School of CIT –
Social Computing
Research Group

Technical
University
of Munich

Natural Language Processing
IN2361

Prof. Dr. Georg Groh

Chapter 10
Machine Translation and Encoder-Decoder Models

- content is based on [1] and [2]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1] and [2]
- citations of [1] and [2] or from [1] and [2] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!
- BIG thanks to Richard and all the guys at Stanford for excellent lecture materials!

Machine Translation

- task: translate sentence from source language to target language

L'homme est né libre, et partout il est dans les fers

⇩

Man is born free, but everywhere he is in chains

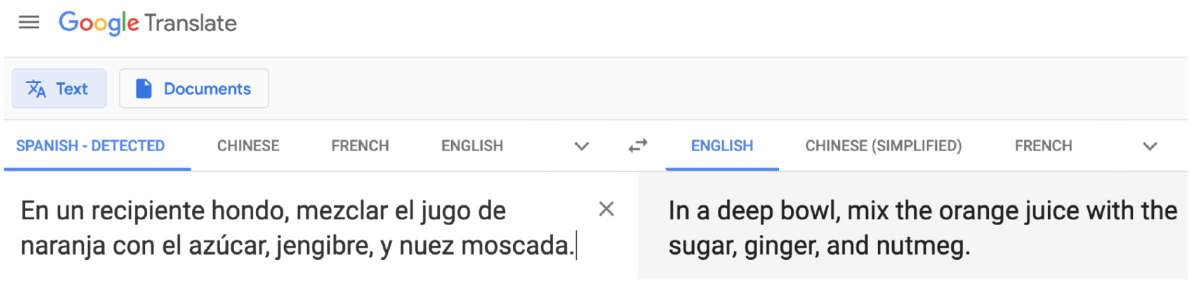
- 1950- … : early machine translation: rule-based systems based on bilingual dictionaries

[3]

Machine Translation

Use cases:

- Information access: translate instructions on web (e.g. recipes, newspaper articles or Wikipedia articles)



- Post-editing: produce draft translation for post-editing by human translator (computer-aided translation (CAT))
- Localization: adapt content to particular language community
- In-the-moment communication: incremental translation, translating speech on-the-fly, image-centric translation

Machine Translation: Statistical Models

- 1990s-2010s: core idea: learn probabilistic model from data

- find best sentence y in target language given a source language sentence x:

$$\text{argmax}_y P(y|x)$$

$$= \text{argmax}_y P(x|y)P(y)$$

Translation Model

Models how words and phrases should be translated.
Learnt from parallel data.

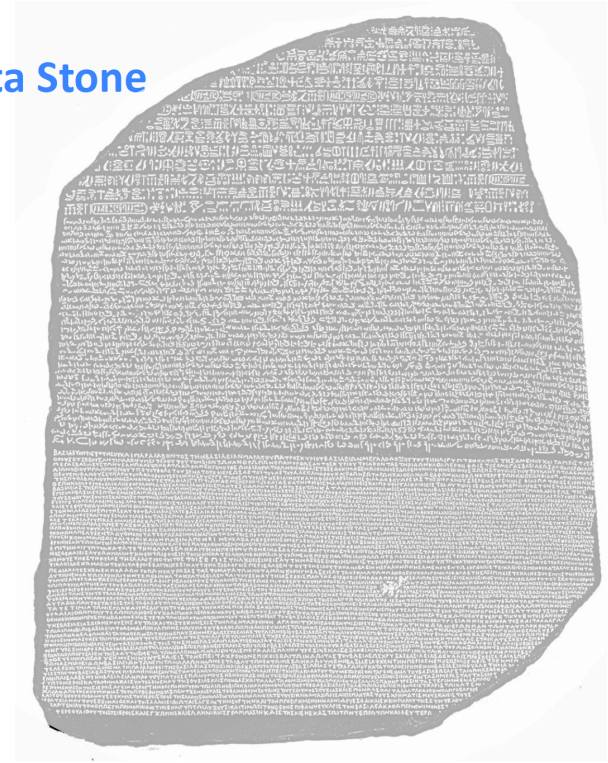
Language Model

Models how to write good English.
Learnt from monolingual data.

Machine Translation: Statistical Models

- translation model $P(x|y)$: large amounts of parallel data required

The Rosetta Stone



Ancient Egyptian

Demotic

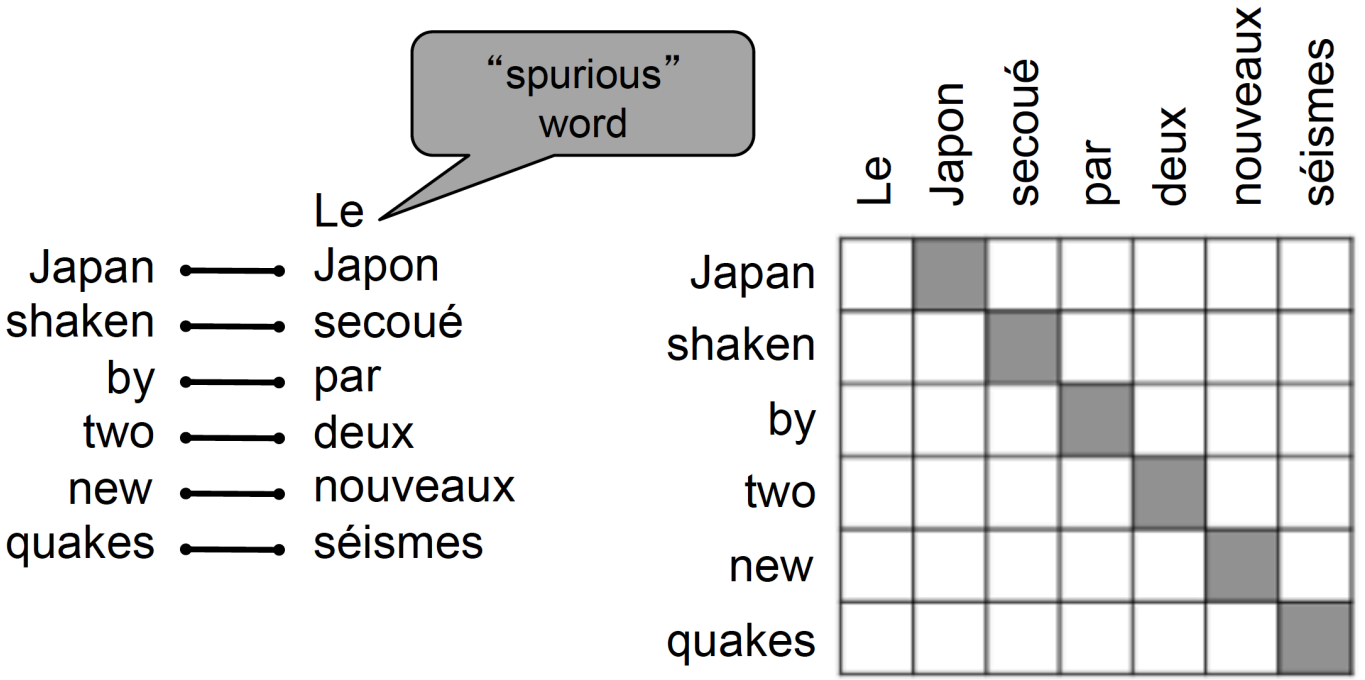
Ancient Greek

- more accurately, we need to learn $P(x, a|y)$, where a is an alignment between source sentence x and target sentence y

Alignment

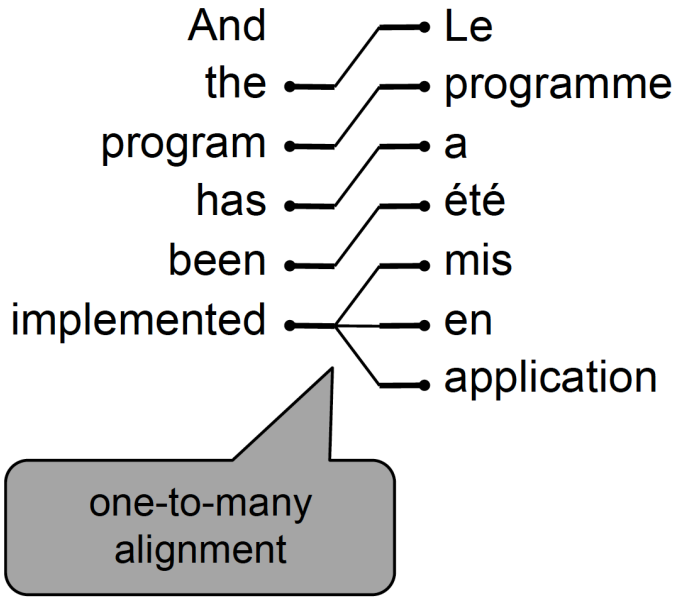
- alignment: correspondence between particular words in translated sentence pair

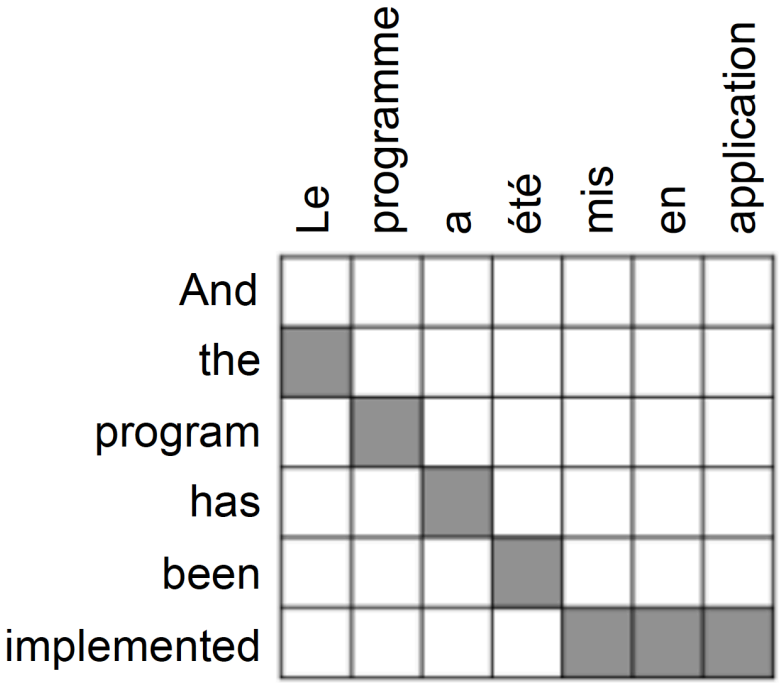
- some words may have no counterpart



Alignment

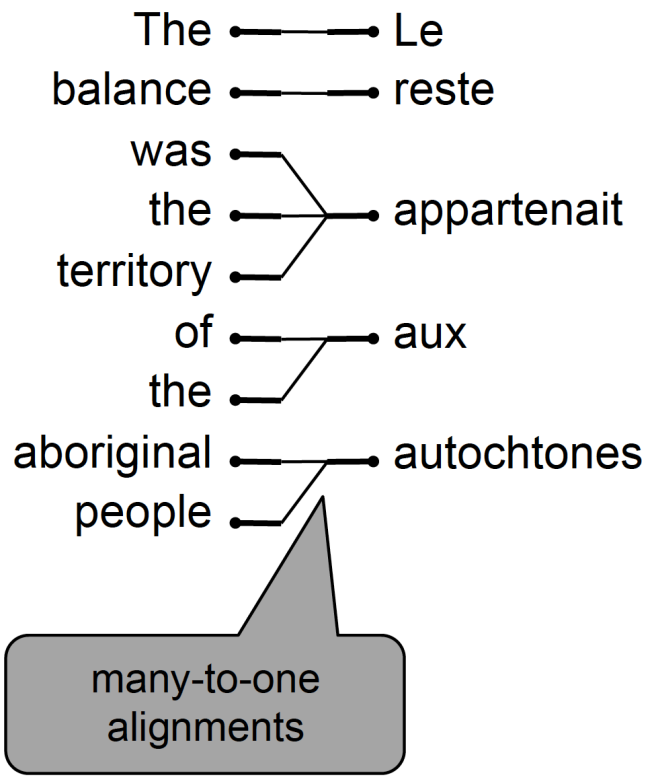
can be one-to-many (“fertile” words)

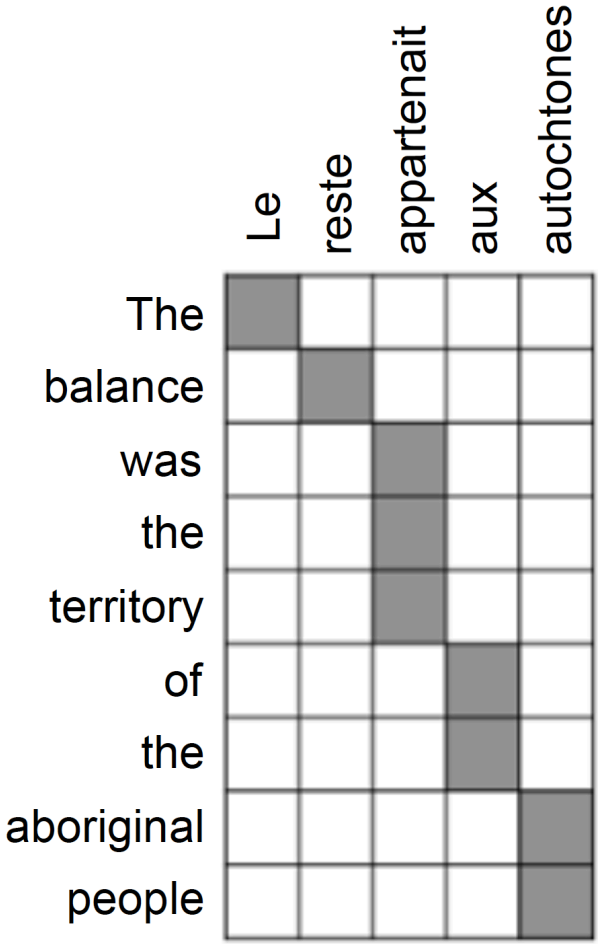




Alignment

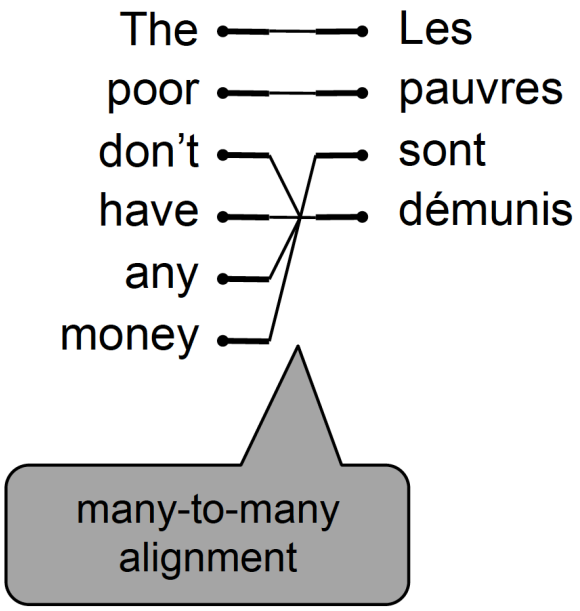
or many-to-one

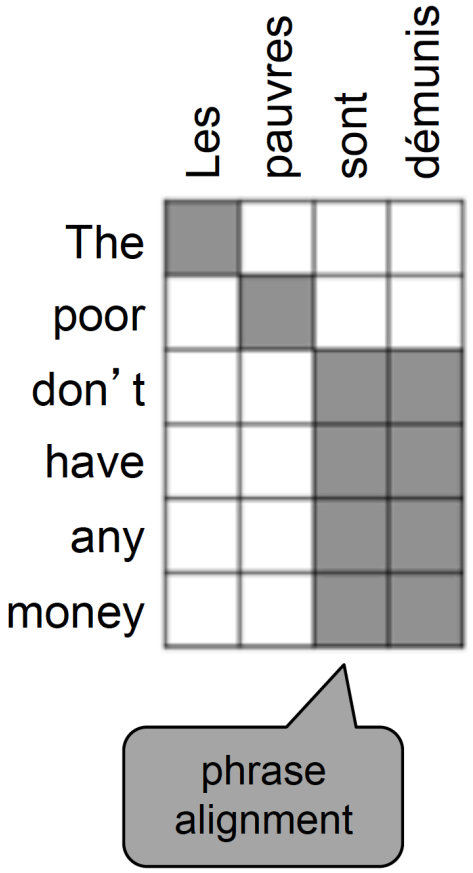




Alignment

or many-to-many (phrase level)





Statistical Models

- best systems extremely complex
- lots of important details
- many separately-designed subcomponents
- lots of feature engineering
  - need to design features to capture particular language phenomena
- require compiling and maintaining extra resources: e.g. tables of equivalent phrases
- → lots of human effort to maintain for each language pair

→ use **Neural Machine Translation (NMT)**

Sequence to Sequence (seq2seq)/ Encoder-Decoder Networks

- In previous chapters:
  - Map input sequence to single score (e.g. sentiment classification)
  - Sequence labelling: map each word to label (e.g. part-of-speech labelling)
- Seq2seq/Encoder-Decoder Networks: Output is complex function of entire input → no direct mapping between words
- Seq2seq in Machine translation:
  - English: *He wrote a letter to a friend*
    Japanese: *tomodachi ni tegami-o kaita*
    friend to letter wrote
  - 大会/General Assembly 在/on 1982年/1982 12月/December 10日/10 通过了/adopted 第37号/37th 决议/resolution， 核准了/approved 第二次/second 探索/exploration 及/and 和平peaceful 利用/using 外层空间/outer space 会议/conference 的/of 各项/various 建议/suggestions 。

    On 10 December 1982 , the General Assembly adopted resolution 37 in which it endorsed the recommendations of the Second United Nations Conference on the Exploration and Peaceful Uses of Outer Space .

Encoder-Decoder Networks

- In general: Encoder-decoder networks solve seq2seq problems, such as:
  - Machine translation
  - Summarization (map long text to short)
  - Dialogue generation (map user input to system output)
  - Semantic parsing (map string of words to semantic representation)
  - …..

