LING530F: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

Table of Contents

Neural Machine Translation

Machine Translation

- **Phrase-Based Machine Translation (PBMT)**: Many small sub-components that are tuned separately (e.g., Koehn et al. (2003)
- **Neural Machine Translation (NMT)**: builds and trains a single, large neural network that reads a sentence and outputs a translation
- **Encoder-decoder approach (e.g. Sutskever et al. (2014):**
  - An encoder reads and encodes a source sentence into a fixed-length vector.
  - A decoder then outputs a translation from the encoded vector.
- Whole **encoder–decoder system** for a language pair is jointly trained to maximize the probability of a correct translation given a source sentence

Example Setup

Jaguar F-Type, du rallye dans l'air!

Par Jules Humbert | Publié le 16/11/2018 à 11:43

NOUVEAUTÉ - Jaguar rend hommage aux 70 ans de la naissance de ses modèles XK à travers la réalisation de deux F-Type roadster préparées pour le rallye.

Figure: From Le Figaro.

MT as Seq2Seq Learning: English-French

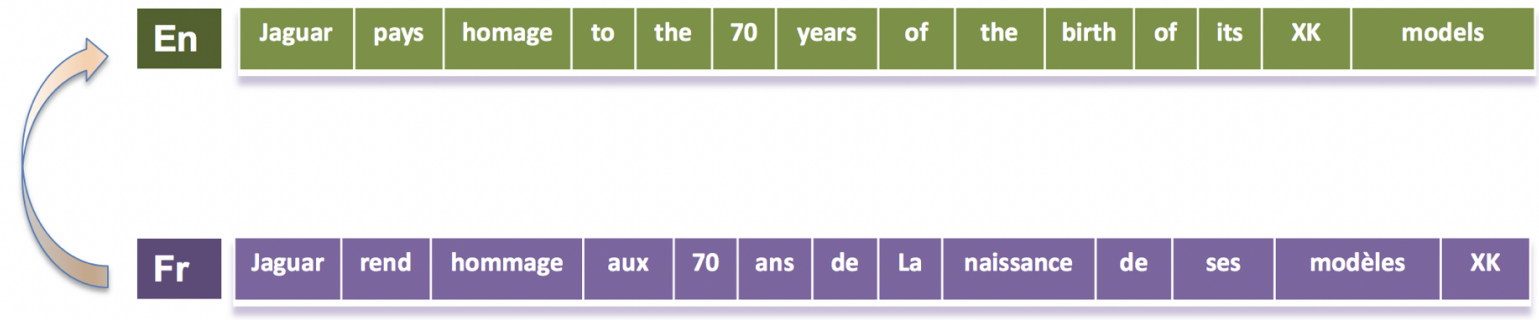
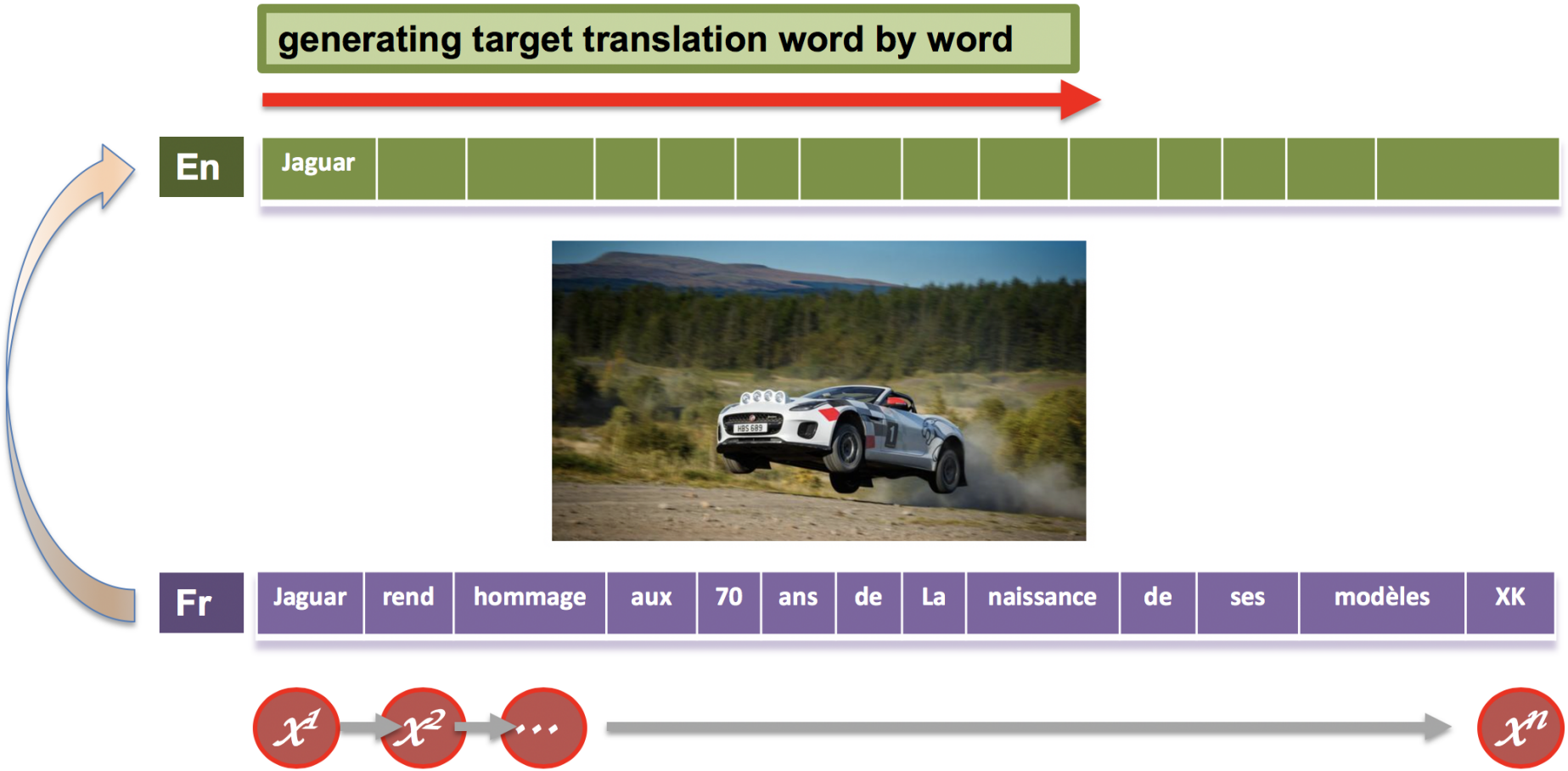
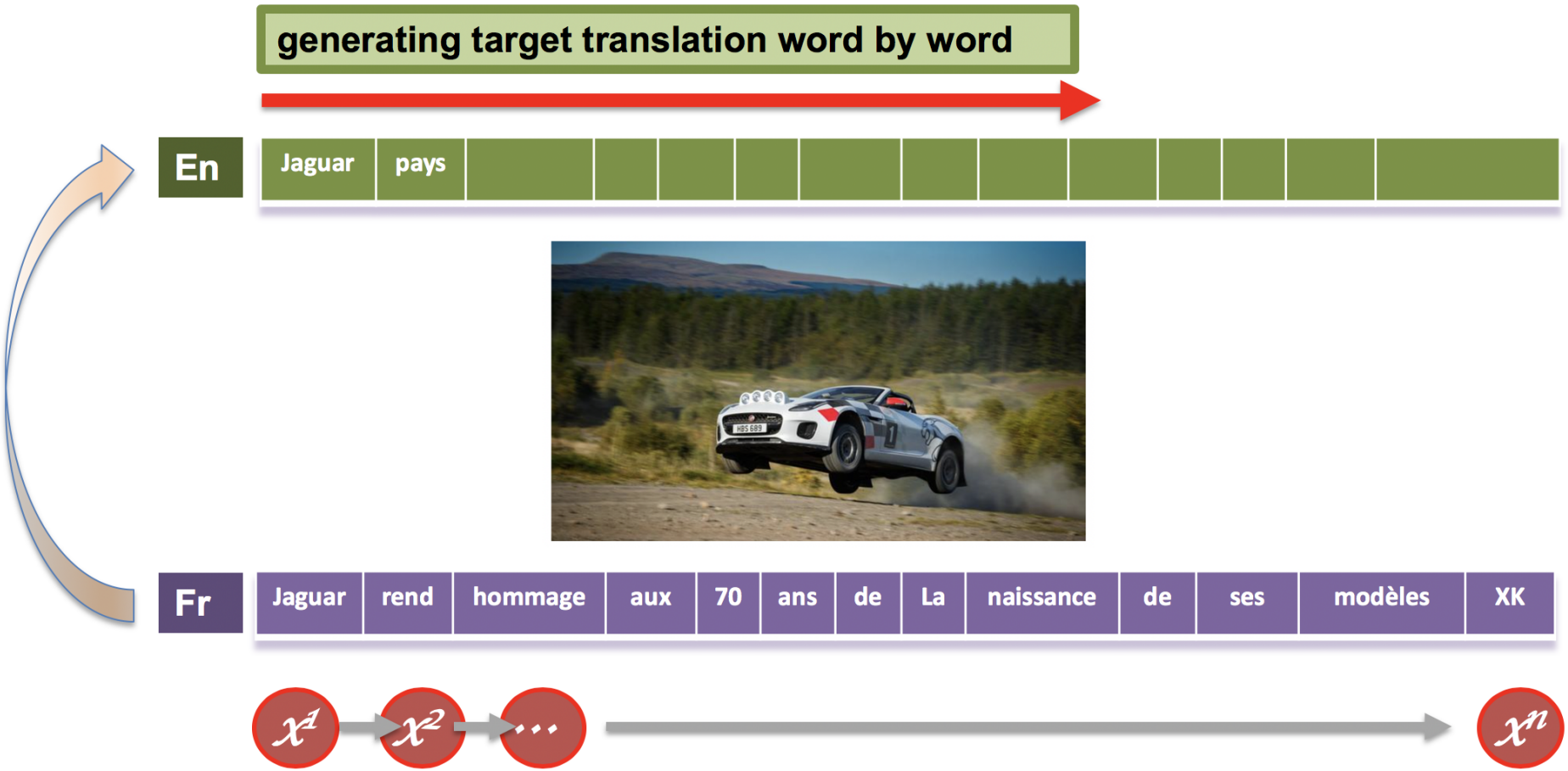


