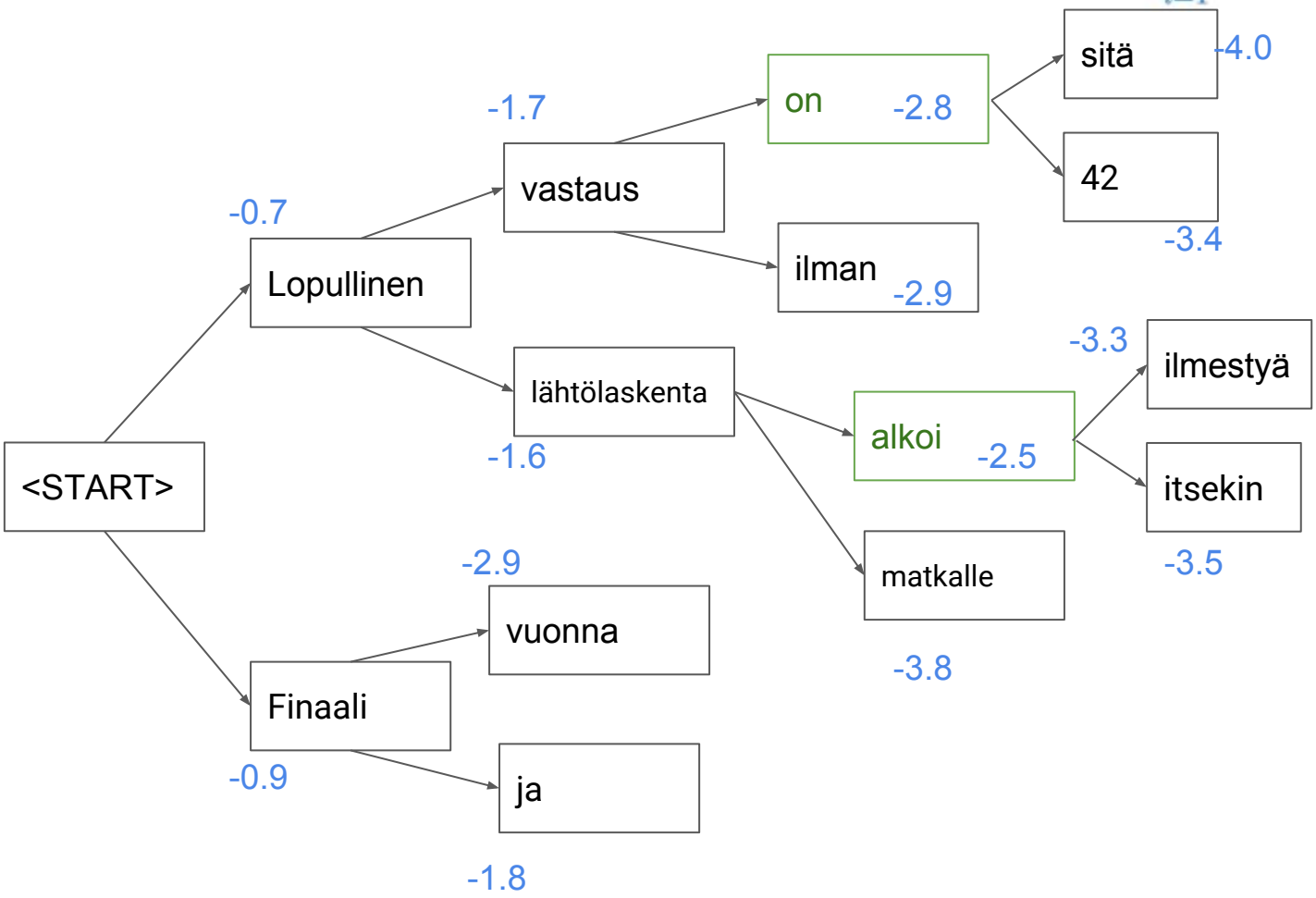
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 *hypotheses*, just keep *k* with highest scores