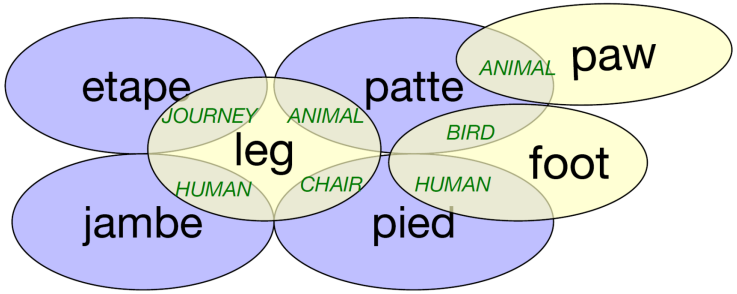
Language Divergences and Typology

- Models need to consider linguistic differences among languages

- Word Order Typology
  - Position of verb compared to subject/object
  - Adjective before/after noun

S V O
English: *He wrote a letter to a friend*
S O V
Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote
V S O
Arabic: *katabt risāla li ṣadq*
wrote letter to friend

- Lexical Divergences
  - Words with multiple meanings
  - Lexical gap: specific word not existing

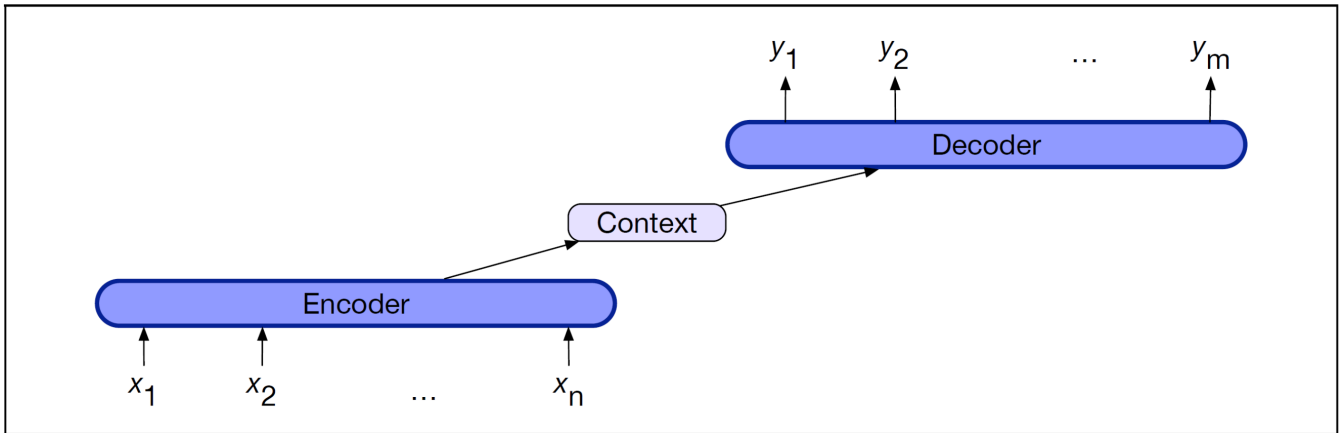


- Referential density: Languages with implicit pronouns and references

[El jefe]$_i$ dio con un libro. $\emptyset_i$ Mostró a un descifrador ambulante.
[The boss] came upon a book. [He] showed it to a wandering decoder.

Encoder-Decoder Networks

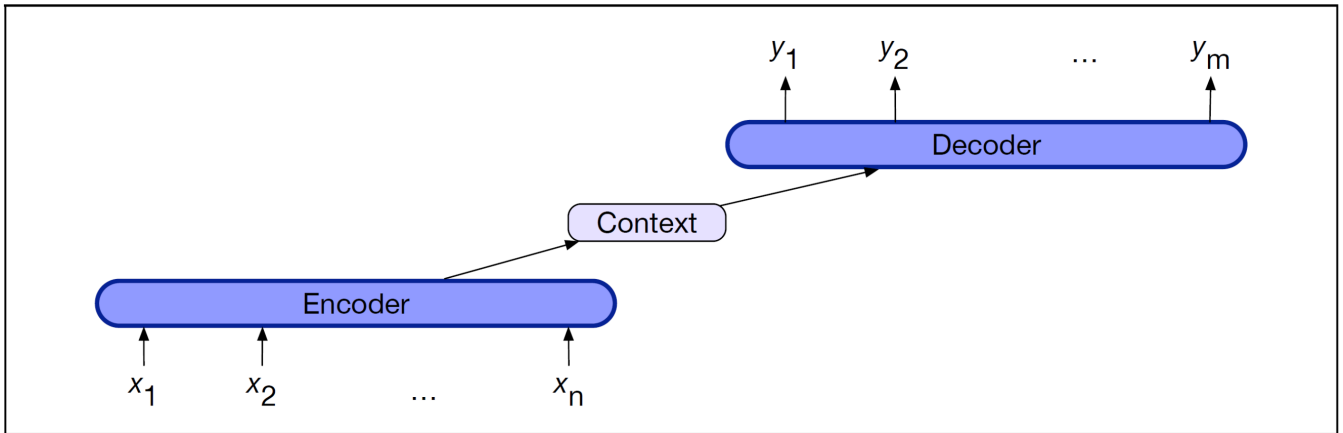
- models capable of generating contextually appropriate, arbitrary length, output sequences
- 3 parts:
  - Encoder: accepts input sequence x_1^n
    → generates contextualized representation h_1^n
  - Context vector c: function of h_1^n, conveys essence of input to decoder
  - Decoder: accepts input c
    → generates hidden states h_1^m, from which output sequence y_1^m can be obtained
- Can use RNNs / LSTMs, CNNs or Transformer architectures



Encoder-Decoder Networks

- models capable of generating contextually appropriate, arbitrary length, output sequences
- 3 parts:
  - Encoder: accepts input sequence x_1^n
    → generates contextualized representation h_1^n
  - Context vector c: function of h_1^n, conveys essence of input to decoder
  - Decoder: accepts input c
    → generates hidden states h_1^m, from which output sequence y_1^m can be obtained
- Can use RNNs / LSTMs, CNNs or Transformer architectures

Older notation from Jurafsky: $x_1^n == x_{1:n}$



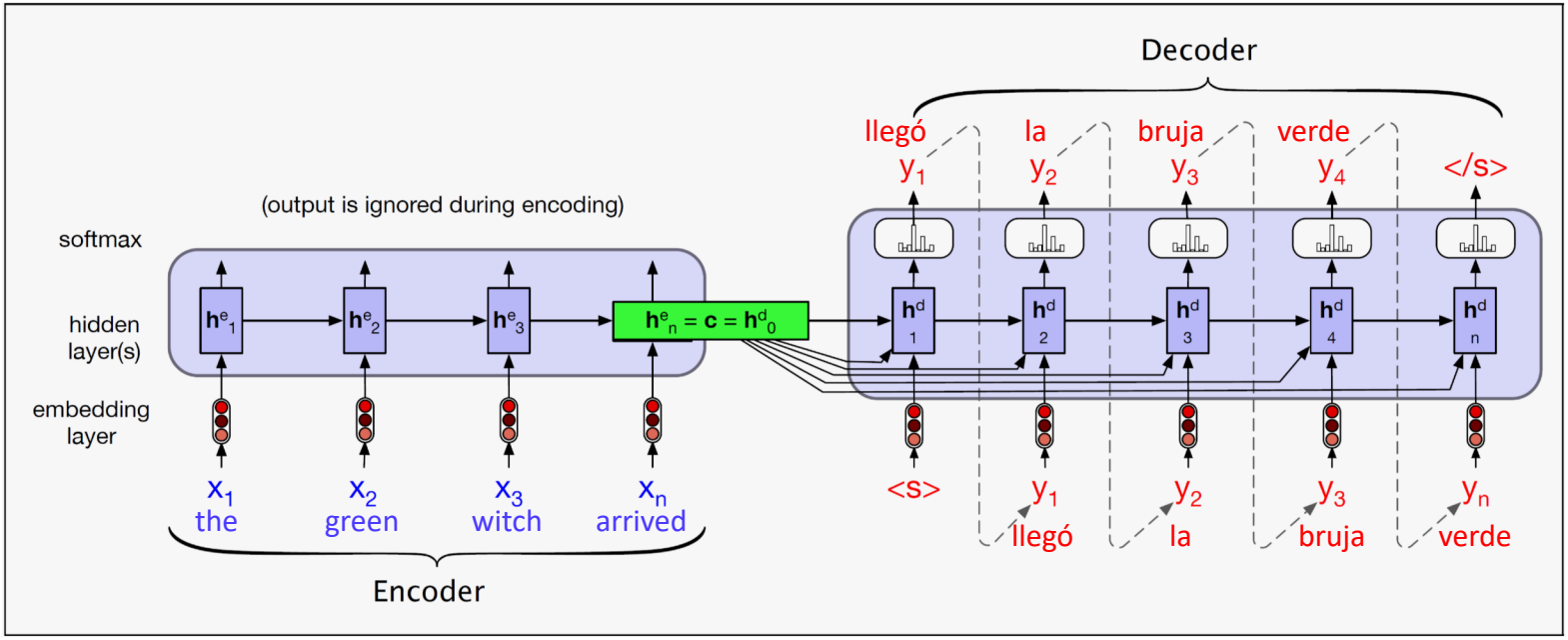
Encoder-Decoder with RNNs

- Recap: Autoregressive language model:

$$p(y) = p(y_1)p(y_2|y_1)p(y_3|y_1, y_2) \dots p(y_m|y_1, \dots, y_{m-1})$$

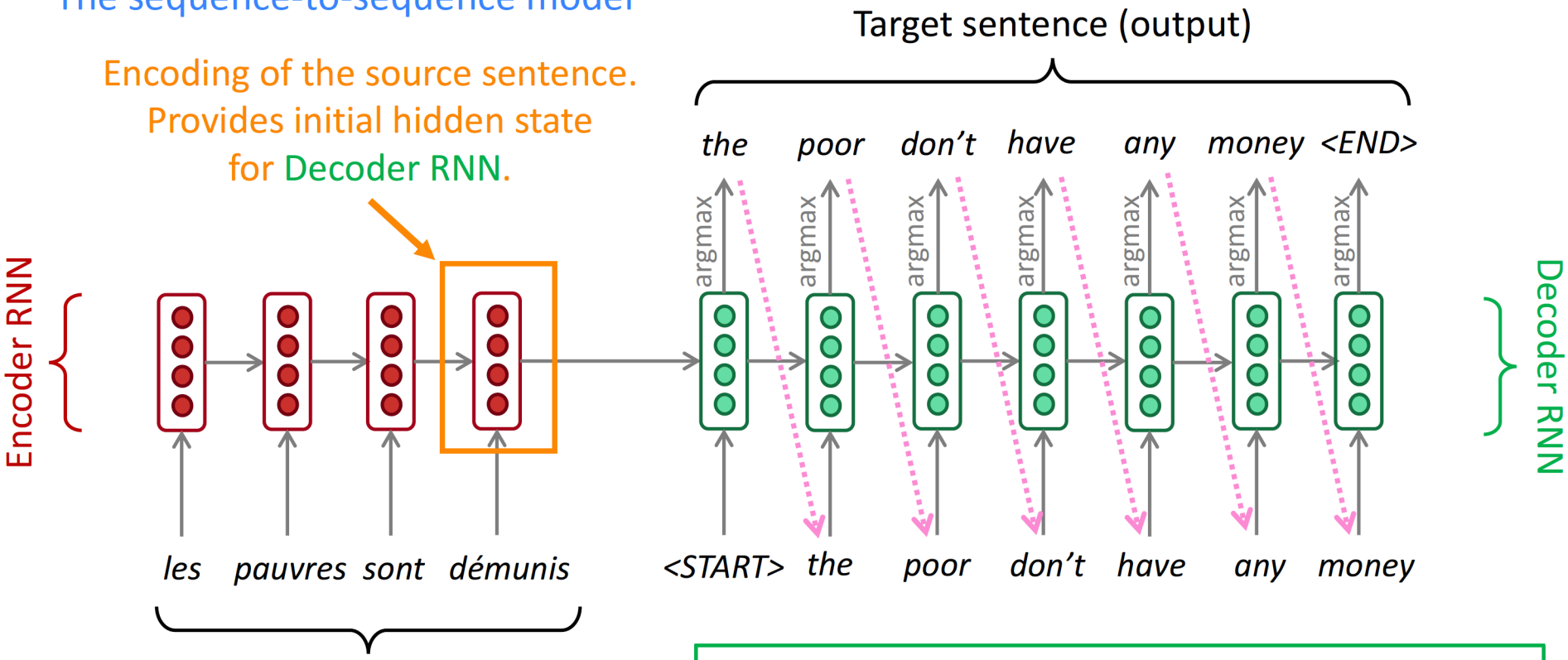
- Adaptions for translation model from source to target language:
  - add sentence separation at end of source text
  - Conditional language model: condition on source text

→ $p(y|x) = p(y_1|x)p(y_2|y_1, x)p(y_3|y_1, y_2, x) \dots p(y_m|y_1, \dots, y_{m-1}, x)$



Encoder-Decoder with RNNs

The sequence-to-sequence model



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

Encoder RNN produces an encoding of the source sentence.

Note: This diagram shows **test time** behavior: decoder output is fed in ⋯→ as next step's input

Training Encoder-Decoder Networks

- Training data: tuples of source-target pairs
- Approach
  - Give source text to model
  - Starting with separator token: train autoregressively to predict next target word
  - Calculate loss for each token
  - Average token losses for sentence level loss

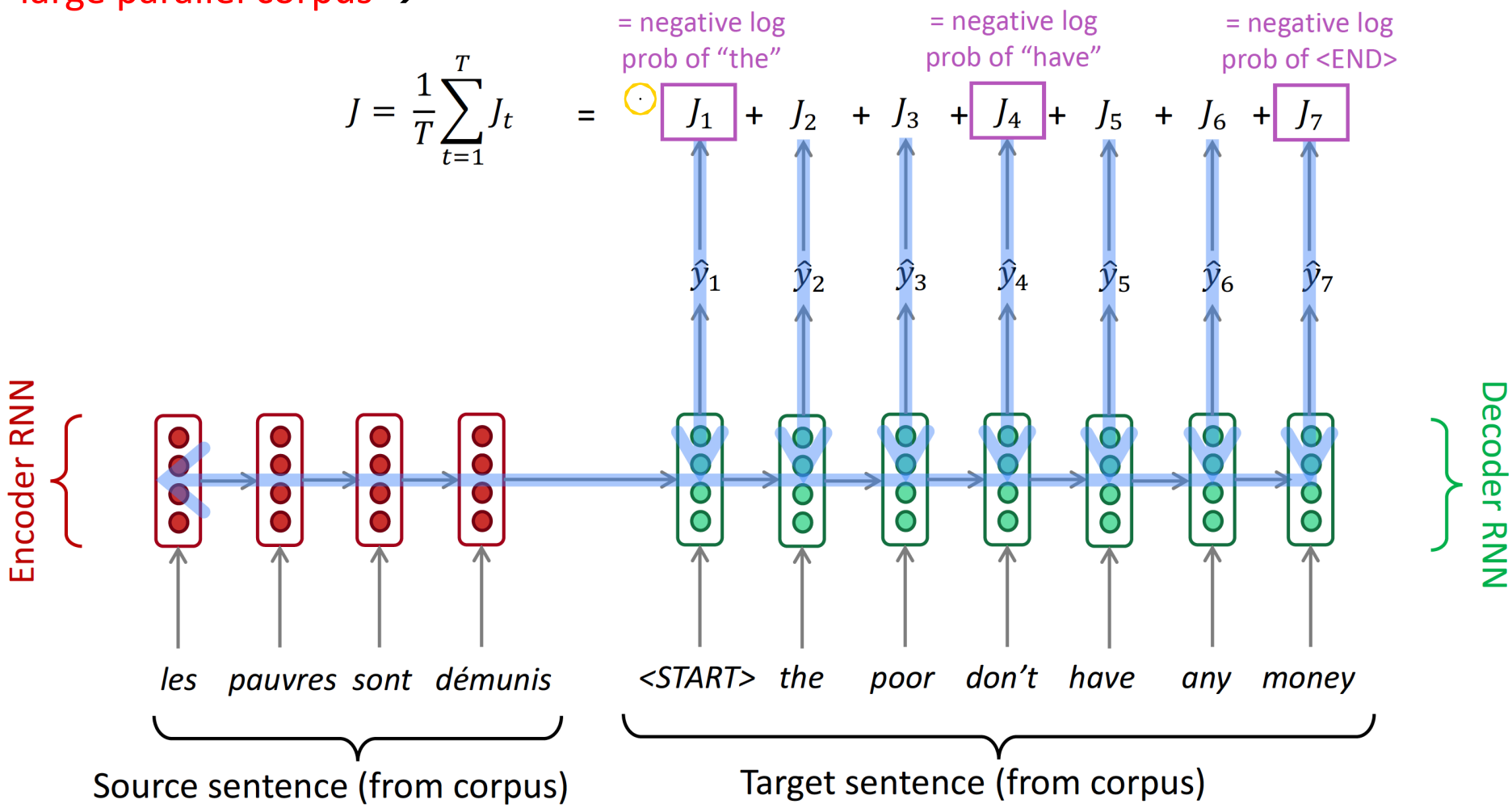
- Teacher forcing
  - Inference: Previous prediction as input for next word
  - Training: Gold target token instead of prediction as input

  → not propagate wrong predictions

  → speed up training

Training Encoder-Decoder Networks

large parallel corpus →

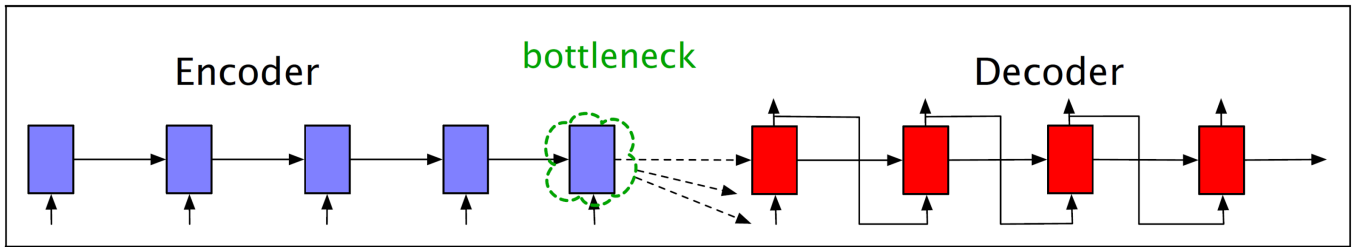


Seq2seq is optimized as a **single system.**
Backpropagation operates *"end to end"*.