Figure: Translation by Google, November, 20, 2018.

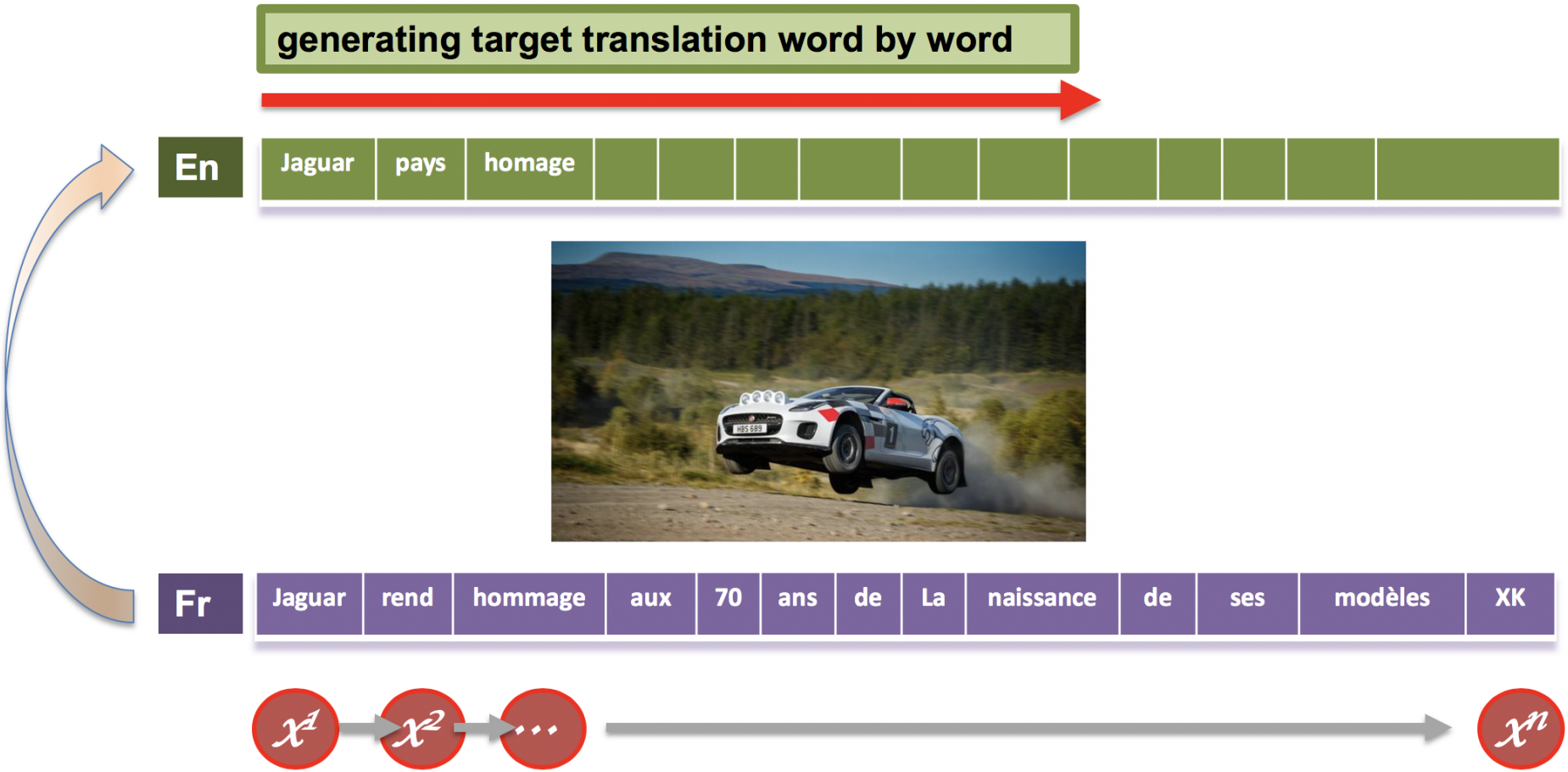
Generating Target (En) One Word at a Time