Beam search decoding: example

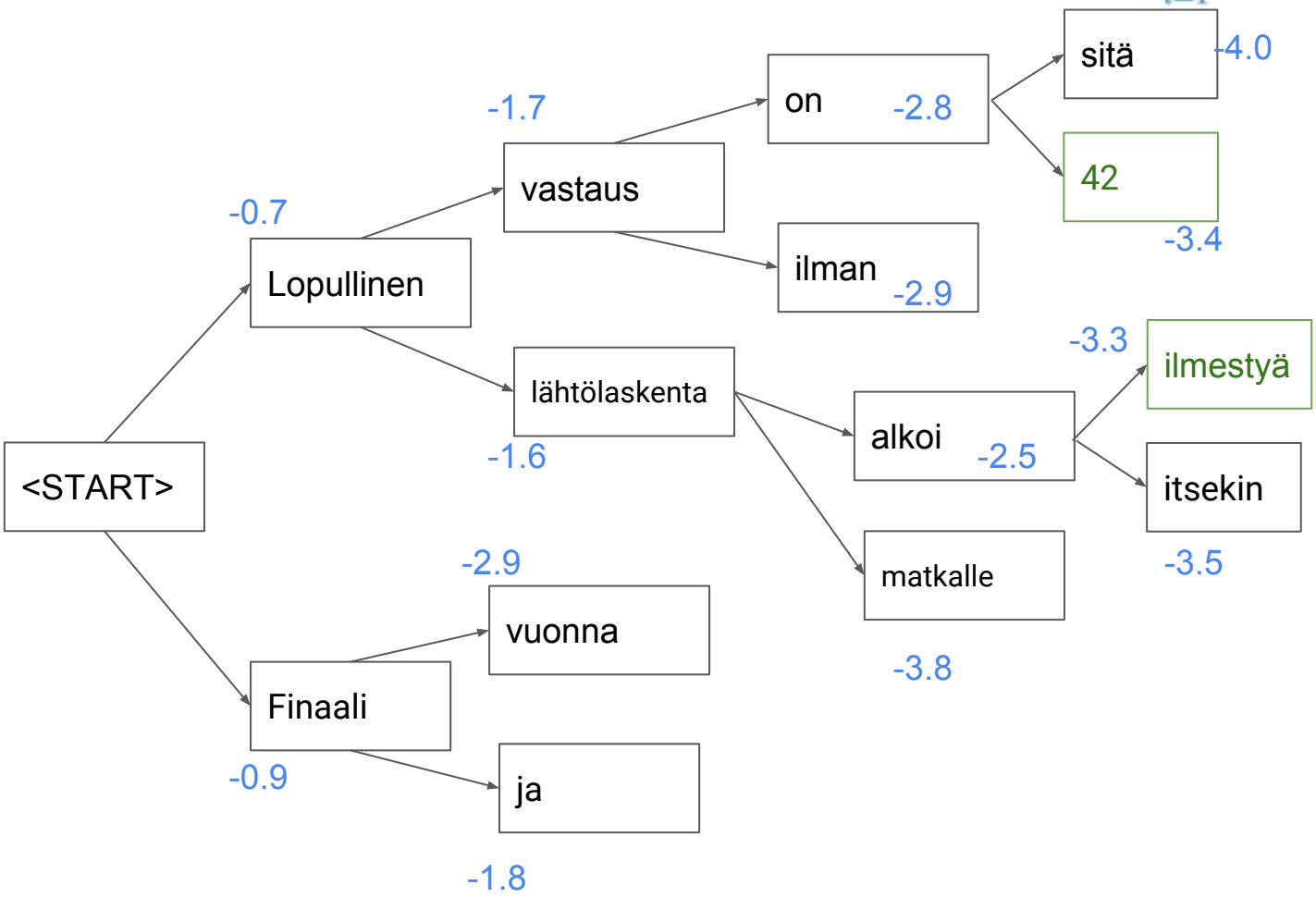
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