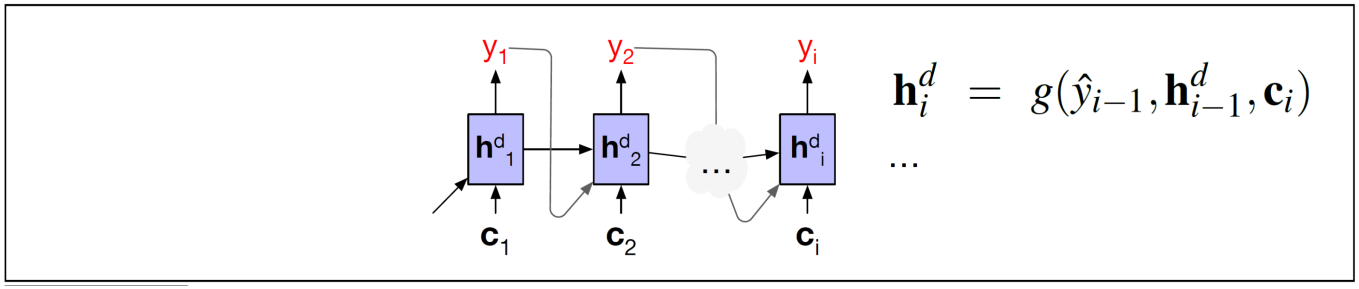
Attention

last encoder state need to represent all information in source sentence

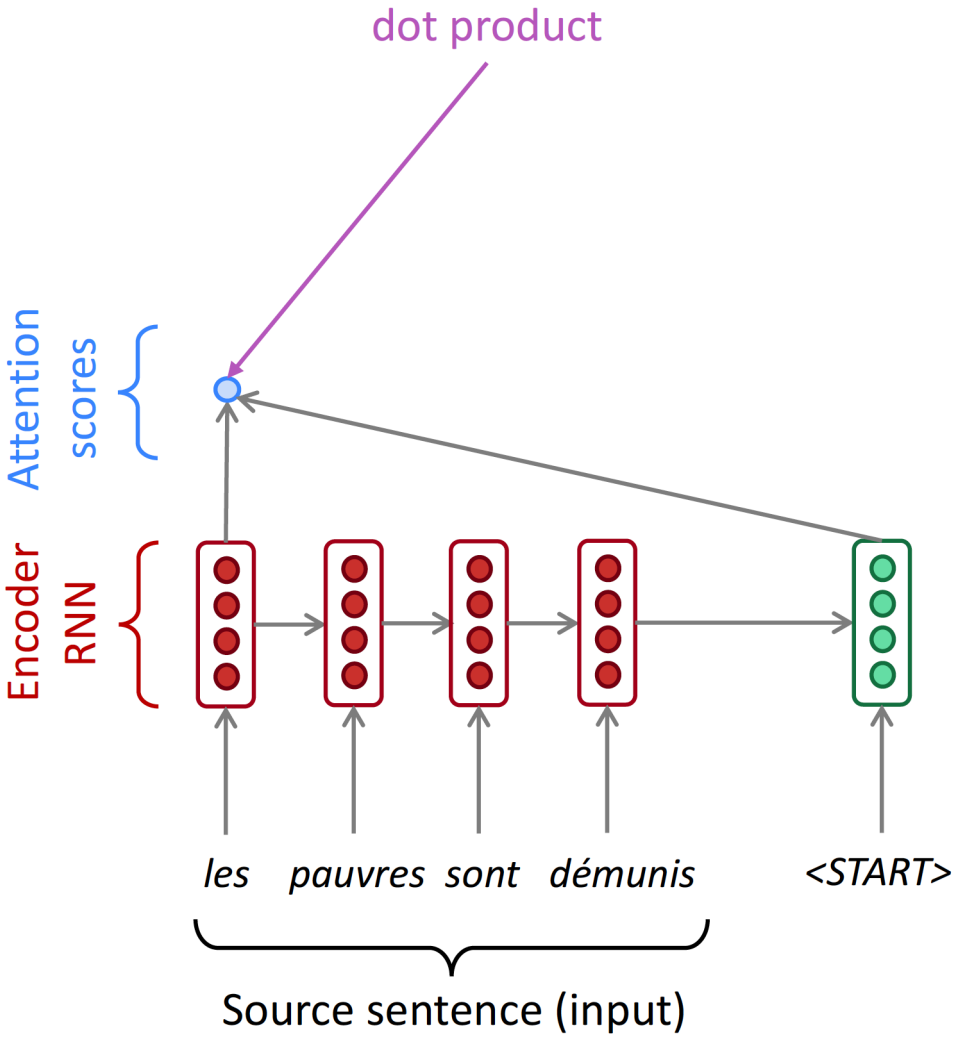
- Bottleneck problem:
  - Encoder final state only info about source sentence for decoder
  - → must encode ALL information in one neuron



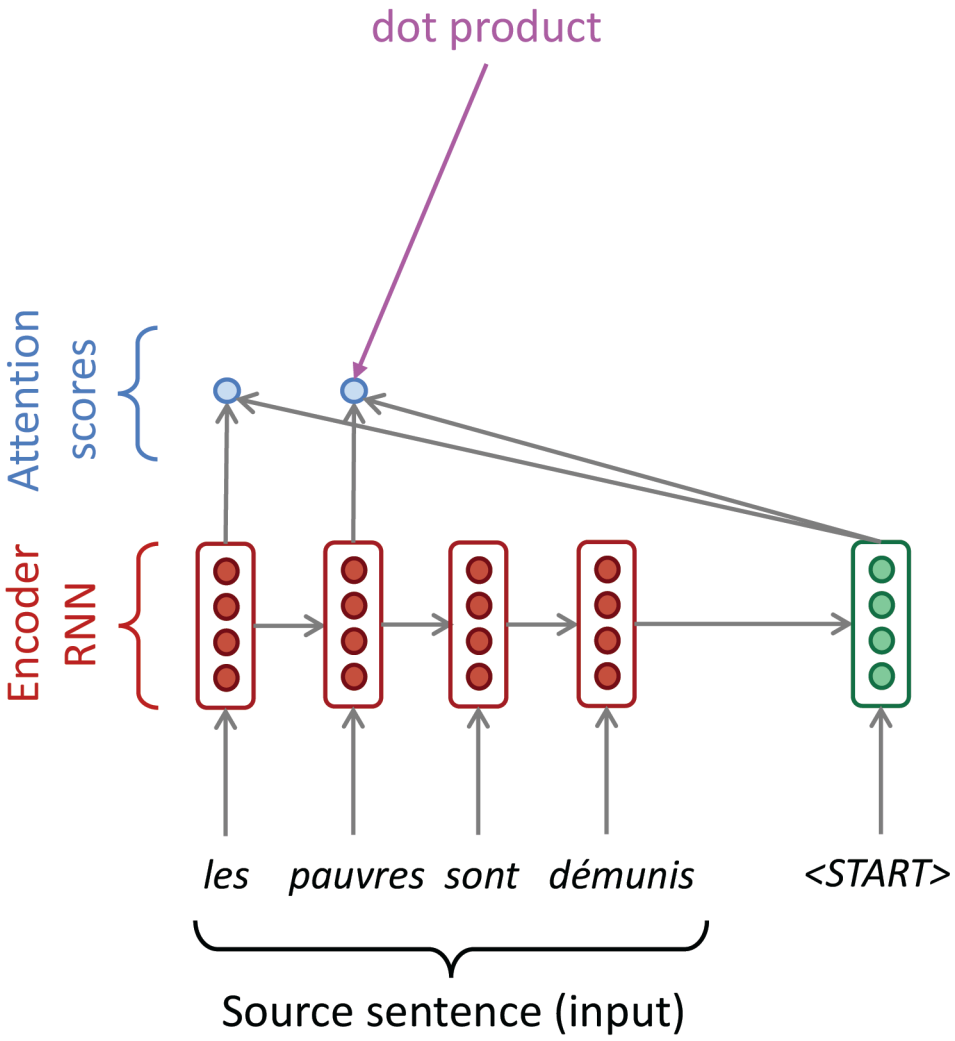
- Attention mechanism: Include all encoder hidden states in context vector
  - Weighted sum of encoder hidden states
    → focus on relevant part in source text
  - Context vector different for each decoding step