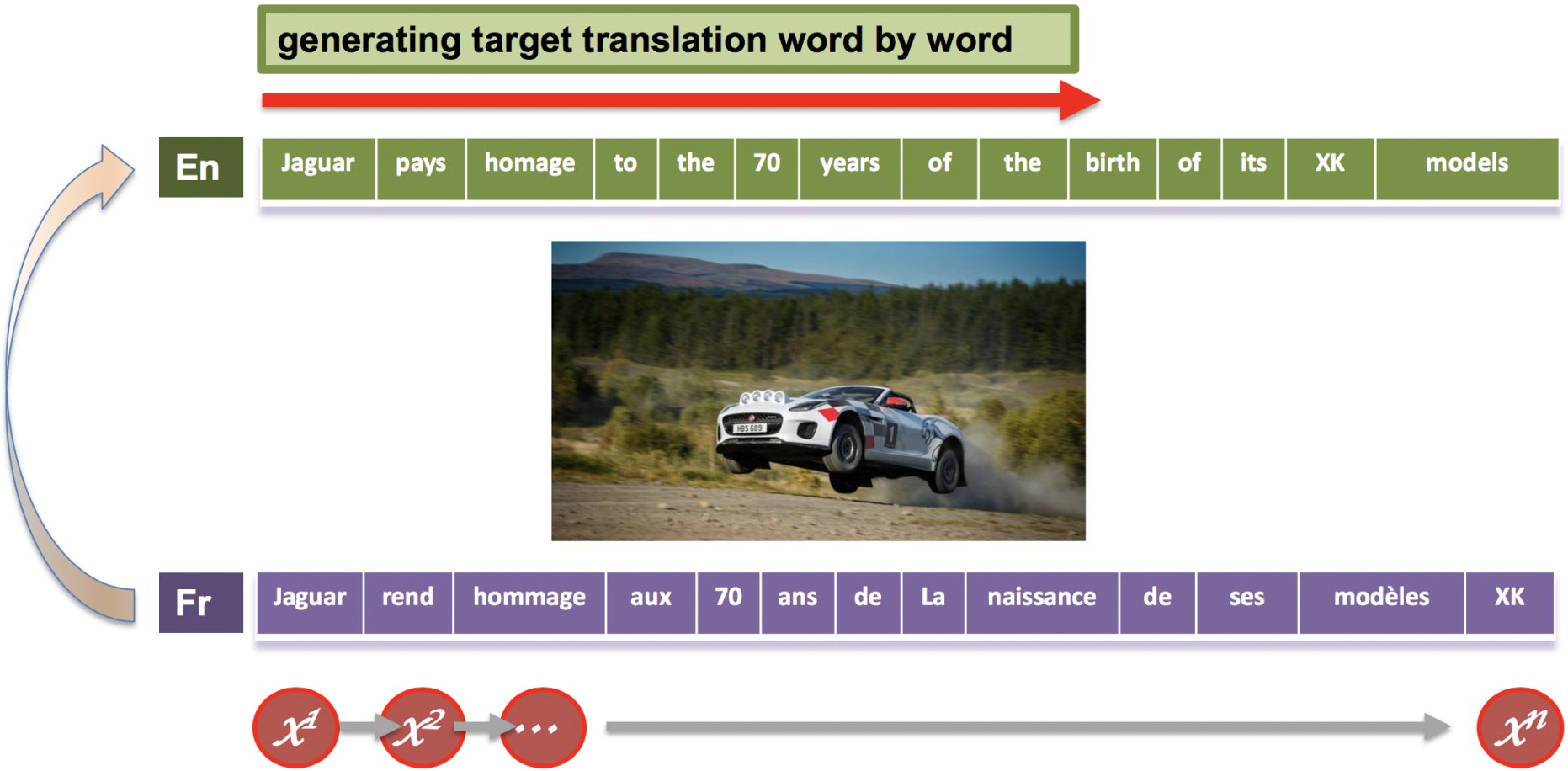
Generating Target (En) One Word at a Time *Contd. I*



Generating Target (En) One Word at a Time *Contd. II*



Generating Target (En) One Word at a Time *Contd. III*



Encoder Details

1: Encoder

Encoder reads the input sentence $\mathbf{x}$ in to a vector c

$$\mathbf{x} = (x_1, \ldots, x_{T_s})$$

Common to use an RNN:

$$h_t = f(x_t, h_{t-1})$$

and

$$c = q(h_1, \ldots, h_{T_x})$$

– where h_t is a hidden state at time step t and c vector of hidden states, and f and q are **nonlinear functions** (e.g., LSTM)

Decoder Details

Decoder

- **Decoder trained to predict next word** $y_{t\prime}$ given (1) the context vector c and (2) all the previously predicted words $\{y_1, \ldots, y_{t\prime-1}\}$
- i.e., The **decoder defines a probability over the translation y** by decomposing the joint probability into the ordered conditionals:

2: Decoder

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | \{y_1, \ldots, y_{t-1}\}, c)$$

where

$$\mathbf{y} = (y_1, \ldots y_{T_y})$$

Decoder Details *Contd.*

3: Decoder *Contd.*

With an **RNN**, each conditional probability is modelled as:

$$p(y_t|\{y_1, \ldots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

where g is a nonlinear function that outputs the probability of y_t and s_t is the hidden state of the RNN.