For each of the *k hypotheses*, find top *k* next words and calculate scores

Beam search decoding: example

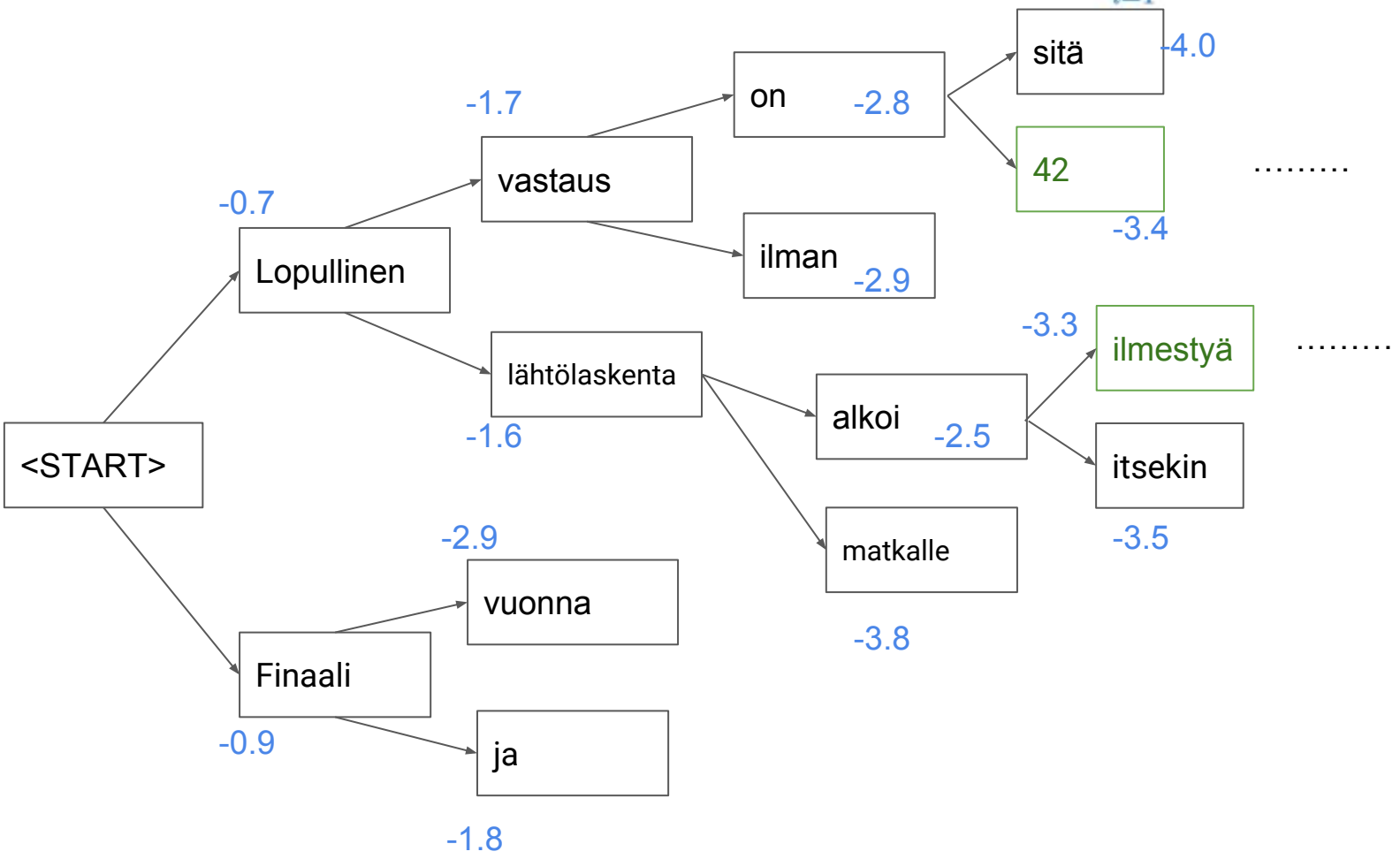
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 *hypotheses*, just keep *k* with highest scores