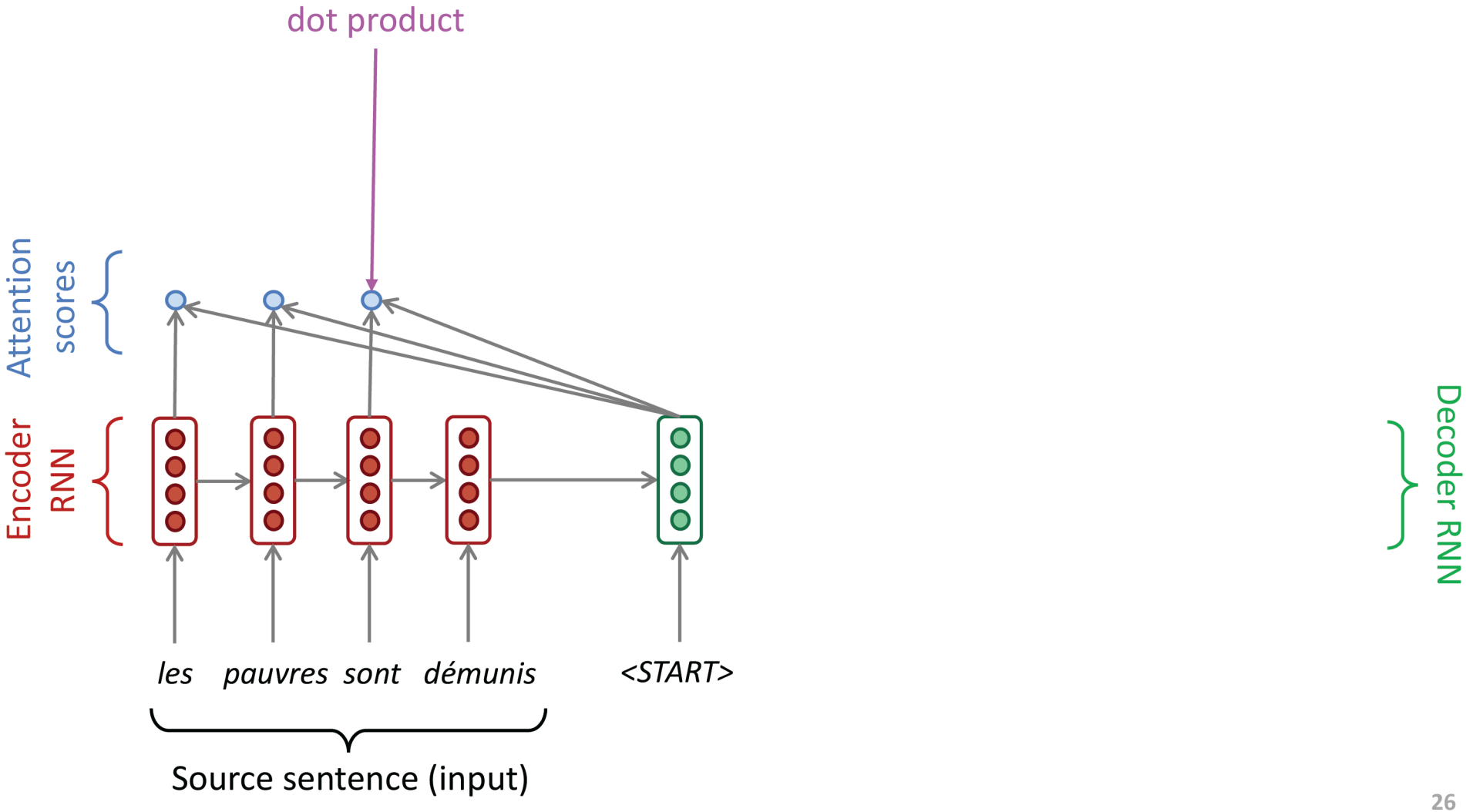
Dot-Product Attention



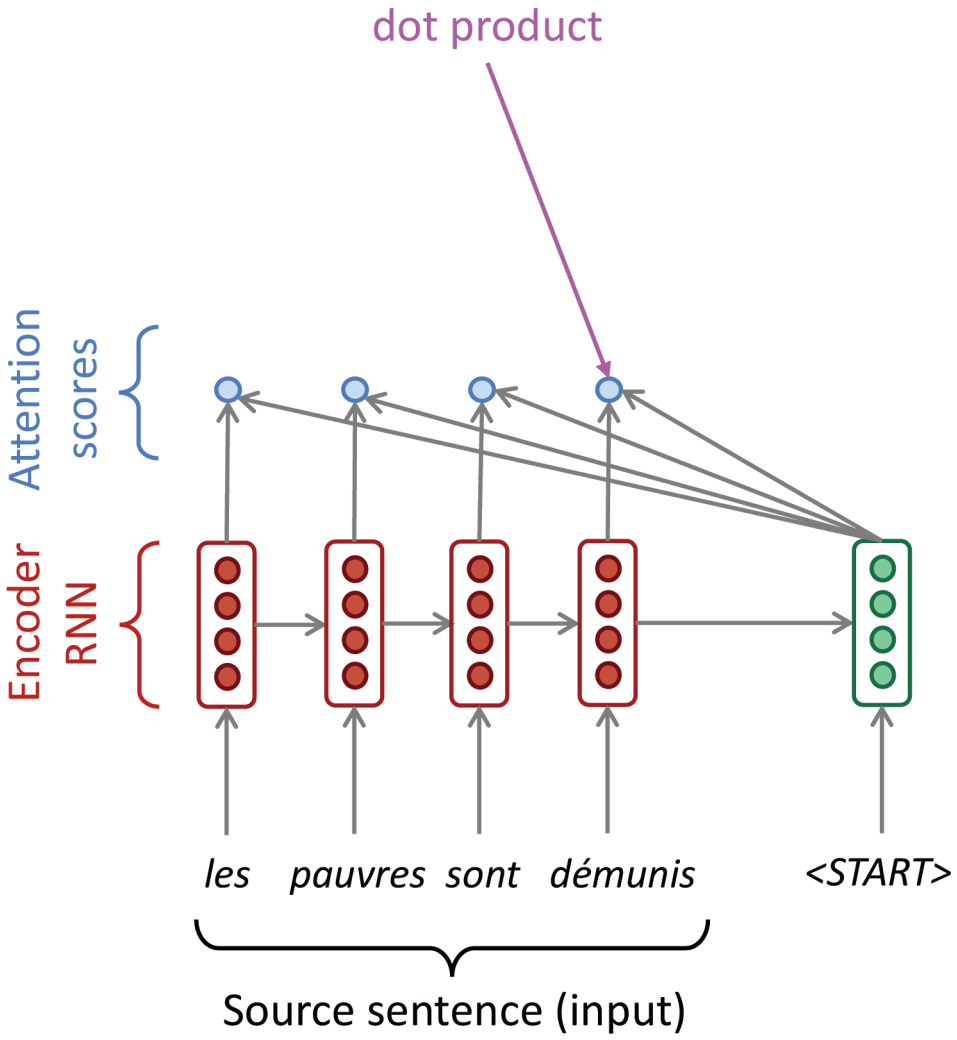
Dot-Product Attention



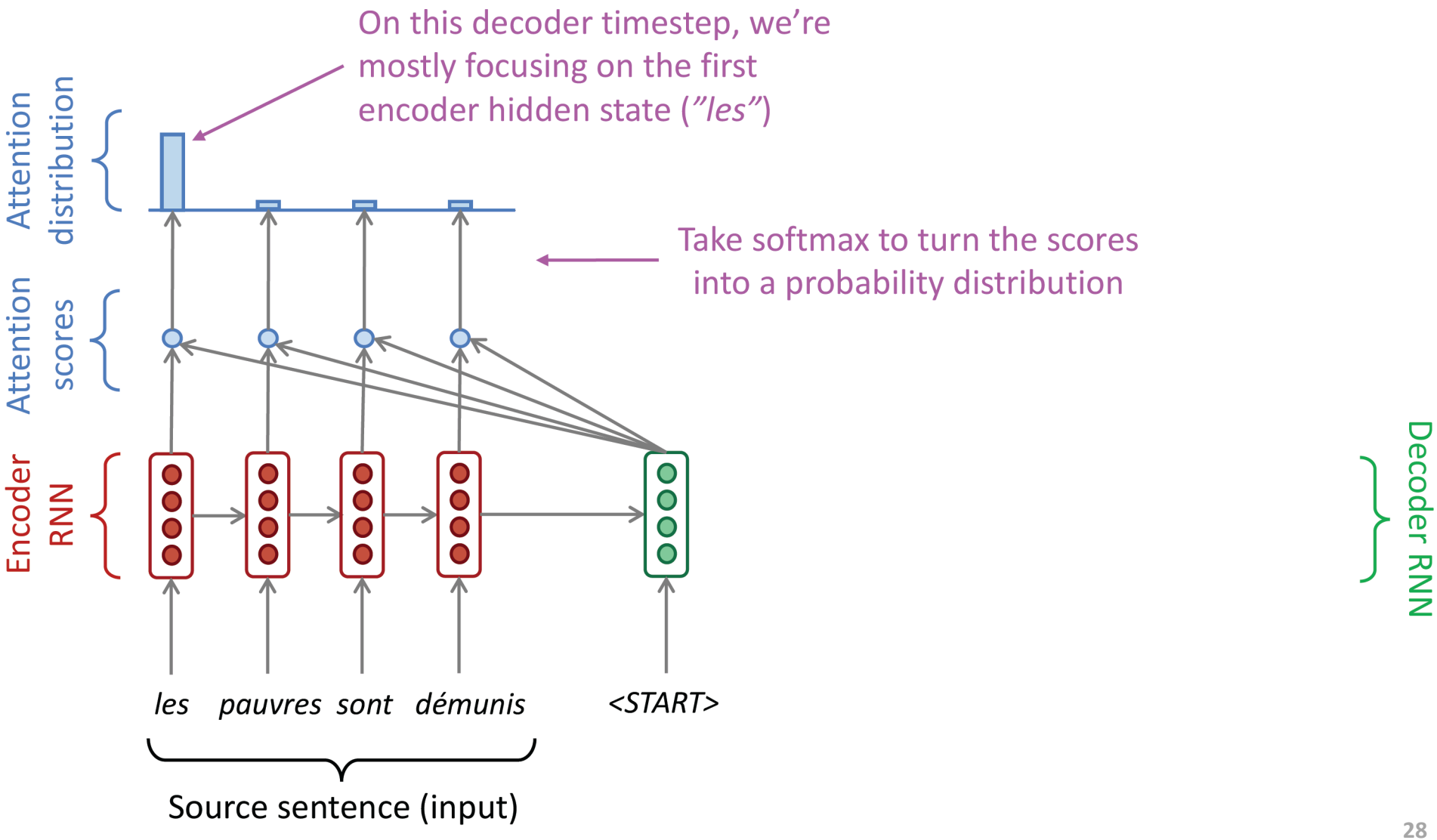
Dot-Product Attention



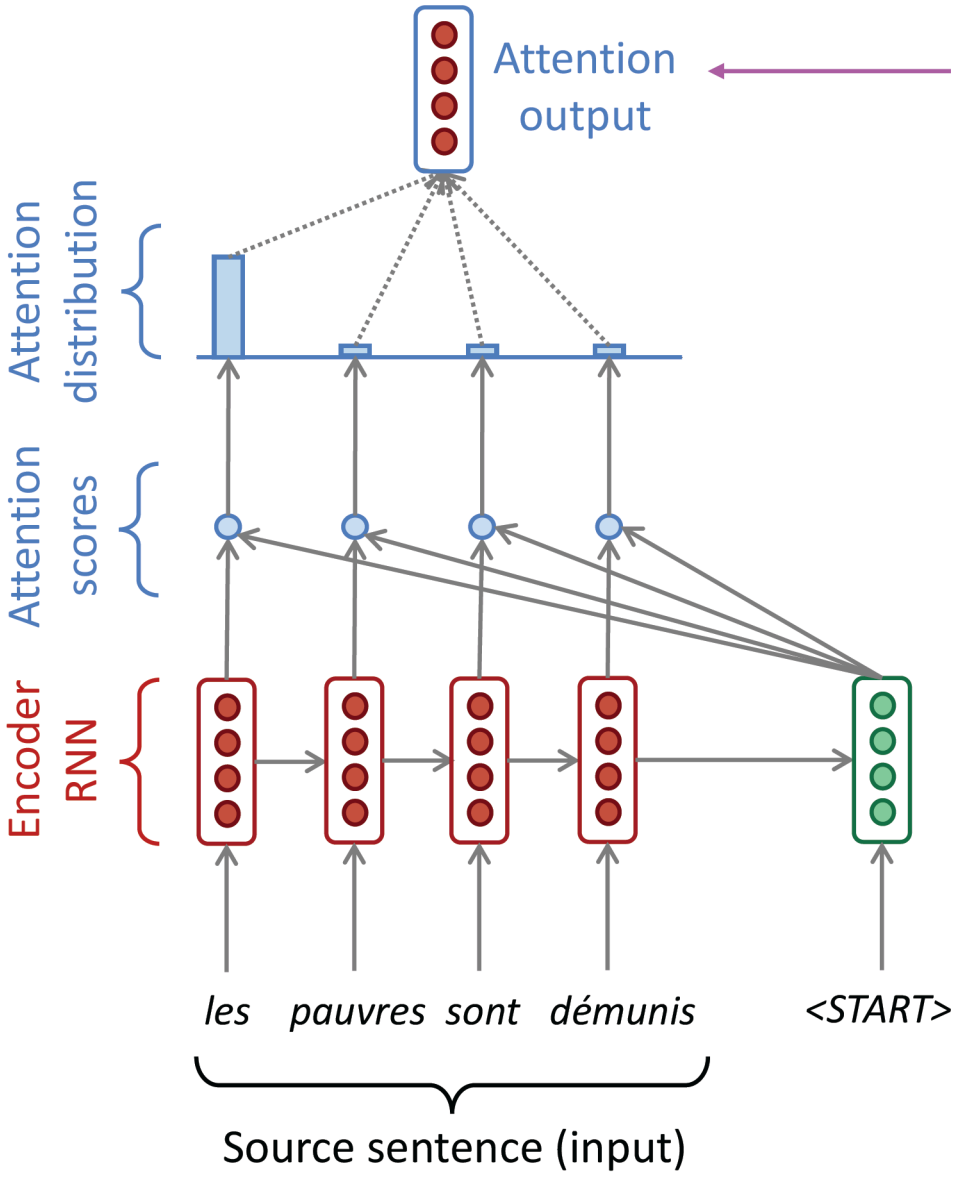
Dot-Product Attention



Dot-Product Attention



Dot-Product Attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the hidden states that received high attention.

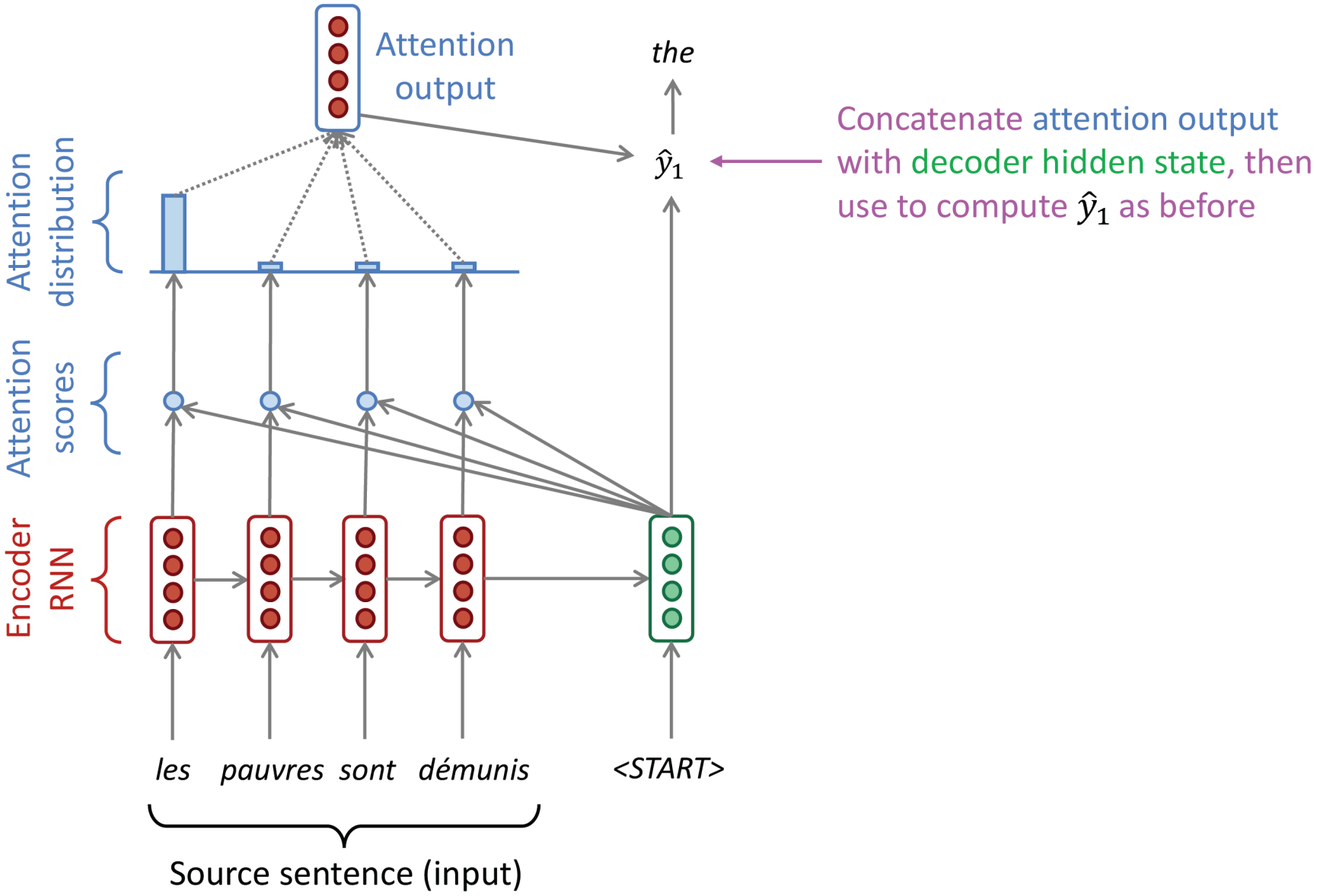
So by this we give information of all source tokens to start token