Neural Machine Translation *Contd.*

Compressing Long Sentences

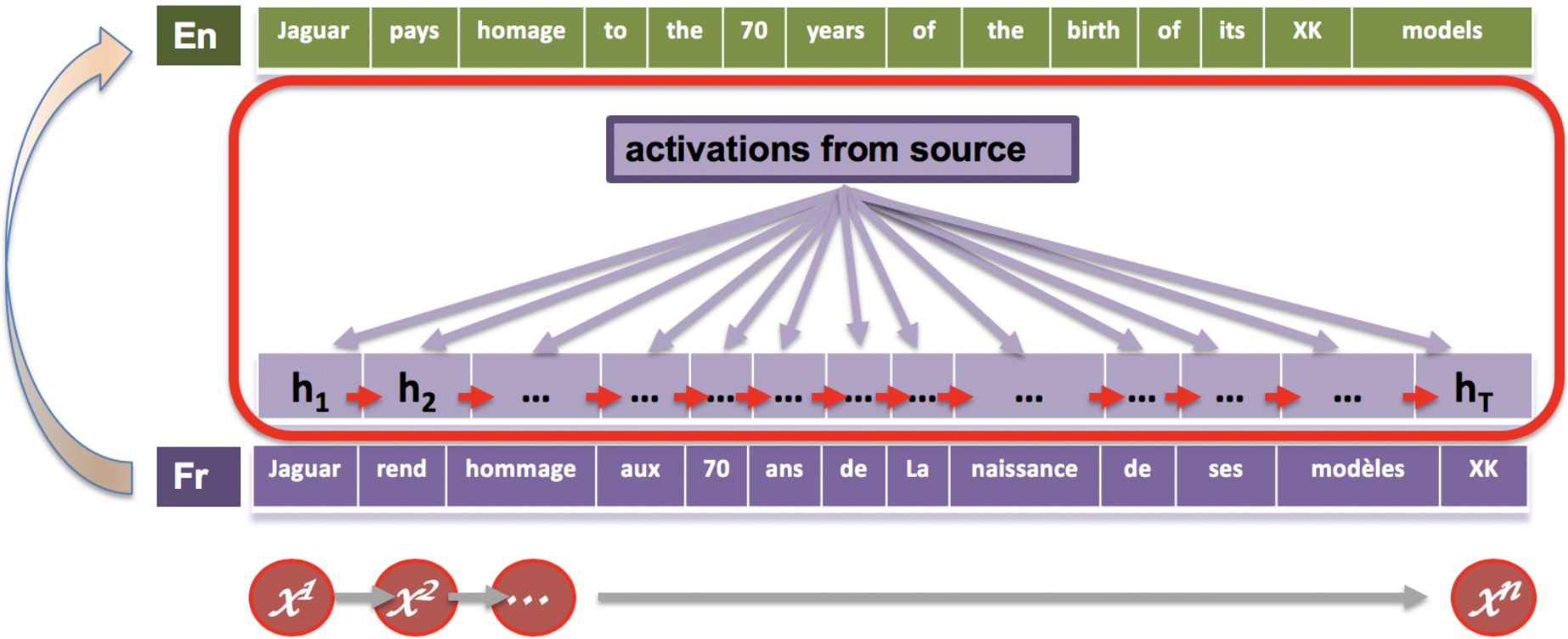
- **Issue:** Difficult to compress all the necessary information of a source sentence into a fixed-length vector, especially for **long sentences** (Cho et al., 2014)
- **Bahdanau et al. (2015)**: learn to 'align' and translate jointly:
  - A: **Generate a word** in a translation,
  - B: **(soft-)search for a set of positions in a source sentence** where the most relevant information is concentrated
  - C: **Predict a target word** based on the context vectors associated with (1) these source positions and (2) all the previous generated target words