Beam search decoding: example

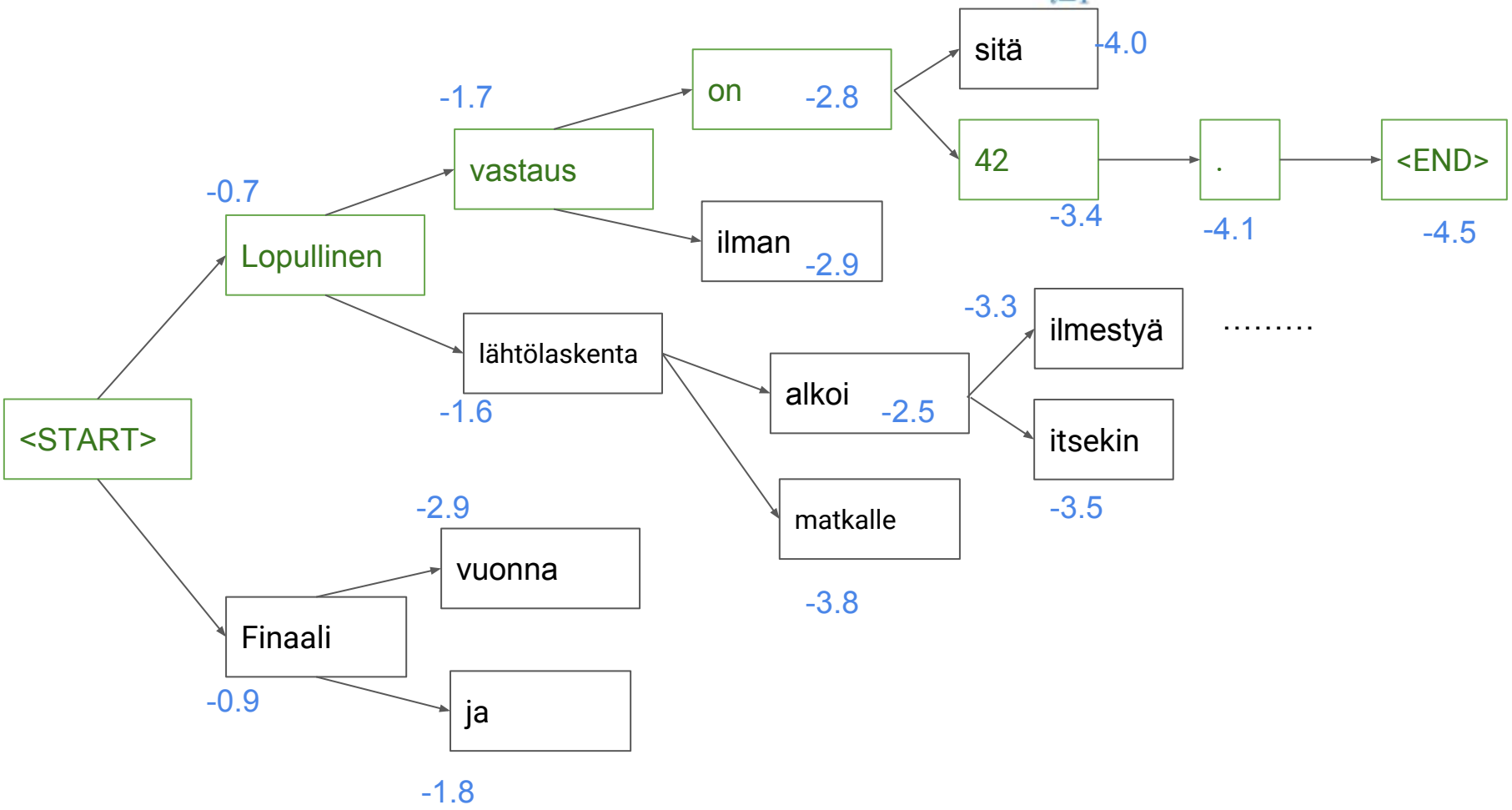
- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Continue until the <END> token

Beam search decoding: example

- Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



This is the top-scoring hypothesis!
Backtrack to obtain the full hypothesis

Beam search decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces a <END> token
  - For example: <START> Lopullinen vastaus on 42 . <END>

- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.

- Usually we continue beam search until:
  - We reach timestep *T* (where *T* is some pre-defined cutoff), or
  - We have at least *n* completed hypotheses (where *n* is pre-defined cutoff)

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?

- Each hypothesis $y_1,\ldots,y_t$ on our list has a score

$$\text{score}(y_1,\ldots,y_t) = \log P_{\text{LM}}(y_1,\ldots,y_t|x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

- Problem with this: longer hypotheses have lower scores

- Fix: Normalize by length. Use this to select top one instead:

$$\frac{1}{t}\sum_{i=1}^{t} \log P_{\text{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

Advantages of NMT

NMT has many qualities:

- Fluent translations
  - Better use of context
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

Disadvantages of NMT

- NMT is difficult to interpret and control
  - Hard to debug
  - can’t easily specify rules or guidelines for translation

- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs
  - Using common sense is still hard
  - NMT picks up biases in training data