Also we give each corresponding token information directly

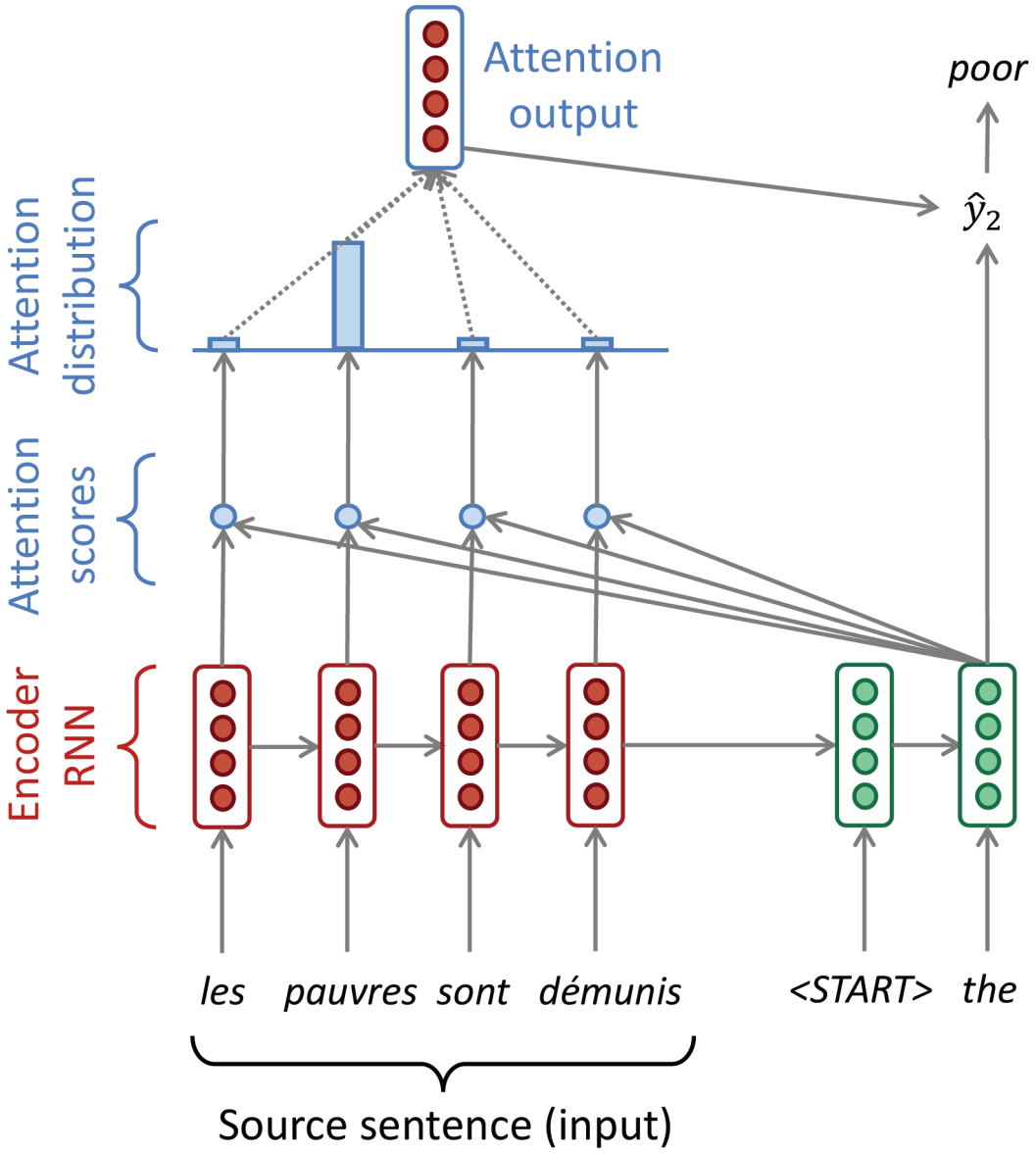
Also we use attention information

Decoder RNN

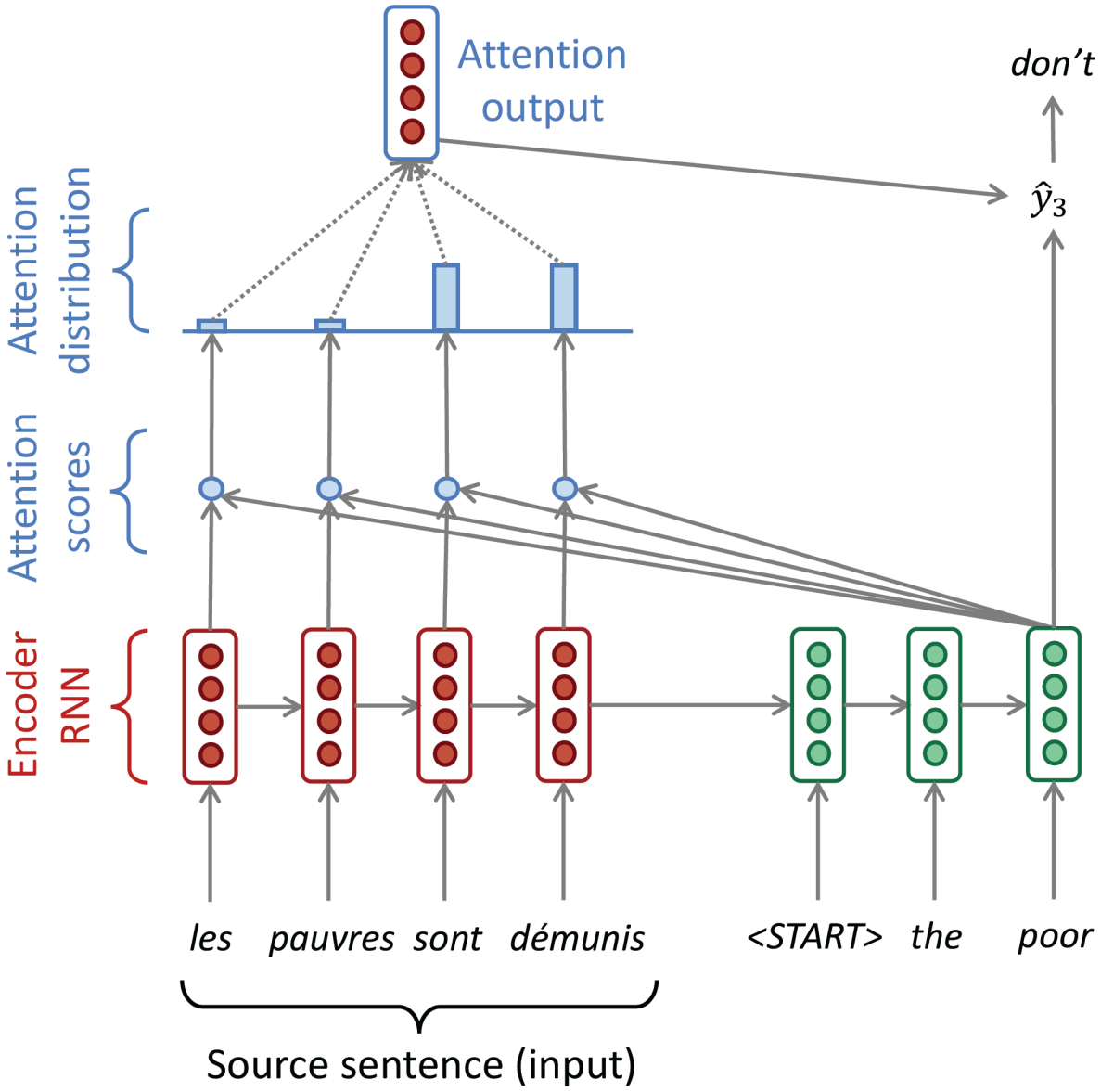
Dot-Product Attention



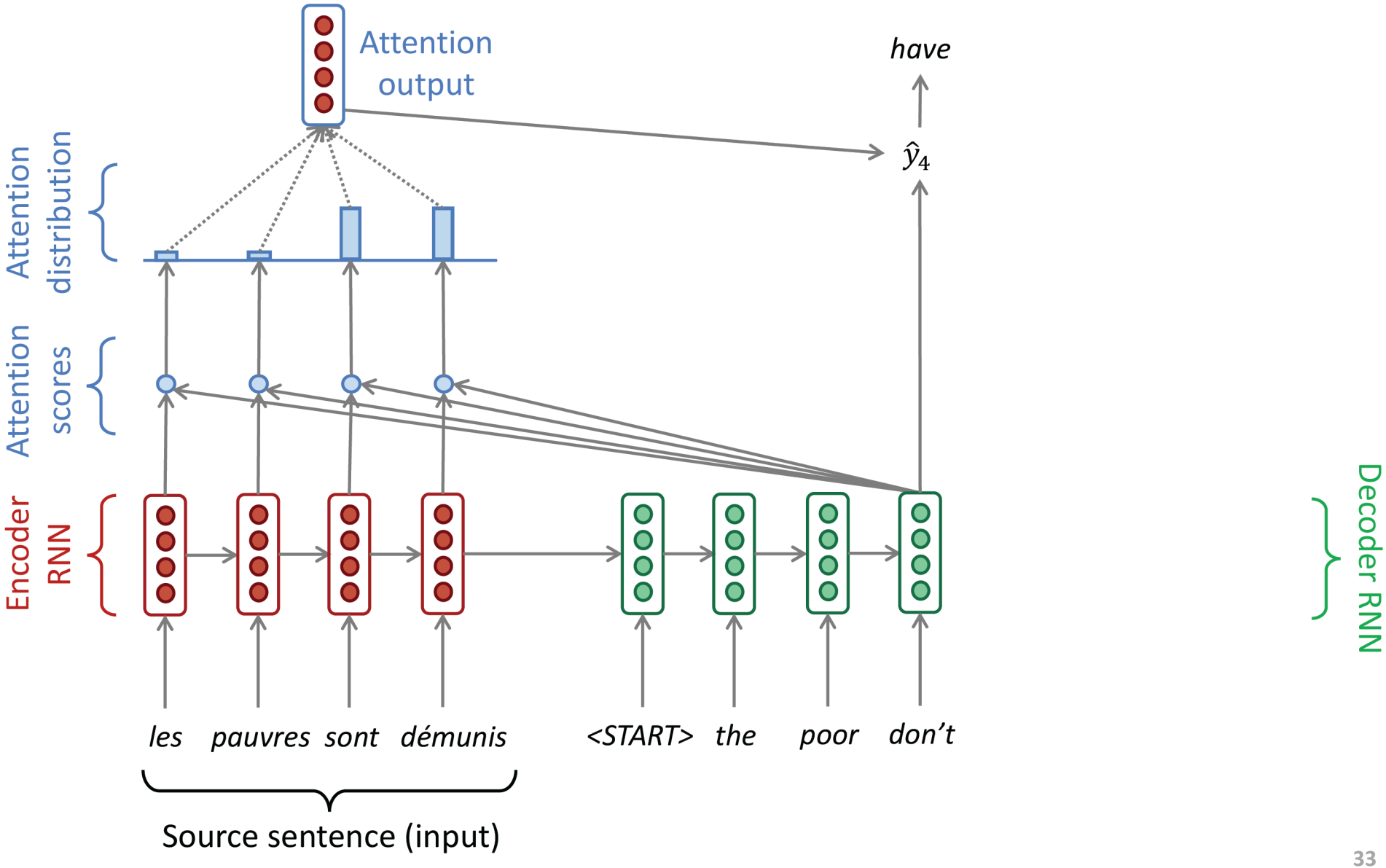
Dot-Product Attention



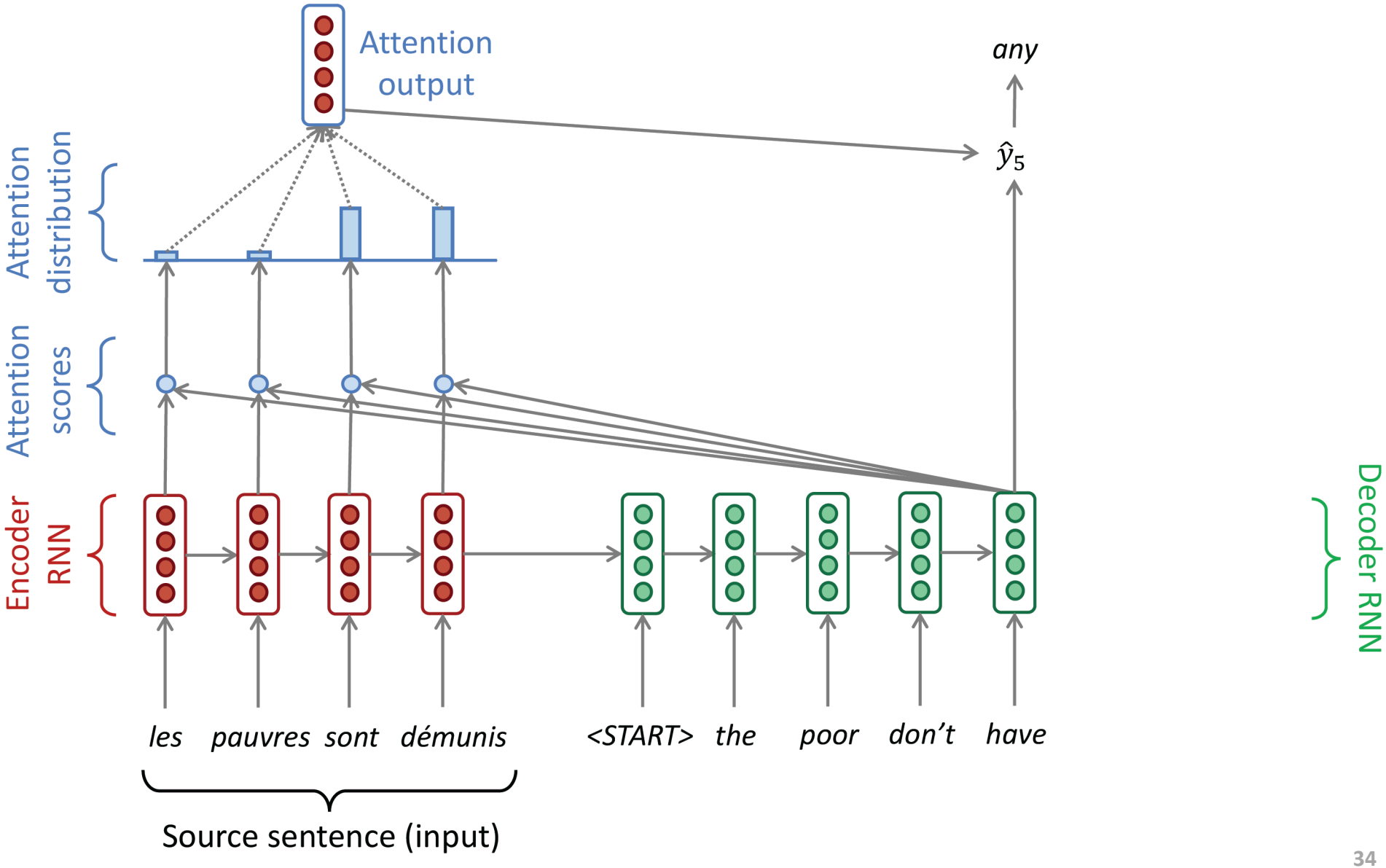
Dot-Product Attention



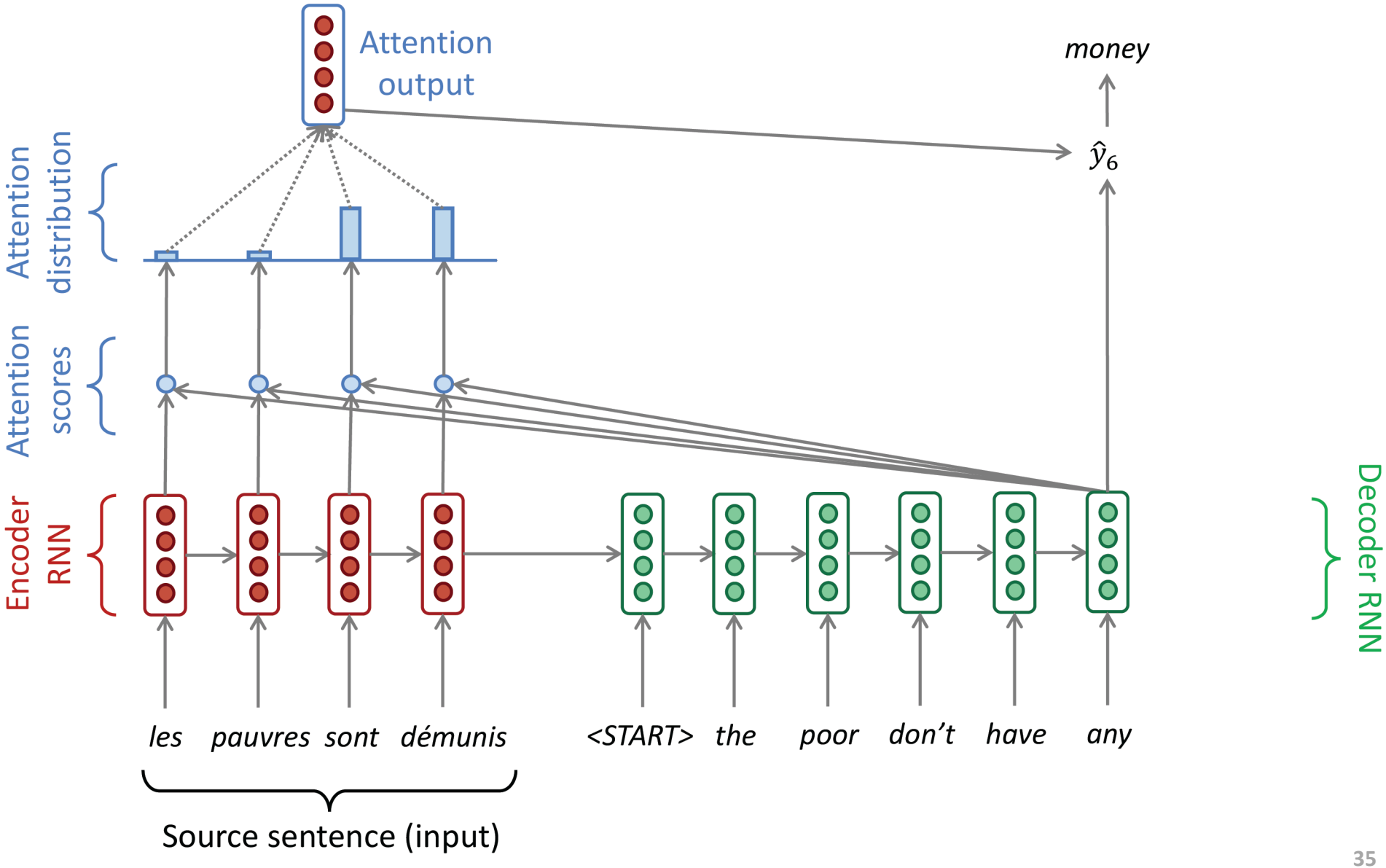
Dot-Product Attention



Dot-Product Attention



Dot-Product Attention



Dot-Product Attention

- Calculate relevance of each encoder state to i-th decoder state:
  - Similarity score between *j*-th encoder and *i*-th decoder hidden states:

$$score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) = \mathbf{h}^d_{i-1} \cdot \mathbf{h}^e_j$$

→ Normalize score and repeat for each encoder state

$$\alpha_{ij} = \text{softmax}(score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) \; \forall j \in e)$$

$$= \frac{\exp(score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)}{\sum_k \exp(score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_k))}$$

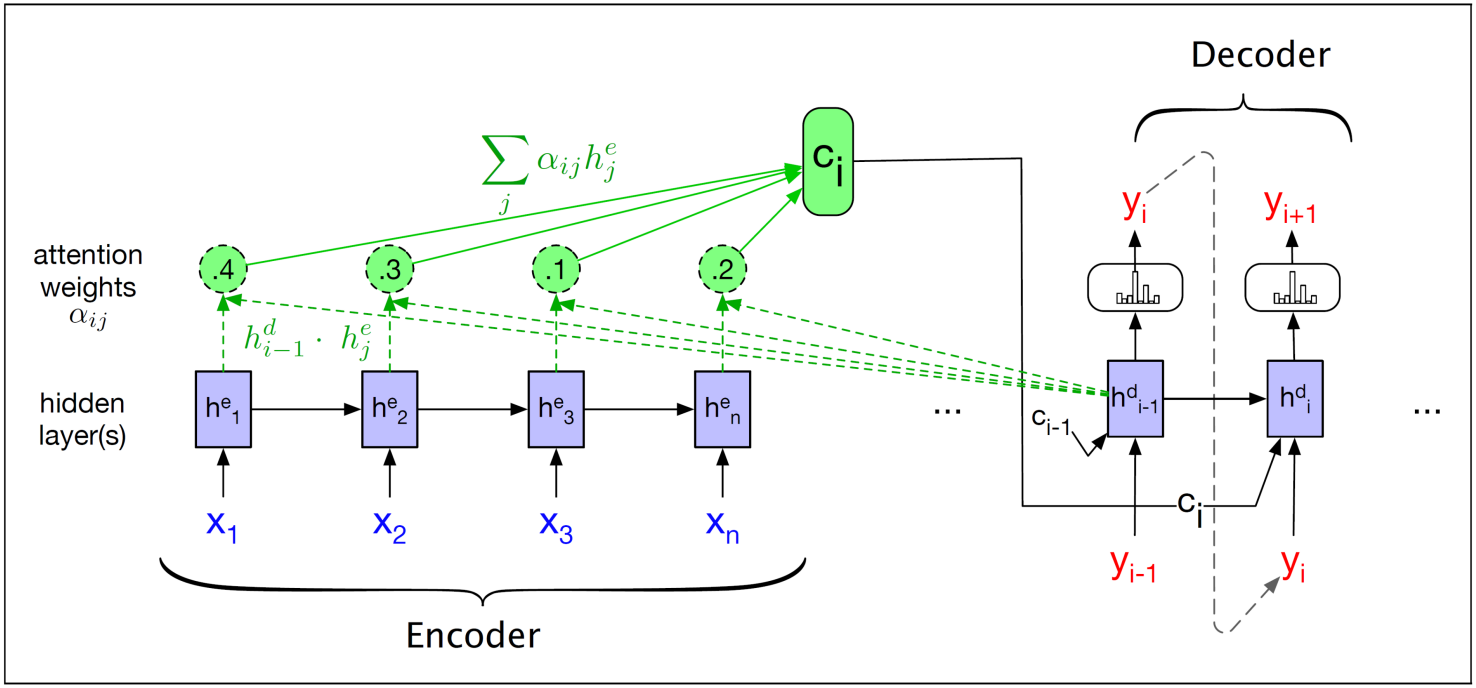
- Possibility to include weights in score function: $score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) = \mathbf{h}^d_{t-1} \mathbf{W}_s \mathbf{h}^e_j$
  - Weights optimized during training
  - → Learn aspects relevant in current application
  - → Allow different dimensions in encoder and decoder

Dot-Product Attention

- Given distribution α, calculate fixed-length context vector for *i*-th decoder hidden state

$$\mathbf{c}_i = \sum_j \alpha_{ij}\, \mathbf{h}_j^e$$

- Encoder-decoder with dynamic context vector:



Dot-Product Attention: Summary

- encoder hidden states $\boldsymbol{h}_1^e, \boldsymbol{h}_2^e, \ldots, \boldsymbol{h}_N^e$ (where each $\boldsymbol{h}_1^e \in \mathbb{R}^h$)
- at time-step t:
  - decoder hidden state: $\boldsymbol{h}_t^d \in \mathbb{R}^h$
  - attention scores: $\boldsymbol{scores}^{(t)} = \left[\boldsymbol{h}_t^d \boldsymbol{h}_1^e, \boldsymbol{h}_t^d \boldsymbol{h}_2^e, \ldots, \boldsymbol{h}_t^d \boldsymbol{h}_N^e\right] \in \mathbb{R}^N$
  - attention distribution: $\boldsymbol{\alpha}^{(t)} = \text{softmax}\left(\boldsymbol{scores}^{(t)}\right) \in \mathbb{R}^N$
  - attention output (fixed-length context vector): weighted sum of encoder hidden states:

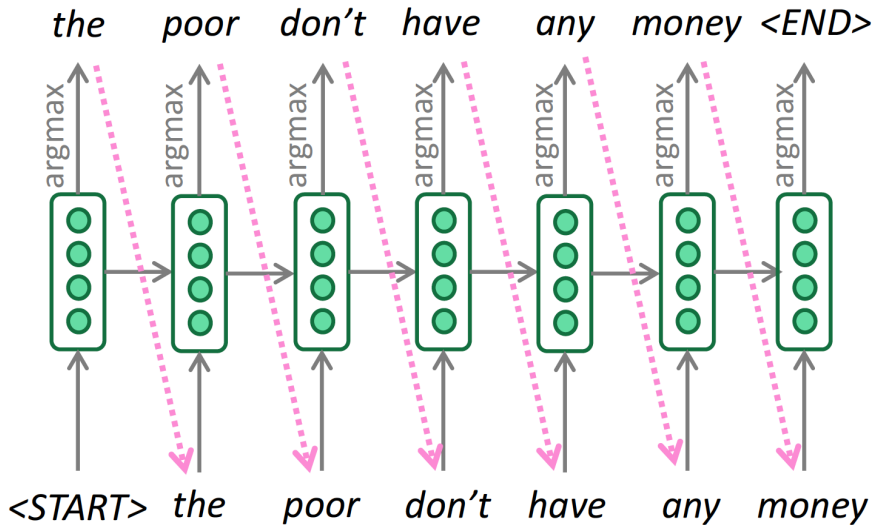
$$\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha_i^{(t)} \boldsymbol{h}_i \quad \in \mathbb{R}^h$$