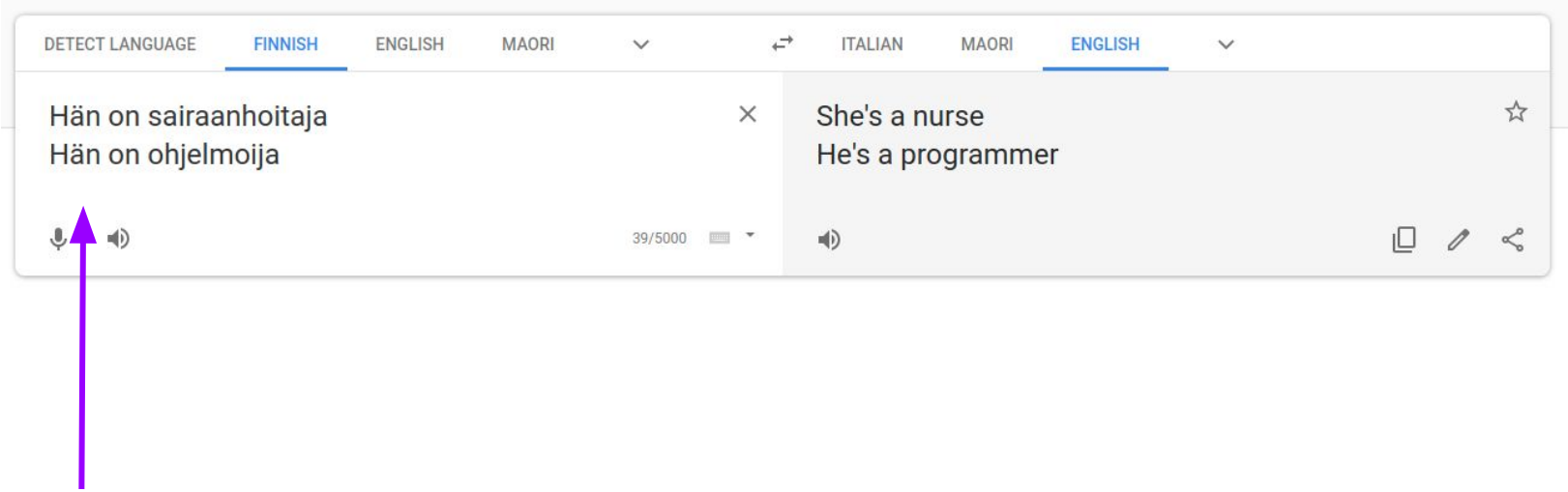
Disadvantages of NMT

- Using common sense is still hard

?

Disadvantages of NMT

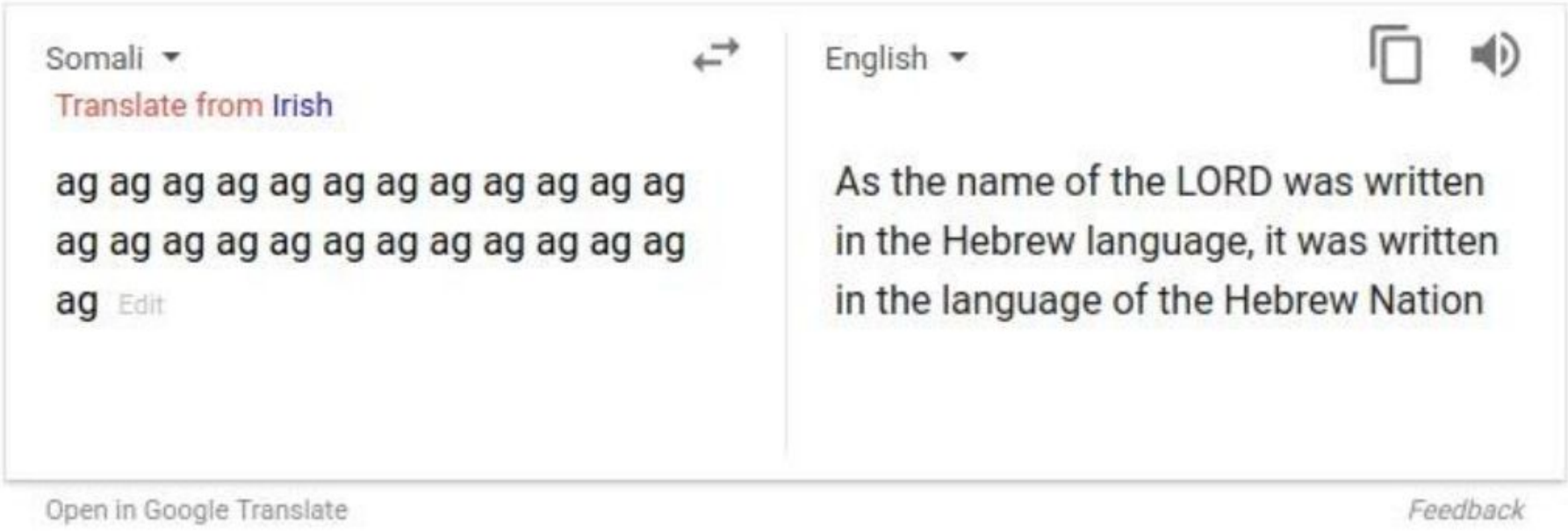
- NMT picks up biases in training data



Didn’t specify gender

Disadvantages of NMT

- Uninterpretable systems do strange things



Picture source:
https://www.vice.com/en_uk/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies
Explanation: https://www.skynettoday.com/briefs/google-nmt-prophecies

Disadvantages of NMT

- Out-Of-Vocabulary (OOV) words

  words not in the vocabulary of the trained NMT model
  - networks have fixed vocabulary
  - poor translation of rare/unknown words

Disadvantages of NMT

- Out-Of-Vocabulary (OOV) words
  words not in the vocabulary of the trained NMT model
  - networks have fixed vocabulary
  - poor translation of rare/unknown words

For example, suppose we want to translate from English to French:

- source sentence:
  “The ecotax portico in Pont-de-Buis was taken down on Thursday morning”
- reference translation:
  “Le portique ecotax de Pont-de-Buis a été démonté jeudi matin”
- machine translation output:
  “Le **unk** de **unk** à **unk** a été pris le jeudi matin”

Underlined words are unknown to the model.
The token **unk** indicates an OOV word.

Example from Luong et. al 2015 https://www.aclweb.org/anthology/P15-1002

Open-Vocabulary Neural Machine Translation

- How do we represent text in NMT?
  - 1-hot encoding:
    - Lookup of word embedding for input
    - Probability distribution over vocabulary for output
  - Large vocabularies
    - Increase network size
    - Decrease training and decoding speed
  - Typical network vocabulary size: 10 000 - 100 000 symbols

vocabulary		representation of "cat"	
		1-hot vector	embedding
0	the	$\begin{bmatrix}0\\1\\0\\.\\0\end{bmatrix}$	$\begin{bmatrix}0.1\\0.3\\0.7\\0.5\end{bmatrix}$
1	cat		
2	is		
.	.		
1024	mat		

Open-Vocabulary Neural Machine Translation

- Translation is an **open-vocabulary** problem:
  - many training corpora contain millions of word types
  - productive word formation processes (compounding; derivation) allow formation and understanding of unseen words
  - names, numbers are morphologically simple, but open word classes

Open-Vocabulary Neural Machine Translation

- Translation is an **open-vocabulary** problem:
  - many training corpora contain millions of word types
  - productive word formation processes (compounding; derivation) allow formation and understanding of unseen words
  - names, numbers are morphologically simple, but open word classes

- Solution 1 - **Back-off Models**:
  1. replace rare words with UNK at training time
  2. when system produces UNK, translate with a back-off method, for example a dictionary

- What are the limitations with this method?

Open-Vocabulary Neural Machine Translation

- Solution 1 - **Back-off Models**:
  1. replace rare words with UNK at training time
  2. when system produces UNK, translate with a back-off method, for example a dictionary

- What are the limitations with this method?
  - **compounds**: hard to model 1-to-many relationships
  - **morphology**: hard to predict inflection with back-off dictionary
  - **names**: if alphabets differ, we need transliteration

- Can we do better?

Open-Vocabulary Neural Machine Translation

- Solution 2 - *wishlist*:
  1. open-vocabulary NMT: encode all words through small vocabulary
  2. encoding generalizes to unseen words
  3. small text size
  4. good translation quality

Open-Vocabulary Neural Machine Translation

- Solution 2 - *wishlist*:
  1. open-vocabulary NMT: encode all words through small vocabulary
  2. encoding generalizes to unseen words
  3. small text size
  4. good translation quality

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Start with a vocabulary of **characters**.
  - Most frequent ngram pairs ↦ a new ngram.
  - *hyperparameter*: when to stop ↦ controls vocabulary size

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Start with a vocabulary of **characters**.
  - Most frequent ngram pairs ↦ a new ngram.
  - *hyperparameter*: when to stop ↦ controls vocabulary size

- Example:

Dictionary

l o w	5
l o w e r	2
n e w e s t	6
w i d e s t	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d