  - usually: concatenate attention output and decoder hidden state:

$$\left[\boldsymbol{c}_t; \boldsymbol{h}_t^d\right] \in \mathbb{R}^{2h}$$

Decoding strategies: Problems of Greedy Decoding

- decoder uses greedy decoding: take most probably word at each step:



$$\hat{y}_t = \text{argmax}_{w \in V} P(w|x, y_1 \ldots y_{t-1})$$

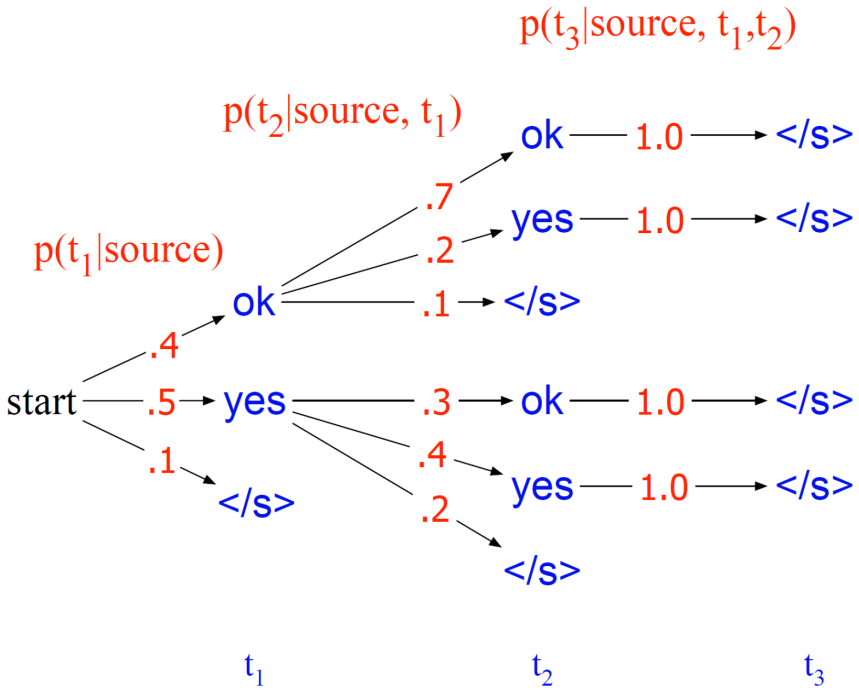
- problem: no way to undo decisions:
  - *les pauvres sont démunis (the poor don't have any money)*
  - → *the* ____
  - → *the poor* ____
  - → *the poor are* ____

So instead of "dont have" after poor if are coming decoder still will decode something about are . It is a problem

Decoding strategies: Problems of Greedy Decoding

- Search tree example:
  - Sequence with highest probability: *"okok</s>"* with *p=.4*.7*1.0*
  - But: greedy decoding chooses *yes* as first word ☹

taking most probable one

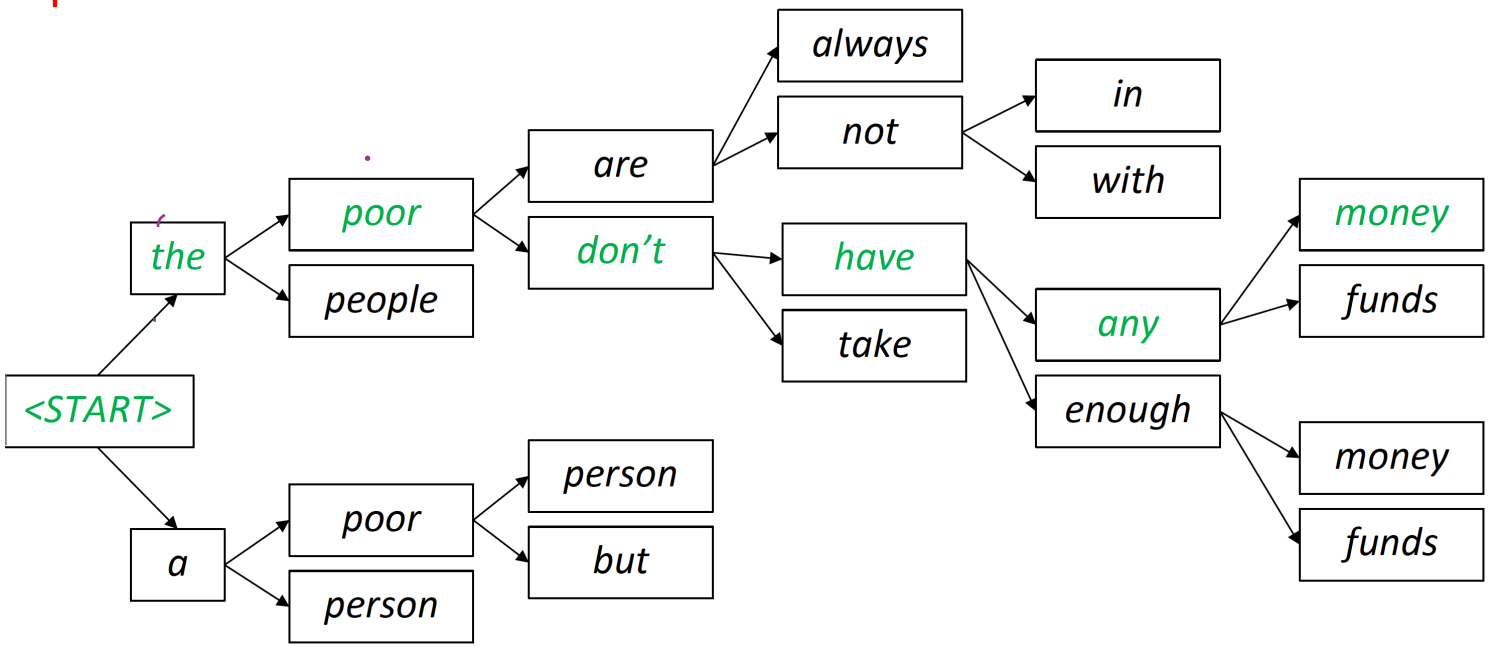


Decoding Strategies: Beam Search Decoding

- → use Beam Search: on each step of decoder, keep track of the k most probable partial translations
  - k: beam size (in practice around 5 to 10)
  - not guaranteed to find optimal solution, but much more efficient!

keep the most probable 2 option

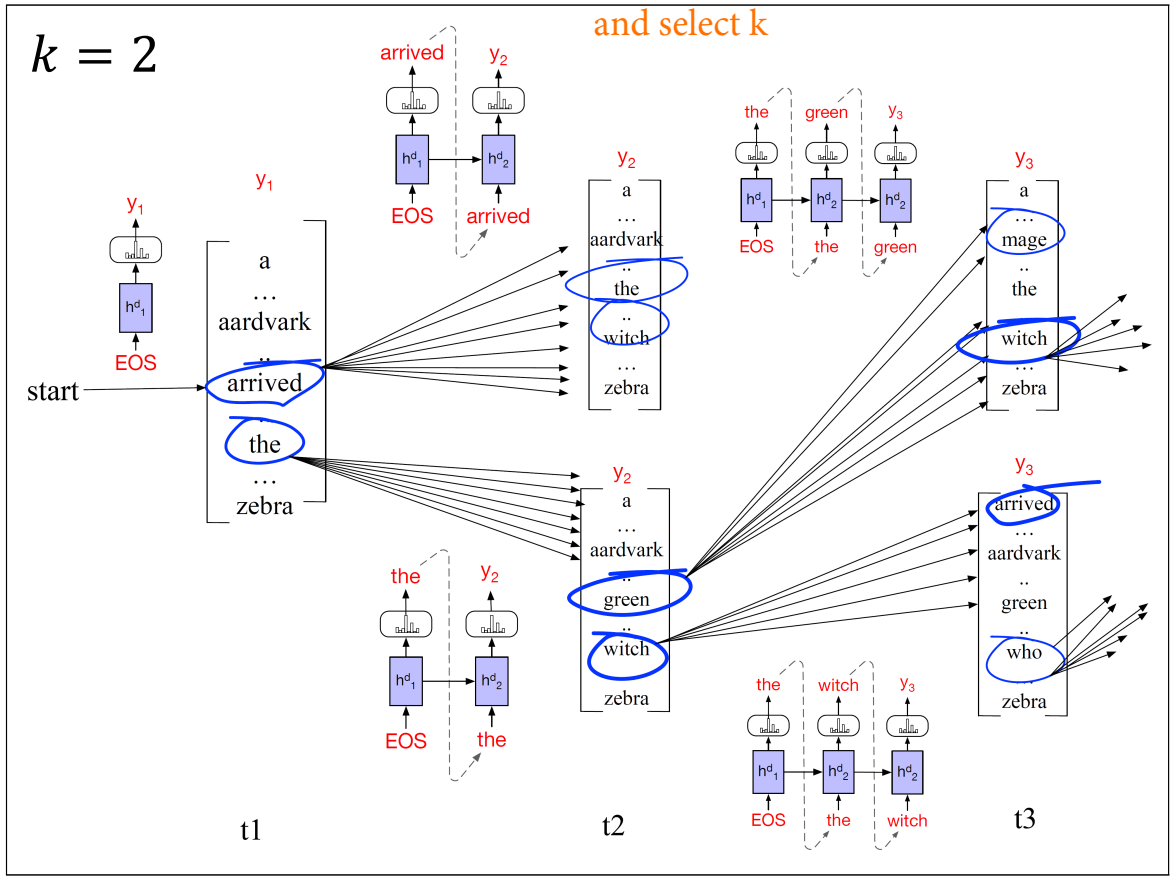
- example: k=2:



Decoding Strategies: Beam Search Decoding

- First decoding step: Softmax over entire vocabulary, select *k*-best options
  → *k* initial words called **hypotheses**
  → parameter *k* called **beam width**
- Subsequent steps: Compute *V* possible extensions by passing trough decoder
  → Rank *k*V* new hypotheses and prune to *k*-best
- Hypothesis complete when </s> produced

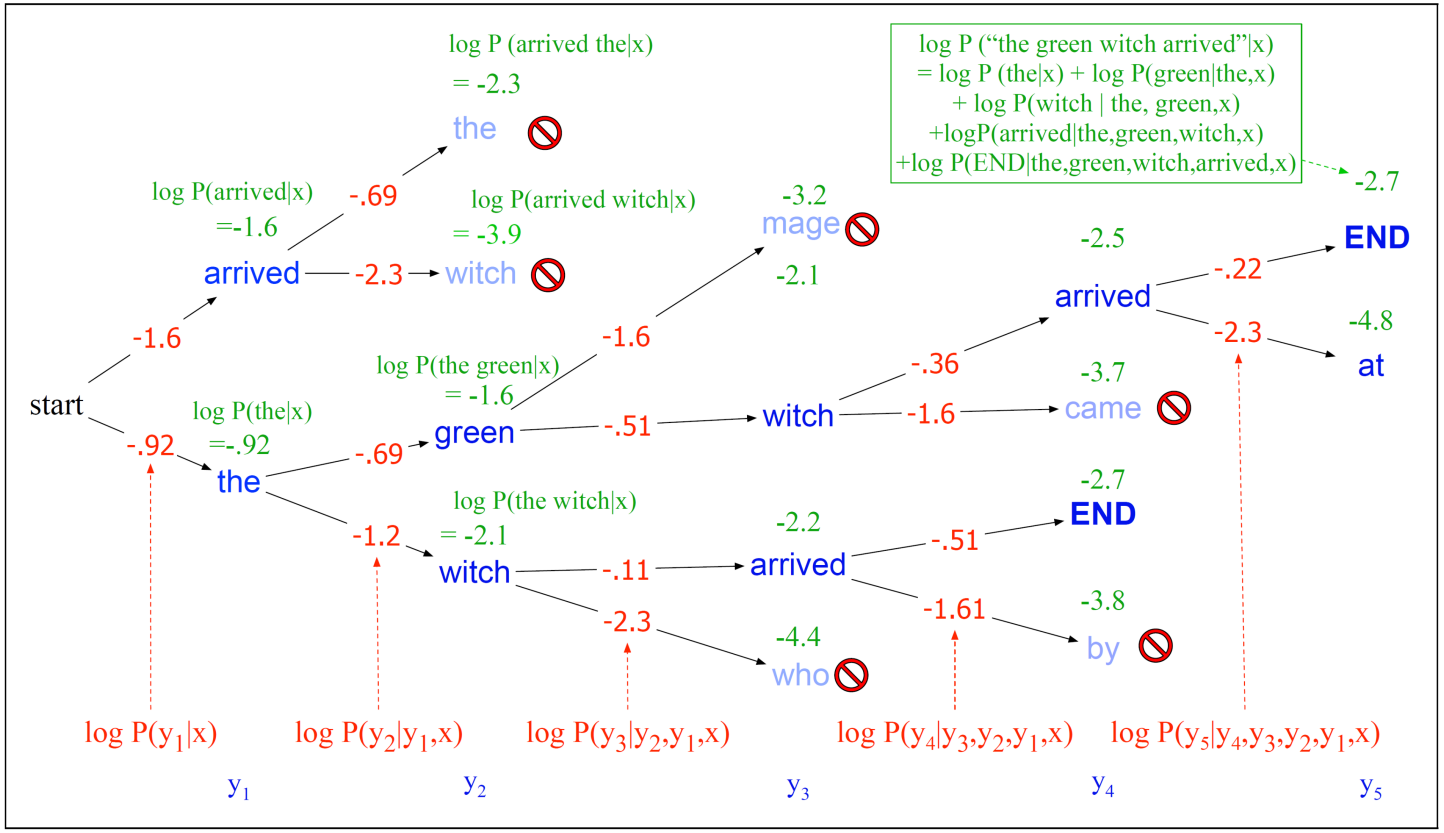
take most probable k, and take compute Vocabulary(V) probs and select k



Decoding strategies: Beam Search

- Ranking hypotheses → Calculate probability of token given existing sequence $P(y_i|x, y_{<i})$

$$
\begin{aligned}
score(y) &= \log P(y|x) \\
&= \log\left(P(y_1|x)P(y_2|y_1,x)P(y_3|y_1,y_2,x)...P(y_t|y_1,...,y_{t-1},x)\right) \\
&= \sum_{i=1}^{t} \log P(y_i|y_1,...,y_{i-1},x)
\end{aligned}
$$



Decoding strategies: Beam Search

- Generate hypothesis until </s> produced
  → Hypotheses with different length

- Problem: Generally lower probability for longer strings
  → length normalization

if the sentences are long we need to multiply it so it going to decrease too much

$$score(y) = -\log P(y|x) = \frac{1}{T}\sum_{i=1}^{t} -\log P(y_i|y_1, \ldots, y_{i-1}, x)$$

Decoding strategies: Beam search algorithm

```
function BEAMDECODE(c, beam_width) returns best paths

  y_0, h_0 ← 0
  path ← ()
  complete_paths ← ()
  state ← (c, y_0, h_0, path)          ;initial state
  frontier ← ⟨state⟩                   ;initial frontier

  while frontier contains incomplete paths and beamwidth > 0
    extended_frontier ← ⟨⟩
    for each state ∈ frontier do
      y ← DECODE(state)
      for each word i ∈ Vocabulary do
        successor ← NEWSTATE(state, i, y_i)
        extended_frontier ← ADDTOBEAM(successor, extended_frontier,
                                      beam_width)

    for each state in extended_frontier do
      if state is complete do
        complete_paths ← APPEND(complete_paths, state)
        extended_frontier ← REMOVE(extended_frontier, state)
        beam_width ← beam_width - 1
    frontier ← extended_frontier

  return completed_paths

function NEWSTATE(state, word, word_prob) returns new state

function ADDTOBEAM(state, frontier, width) returns updated frontier

  if LENGTH(frontier) < width then
    frontier ← INSERT(state, frontier)
  else if SCORE(state) > SCORE(WORSTOF(frontier))
    frontier ← REMOVE(WORSTOF(frontier))
    frontier ← INSERT(state, frontier)
  return frontier
```

Advantages and Disadvantages of NMT

Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- 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

Advantages and Disadvantages of NMT

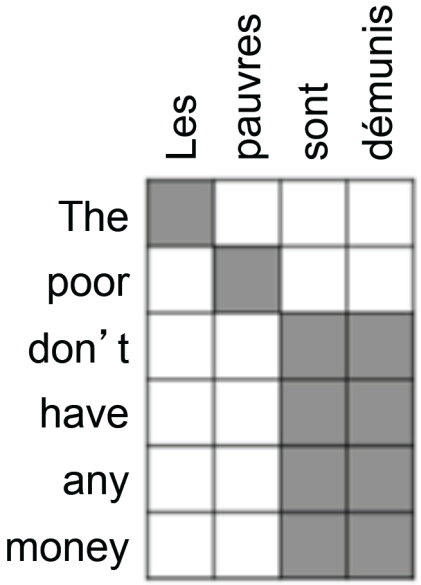
Disadvantages of NMT?

Compared to SMT:

- NMT is less interpretable
  - Hard to debug
- NMT is difficult to control
  - For example, can't easily specify rules or guidelines for translation
  - Safety concerns!