Start with all characters in dictionary

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Start with a vocabulary of **characters**.
  - Most frequent ngram pairs ↦ a new ngram.
  - *hyperparameter*: when to stop ↦ controls vocabulary size

- Example:

Dictionary

l o w	5
l o w e r	2
n e w **es** t	6
w i d **es** t	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Start with a vocabulary of **characters**.
  - Most frequent ngram pairs ↦ a new ngram.
  - *hyperparameter*: when to stop ↦ controls vocabulary size

- Example:

Dictionary

l o w	5
l o w e r	2
n e w **est**	6
w i d **est**	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Start with a vocabulary of **characters**.
  - Most frequent ngram pairs ↦ a new ngram.
  - *hyperparameter*: when to stop ↦ controls vocabulary size

- Example:

Dictionary

lo w	5
lo w e r	2
n e w est	6
w i d est	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Automatically decide vocabs for NMT
  - Open-vocabulary: operations learned on training set can be applied to unknown words
  - compression of frequent character sequences improves efficiency
  - trade-off between text length and vocabulary size

Dictionary

lo w	5
lo w e r	2
n e w est	6
w i d est	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, lo

e s → es
es t → est
l o → lo

- Suppose we have the following new word:
  l o w e s t

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Automatically decide vocabs for NMT
  - Open-vocabulary: operations learned on training set can be applied to unknown words
  - compression of frequent character sequences improves efficiency
  - trade-off between text length and vocabulary size

Dictionary

lo w	5
lo w e r	2
n e w est	6
w i d est	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, lo

e s → **es**
es t → est
l o → lo

- Suppose we have the following new word:
  l o w **es** t

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Automatically decide vocabs for NMT
  - Open-vocabulary: operations learned on training set can be applied to unknown words
  - compression of frequent character sequences improves efficiency
  - trade-off between text length and vocabulary size

Dictionary

lo w	5
lo w e r	2
n e w est	6
w i d est	3

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, lo

e s → es
es t → est
l o → lo

- Suppose we have the following new word:
  l o w **est**

Subword units - Byte Pair Encoding (BPE)

- **Subword units - Byte Pair Encoding (BPE)** for word segmentation**:**
  - Automatically decide vocabs for NMT
  - Open-vocabulary: operations learned on training set can be applied to unknown words
  - compression of frequent character sequences improves efficiency
  - trade-off between text length and vocabulary size

Dictionary

lo w	5
lo w e r	2
n e w est	6
w i d est	3

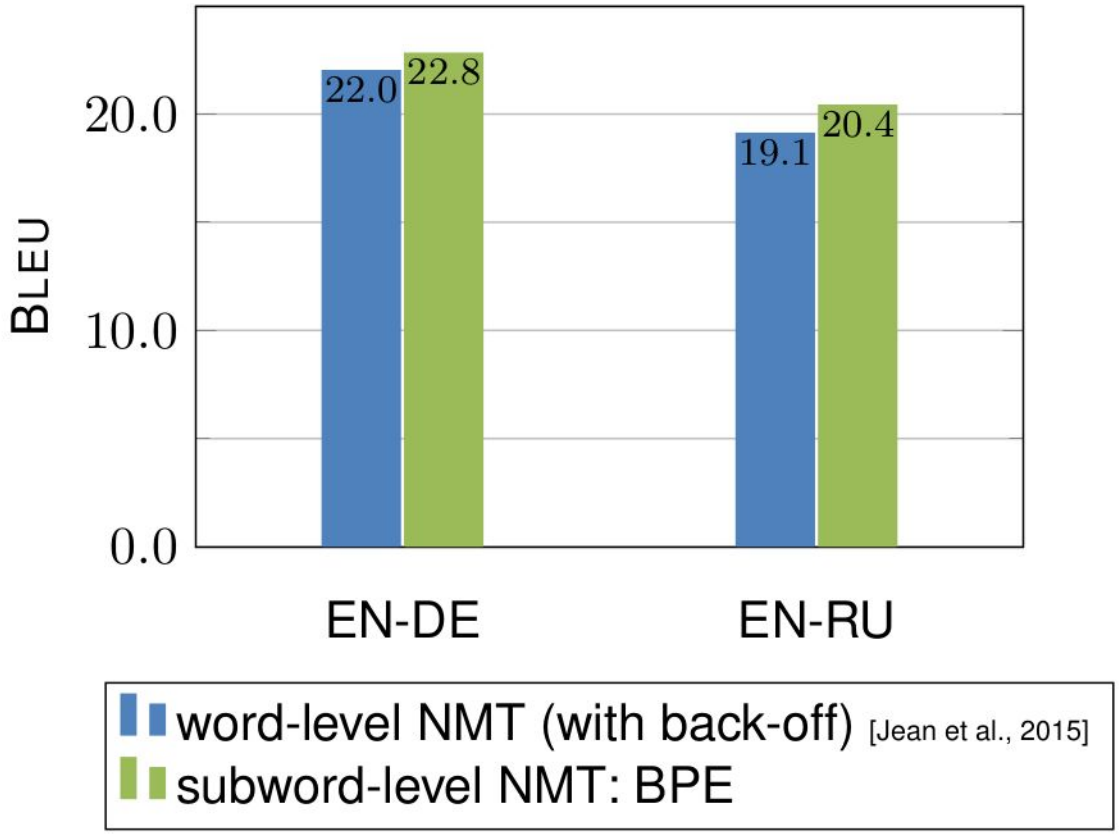
Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, lo