Advantages of Attention

- significantly improves NMT performance
  - very useful to allow decoder to focus on certain parts of the source
- solves the bottleneck problem
  - allows decoder to look directly at source; bypass bottleneck
- helps with vanishing gradient problem
  - Provides shortcut to faraway states
- provides some interpretability
  - by inspecting attention distribution, we can see what the decoder was focusing on
  - we get alignment for free! (we never explicitly trained an alignment system; network just learned alignment by itself)

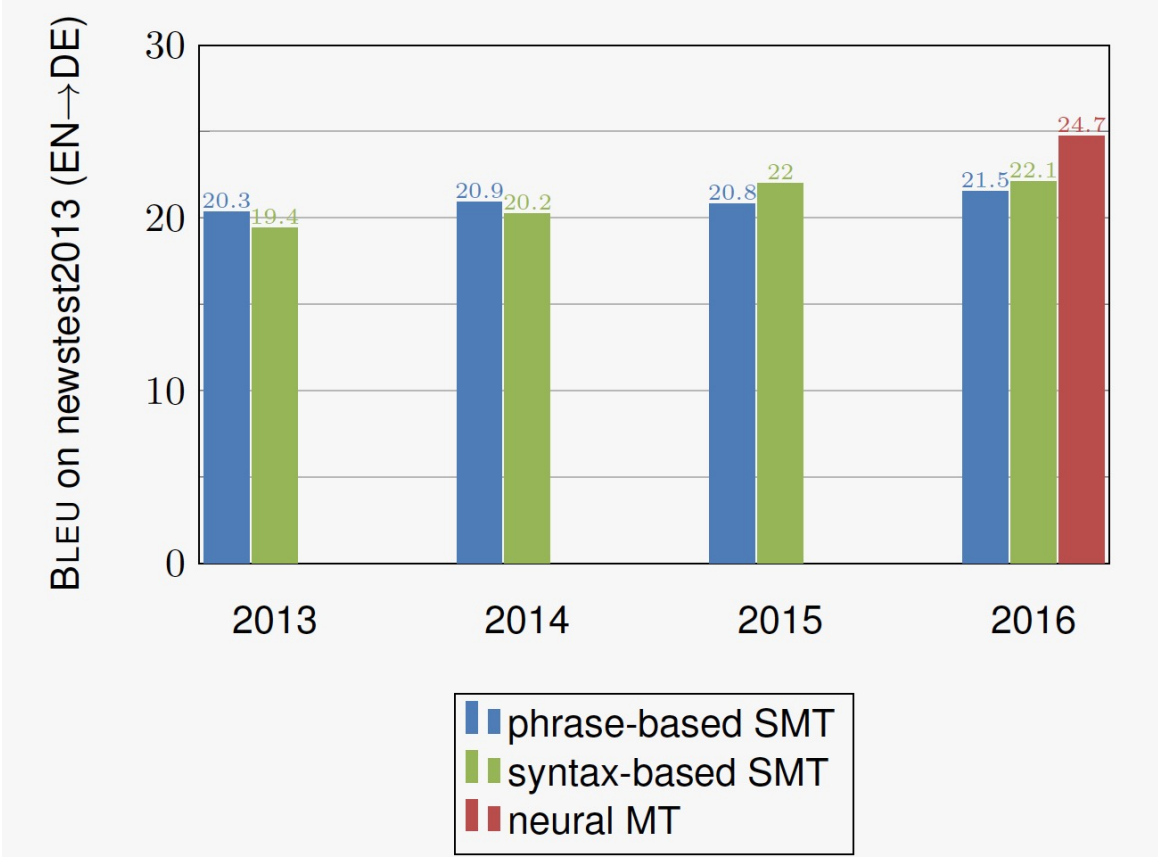


Evaluation of Machine Translation

- BLEU (Bilingual Evaluation Understudy): compare machine translation to one or more human translations & compute similarity score based on:
  - n-gram precision (usually n=3 or 4)
  - & extra penalties for too short machine translations

- disadvantages of BLEU: many valid translations exist → poor BLEU score of an otherwise good translation that just differs in n-gram overlap from the available human translations in corpus

Progress in Machine Translation

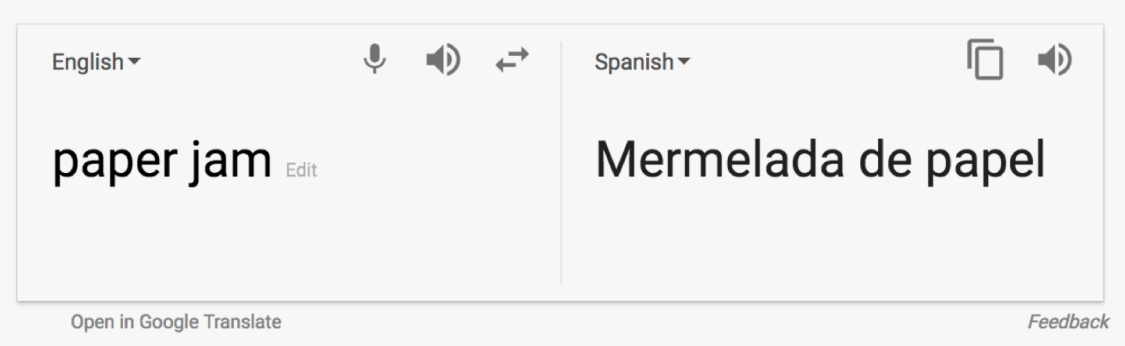


[4]

- 2014: first seq 2 seq paper
- 2016: Google translate switches from SMT to NMT

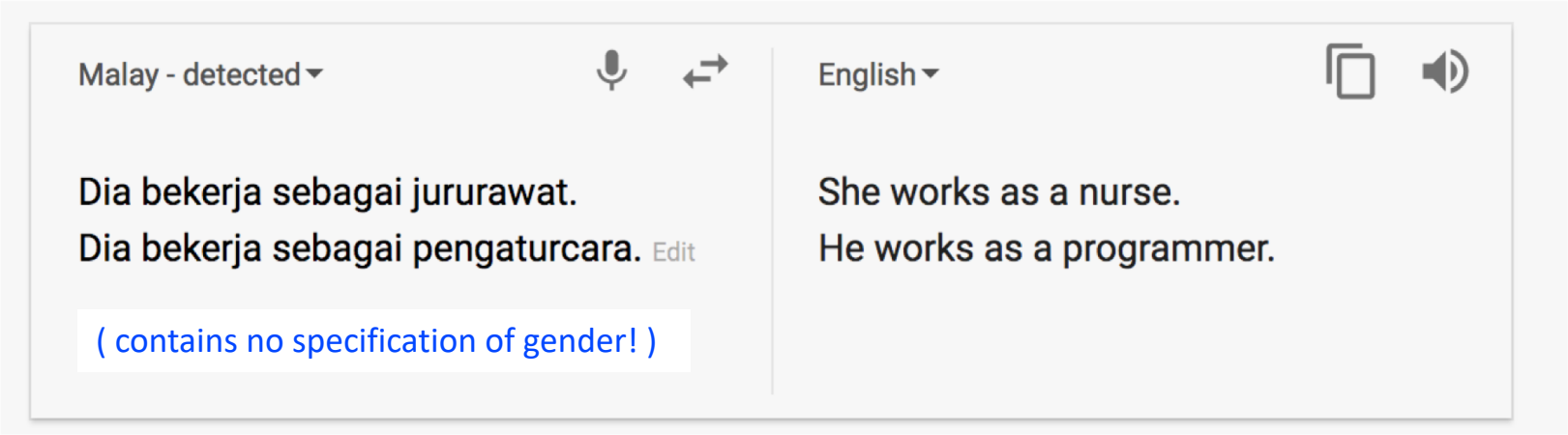
Prevailing Problems

- out-of-vocabulary words
- domain mismatch between train and test data
- maintaining context over longer text
- low-resource language pairs
- using common sense still hard:



Prevailing Problems

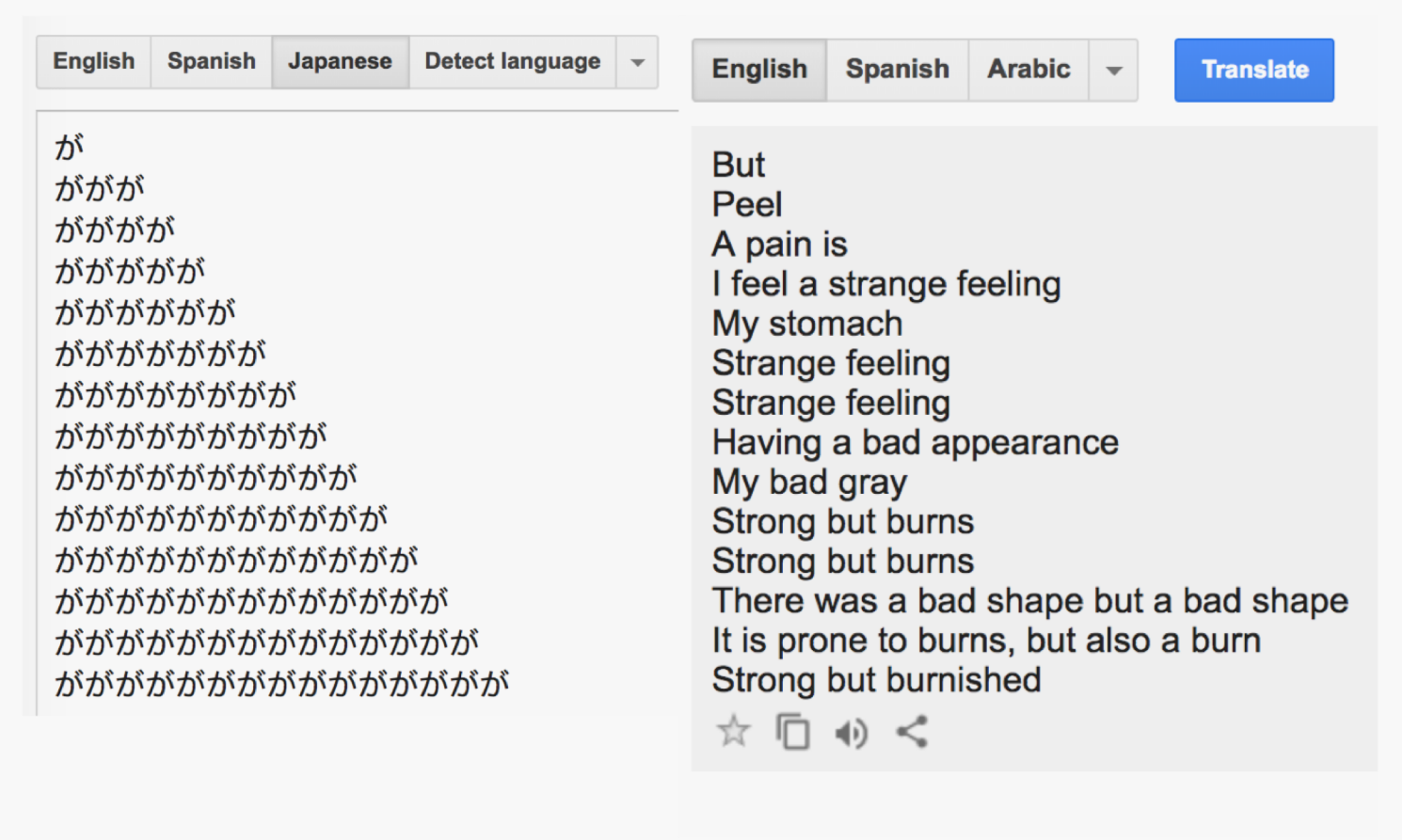
- NMT picks up cultural bias in training data:



[5]

Prevailing Problems

- Uninterpretable systems do strange things [1] :

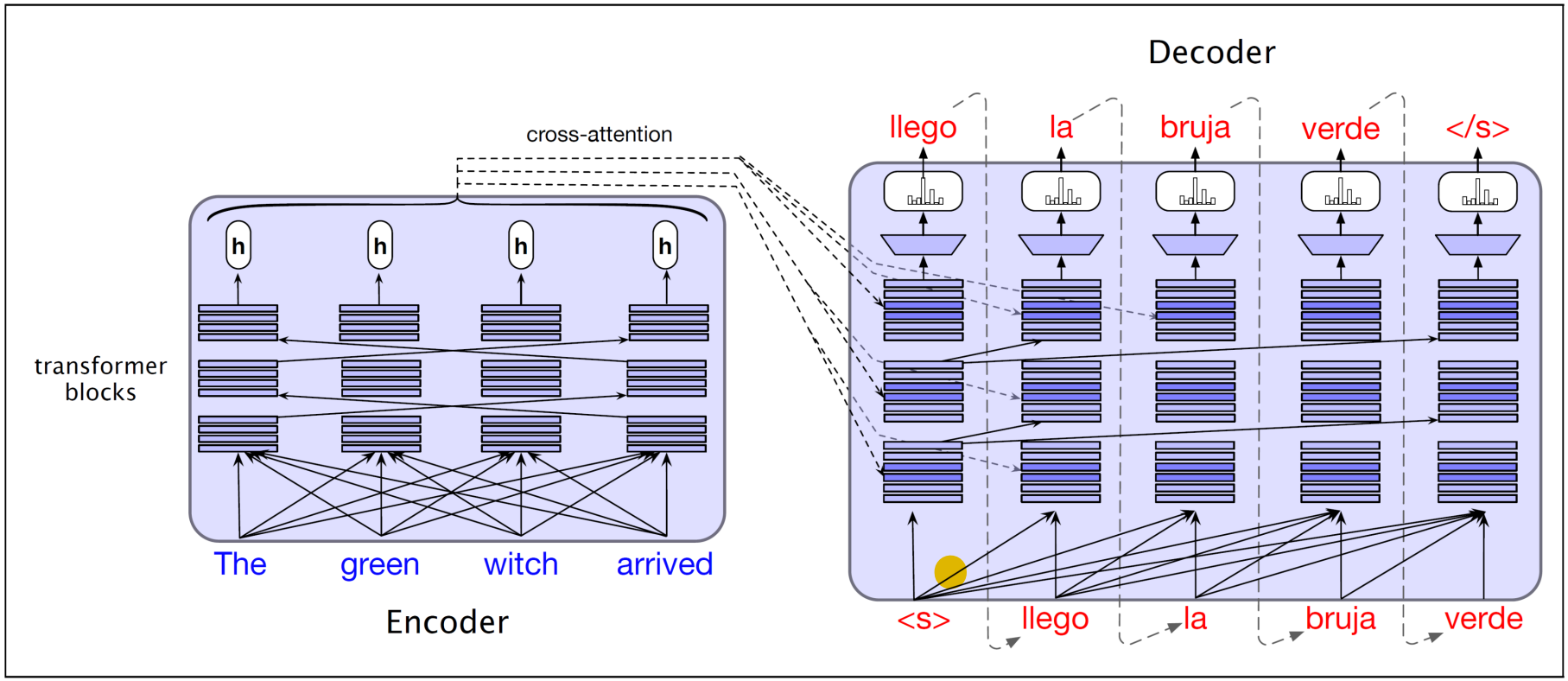


[6]

[1] : philosophical question: is this really a problem? would humans perform systematically better?

Encoder-Decoder with Transformers

- Replace RNN (sequential input) with transformer blocks (parallel input)
- Dot product attention → cross-attention layer



Cross-Attention

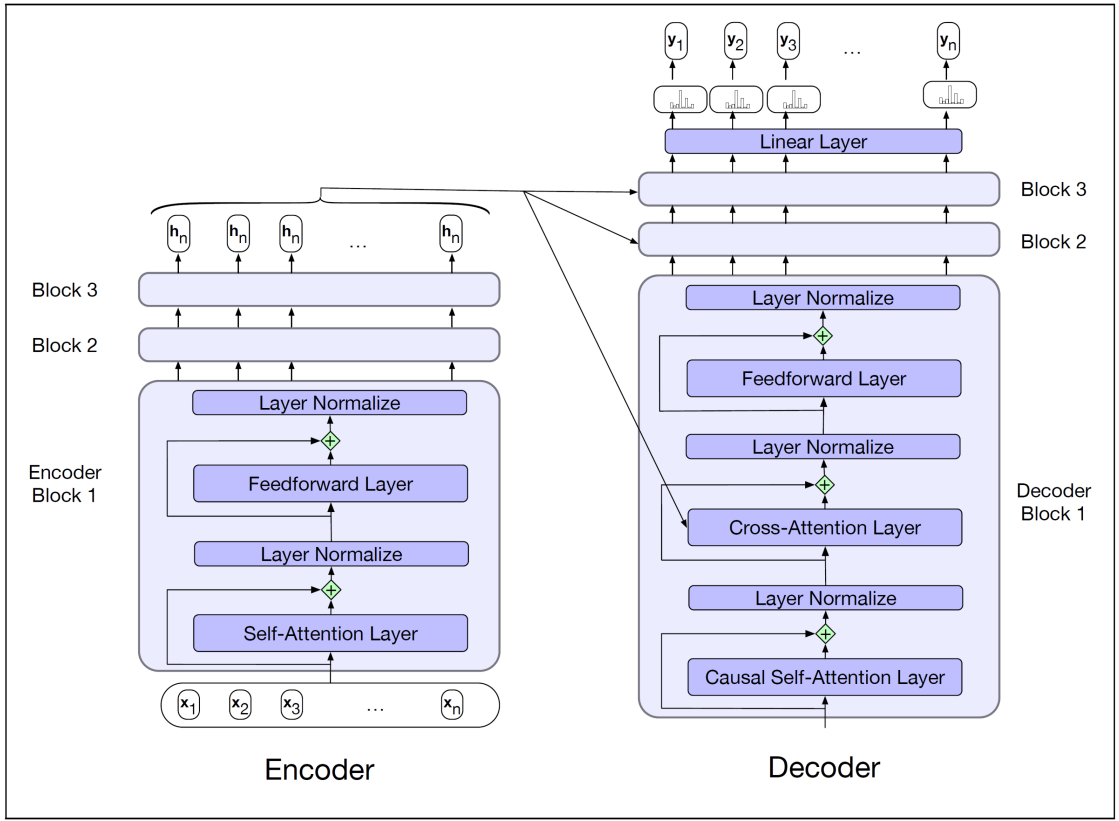
- Recap self-attention: queries, keys and values from previous decoder layer
- Cross-attention: keys and values from encoder output

$$\mathbf{Q} = \mathbf{W}^{\mathbf{Q}}\mathbf{H}^{dec[i-1]};\ \mathbf{K} = \mathbf{W}^{\mathbf{K}}\mathbf{H}^{enc};\ \mathbf{V} = \mathbf{W}^{\mathbf{V}}\mathbf{H}^{enc}$$

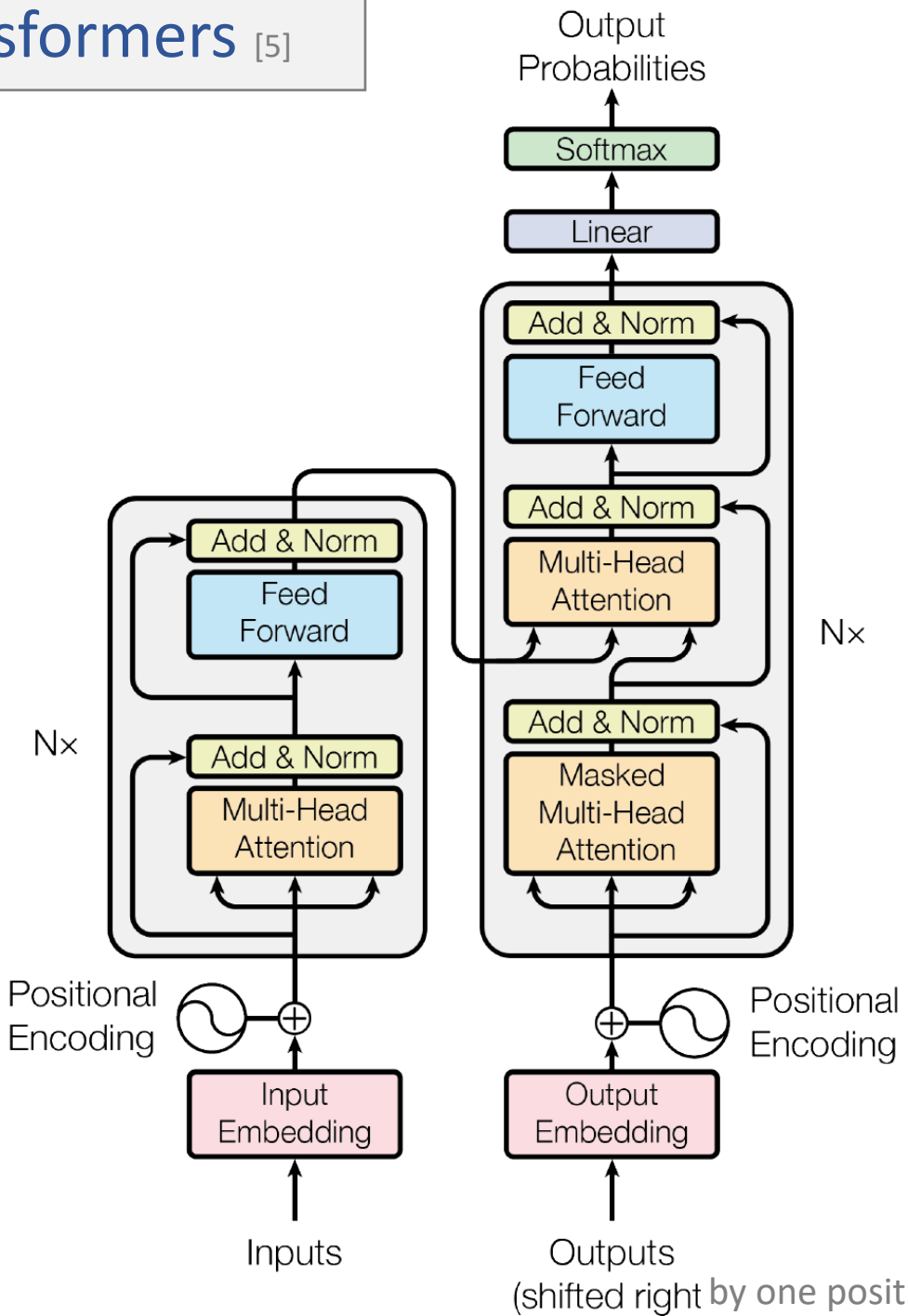
$$CrossAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V}$$

- → decoder attends to all source words

we are not going to replace
self attention
it is new attention



Encoder-Decoder with Transformers [5]

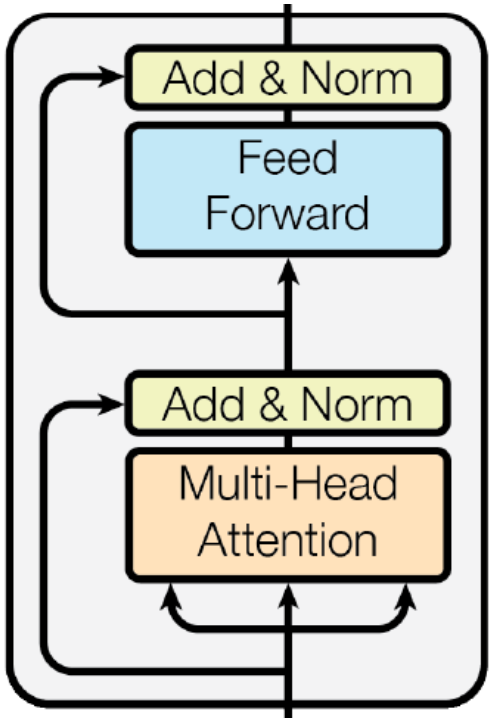


Complete Block of Encoder [5]

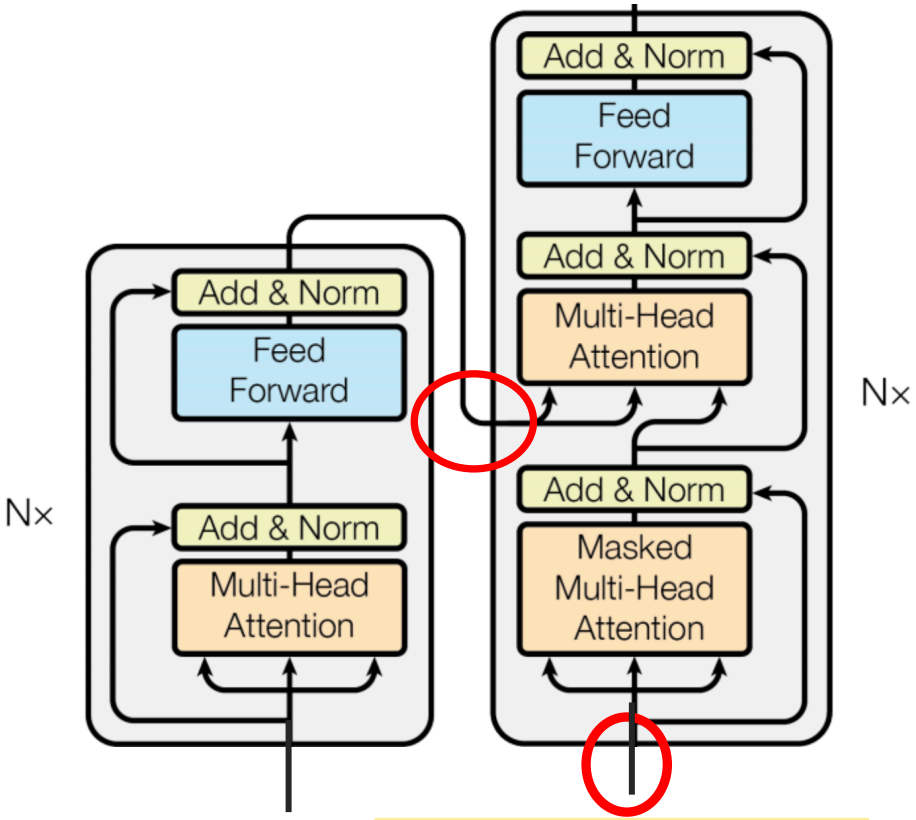
- each block: two “sublayers”:
  - unmasked multihead self attention
  - two-layer fully connected feed forward (C)NN with ReLu:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- each sublayer:
  - residual (short-circuit) connection and Layer Norm:
  - Layer Norm(x + Sublayer(x))
  - Layer Norm changes input to have mean 0 and variance 1

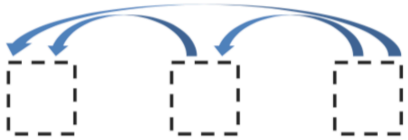


Decoder [5]



encoder decoder attention : queries from previous decoder layer; keys & values from encoder output

decoder self attention on previously generated outputs; masked to prevent information flow from future to past (ensure auto-regressive property)



Tokenization

- Fixed vocabulary built by BPE or wordpiece algorithms
  → corpus with tokens from both languages (enables copying)

- Wordpiece algorithm:
  - Input: Training corpus and desired vocabulary size *V*
  - 1. Init lexicon with all characters
  - 2. Repeat until *V* wordpieces in lexicon
    - Train n-gram LM on training corpus using current lexicon
    - New wordpieces by concatenating two from current lexicon
      → add most probable piece according to language model to lexicon

words: Jet makers feud over seat width with big orders at stake
wordpieces: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

Backtranslation to increase parallel data

- Problem: for some language pairs only limited amount of parallel data but large monolingual corpus available in target language (e.g. English)
  → translate to source language
  → add synthetic data to parallel corpus

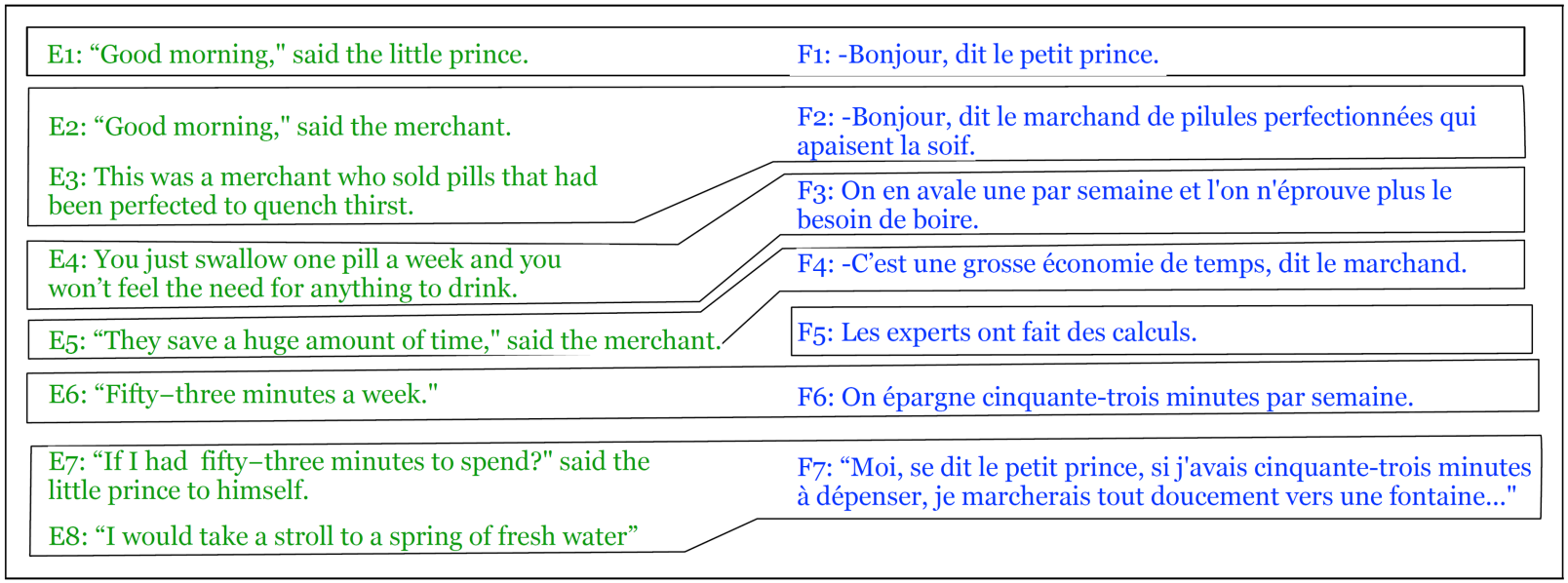
- Backtranslation parameters:
  - Generation of backtranslation
  - Decoding strategy: greedy inference, sampling, Monte Carlo search, ..
  - Ratio of back-translated to natural data

- Monte Carlo search: useful for underfitted decoders
  - Previous approaches: take most probable word
  - Here: sample words according to probabilities

  Softmax probability distribution: {*the*: 0.6; *green*: 0.2; *a*: 0.1; *witch*: 0.1}
  → weighted die with 4 sides weighted 0.6, 0.2, 0.1, 0.1
  → small amount of unprobable words in data

Sentence alignment

- Parallel corpora for training
  → align sentences from both languages
- Score function to measure likelihood of two spans being translations
  e.g. cosine similarity between multilingual embeddings
- Alignment algorithm to find alignment based on scores



Evaluation of Machine Translation

- Evaluation along two dimensions:
  - **Adequacy / Faithfulness**: Preservation of exact meaning
  - **Fluency**: quality of target language text

anlam olarak ayni mi

gramatik olarak target language dogru mu

- Human evaluators
  - Often online crowdworkers
  - Multilingual: rate criterion on defined scale
  - Monolingual: Compare to reference translation or rank pairs of candidate translations
  - Disagreement between raters → normalize and exclude outliers

Automatic Evaluation Metrics: Character overlap

- Compared to humans: less expensive but less accurate
- Assumption: High overlap between MT and human gold translation desired
- **Character F-score (chrF)**: rank candidate by character n-gram overlaps

chrP percentage of character 1-grams, 2-grams, ..., k-grams in the hypothesis that occur in the reference, averaged.

chrR percentage of character 1-grams, 2-grams,..., k-grams in the reference that occur in the hypothesis, averaged.

→ combine precision and recall in weighted F-score
($\beta = 2$ weights recall twice as much as precision)

$$\text{chrF}\beta = (1+\beta^2)\frac{\text{chrP}\cdot\text{chrR}}{\beta^2\cdot\text{chrP}+\text{chrR}}$$

chrF example

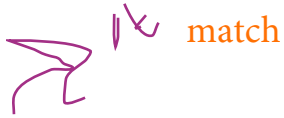
- chrF2,2 → 2-grams and $\beta = 2$

REF: witness for the past,
HYP1: witness of the past, chrF2,2 = .86
HYP2: past witness chrF2,2 = .62

- 1. Remove whitespaces

REF: witnessforthepast, (18 unigrams, 17 bigrams)
HYP1: witnessofthepast, (17 unigrams, 16 bigrams)

- 2. Count matching unigrams and bigrams



unigrams that match: w i t n e s s f o t h e p a s t , (17 unigrams)
bigrams that match: wi it tn ne es ss th he ep pa as st t, (13 bigrams)

- 3. Calculate precision and recall

unigram P:	17/17 = 1	unigram R:	17/18 = .944
bigram P:	13/16 = .813	bigram R:	13/17 = .765

- 4. Combine to F-Score

$$\text{chrP} = (17/17 + 13/16)/2 = .906$$
$$\text{chrR} = (17/18 + 13/17)/2 = .855$$
$$\text{chrF2,2} = 5\frac{\text{chrP} * \text{chrR}}{4\text{chrP} + \text{chrR}} = .86$$

Automatic Evaluation Metrics: Practical Considerations

chrachter F score

- chrF used to compare performances of two models
  → statistical significance test

- More common metric: **BiLingual Evaluation Understudy (BLEU)**
  - Word instead of character n-gram overlap
    → sensitive to tokenization
  - Purely precision-based

same with chrF but word level
just precision looked

- N-gram overlap limitations:
  - No synonym awareness
  - Robust to position changes → bad word order evaluation
  - No coverage of cross-sentence properties
  - Reference summary needed

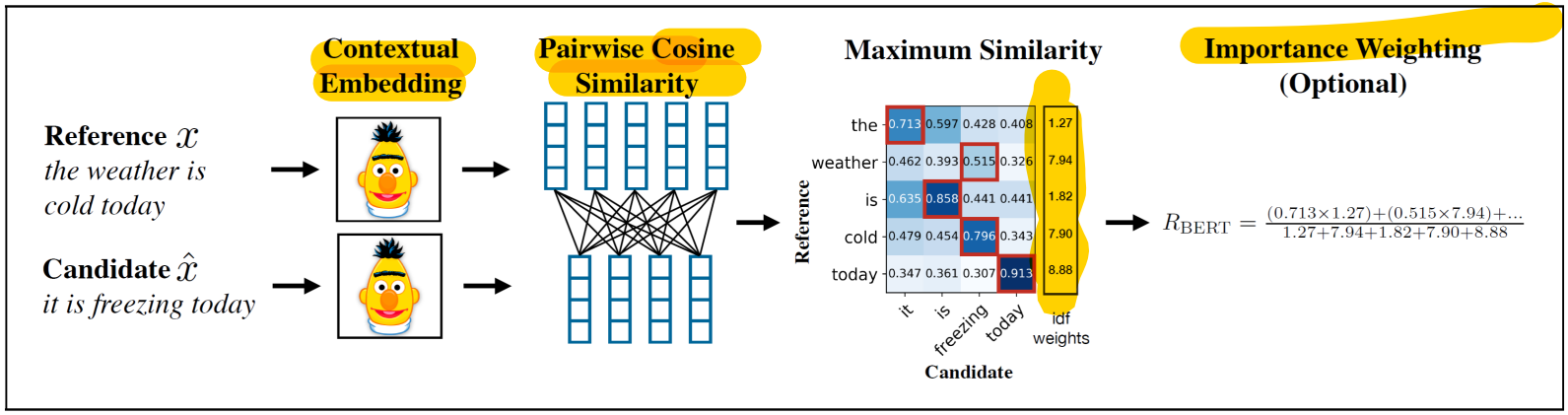
supervised

Automatic Evaluation Metrics: Embedding Based

- Embeddings to include knowledge about synonyms and paraphrases
- Given dataset with (reference translation, candidate translation, human ranking) triplets

  → train predictor by passing reference and candidate through e.g. BERT

- BERTScore: no supervised training data → token-wise cosine similarity of embedding representations



- Token in x matched to $\hat{x}$ → recall
- Tokens in $\hat{x}$ matched to x → precision

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j$$

$$P_{\text{BERT}} = \frac{1}{|\tilde{x}|}\sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j$$

Bias and Ethical Issues

- Translation Hungarian (gender neutral pronoun ő) to English

Hungarian (gender neutral) source	English MT output
ő egy ápoló	she is a nurse
ő egy tudós	he is a scientist
ő egy mérnök	he is an engineer
ő egy pék	he is a baker
ő egy tanár	she is a teacher
ő egy vesküvőszervező	she is a wedding organizer
ő egy vezérigazgató	he is a CEO

- Results difficult to control and interpret → Confidence score of prediction

- Focus on high-resource languages to English
  → efforts to increase low-resourced language corpora

Bibliography

(1) Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft, version Jan, 2022); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL, Oct 2022) (this slideset is especially based on chapter 10)

(2) Richard Socher et al: "CS224n: Natural Language Processing with Deep Learning", Lecture Materials (slides and links to background reading) http://web.stanford.edu/class/cs224n/ (URL, May 2018), 2018

(3) https://youtu.be/K-HfpsHPmvw (URL, Aug 2018) (in [2])

(4) Rico Sennrich (University of Edinburgh): Neural Machine Translation: Breaking the Performance Plateau, Talk July 2016; http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf (URL, Aug 2018) (in [2])

(5) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I., 2017. Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008) and arXiv:1706.03762v5

Recommendations for Studying

- minimal approach:

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

  minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

  == standard approach