e s → es
es t → est
l o → lo

- Suppose we have the following new word:
  lo w est

Byte Pair Encoding (BPE)
Translation quality



Byte Pair Encoding (BPE) Examples

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
word-level (with back-off)	Forschungsinstitute
BPE	Gesundheits|forsch|ungsin|stitute
source	rakfisk
reference	ракфиска (rakfiska)
word-level (with back-off)	rakfisk → UNK → rakfisk
BPE	rak|f|isk → рак|ф|иска (rak|f|iska)

Byte Pair Encoding (BPE)

- BPE-level subword segmentation is currently the most widely used technique for open-vocabulary NMT
- BPE allows open vocabulary
  - how well it generalizes is still an open question

- Segmentation Variants:
  - morphologically motivated subword units [Sánchez-Cartagena and Toral, 2016, Tamchyna et al., 2017, Huck et al., 2017, Pinnis et al., 2017]
  - probabilistic segmentation and sampling [Kudo, 2018]
  - fully character-level Models [Ling et al. 2015, Lee et al. 2016]

Readings

- Neural Translation Models, Encoder-Decoder approach, Training and Beam Search:
  Philipp Koehn (2017)
  Neural machine translation. Section: 13.5
  https://arxiv.org/pdf/1709.07809.pdf

- Encoder-Decoder approach:
  Sutskever, I., Vinyals, O., and Le, Q. V. (2014).
  Sequence to Sequence Learning with Neural Networks.
  https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

- Byte Pair Encoding (BPE):
  Sennrich, R., Haddow, B., and Birch, A. (2016).
  Neural Machine Translation of Rare Words with Subword Units.
  https://www.aclweb.org/anthology/P16-1162

Readings

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015).
Character-based Neural Machine Translation.
https://arxiv.org/pdf/1511.04586.pdf

Lee, J., Cho, K., and Hofmann, T. (2016).
Fully Character-Level Neural Machine Translation without Explicit Segmentation.
https://aclweb.org/anthology/Q17-1026

Kudo, T. (2018).
Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates.
https://aclweb.org/anthology/P18-1007

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, Wojciech Zaremba (2015)
Addressing the Rare Word Problem in Neural Machine Translation
https://www.aclweb.org/anthology/P15-1002

Tamchyna, A., Weller-Di Marco, M., and Fraser, A. (2017).
Modeling Target-Side Inflection in Neural Machine Translation.
https://www.aclweb.org/anthology/W17-4704

Sánchez-Cartagena, V. M. and Toral, A. (2016).
Abu-MaTran at WMT 2016 Translation Task: Deep Learning, Morphological Segmentation and Tuning on Character Sequences.
https://www.aclweb.org/anthology/W16-2322

Huck, M., Riess, S., and Fraser, A. (2017).
Target-side Word Segmentation Strategies for Neural Machine Translation.
https://www.statmt.org/wmt17/pdf/WMT06.pdf

Pinnis, M., Krislauks, R., Deksne, D., and Miks, T. (2017).
Neural Machine Translation for Morphologically Rich Languages with Improved Sub-word Units and Synthetic Data.
https://link.springer.com/chapter/10.1007/978-3-319-64206-2_27

Stig-Arne Grönroos, Sami Virpioja, Mikko Kurimo (2018).
Cognate-aware morphological segmentation for multilingual neural translation.
https://aclweb.org/anthology/W18-6410