Bahdanau et al. (2015) in a Nutshell

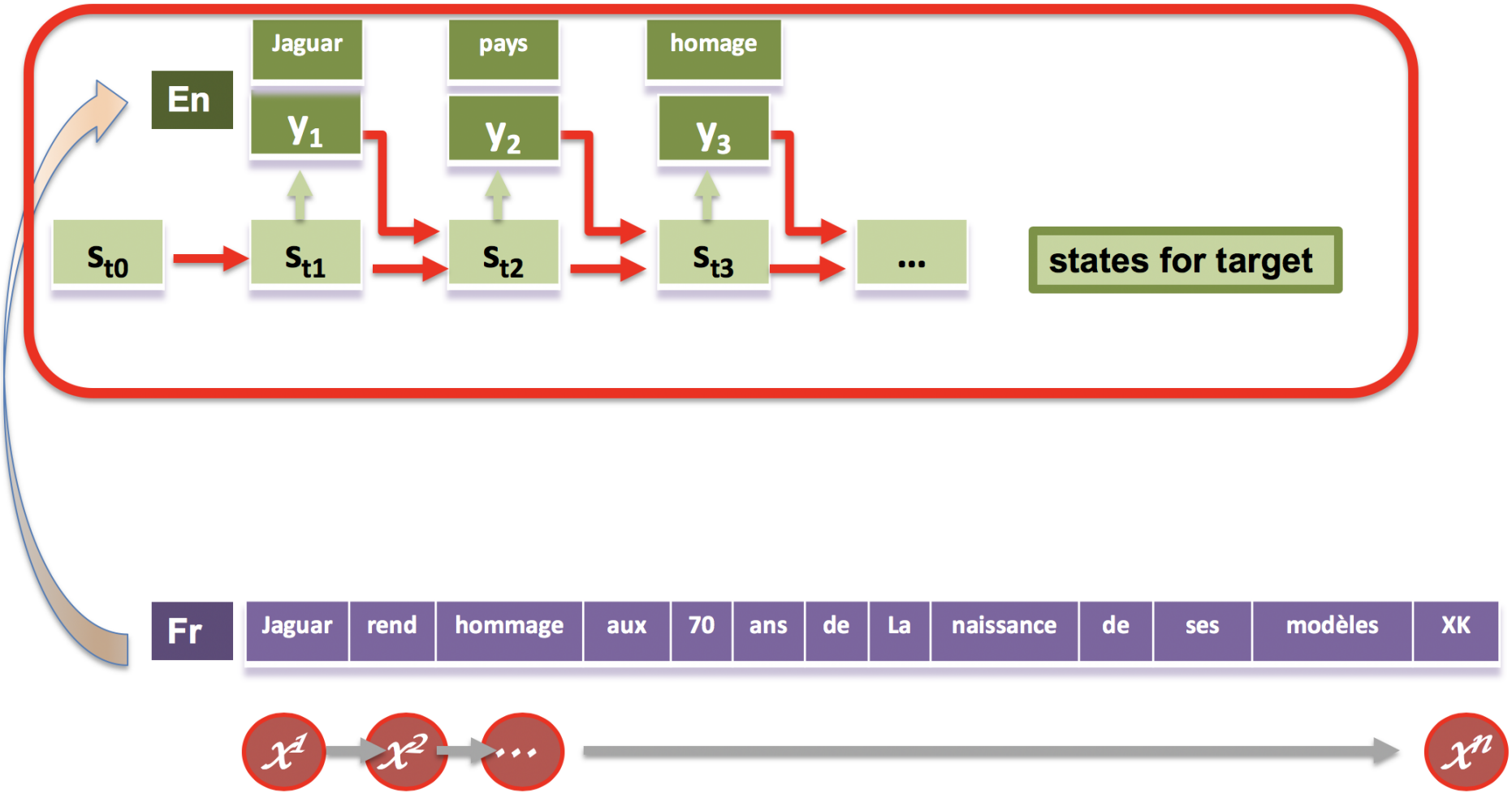
Compressing Long Sentences

- **Bahdanau et al. (2015) advantage**: Does not attempt to encode a whole input sentence into a single fixed-length vector
- **Instead**:
  - A: Encodes the input sentence into a sequence of vectors
  - B: Chooses a subset of these vectors adaptively while decoding the translation
- New approach **frees a neural translation model from having to squash all the information of a source sentence into a fixed-length vector**
- Copes **better with long sentences**

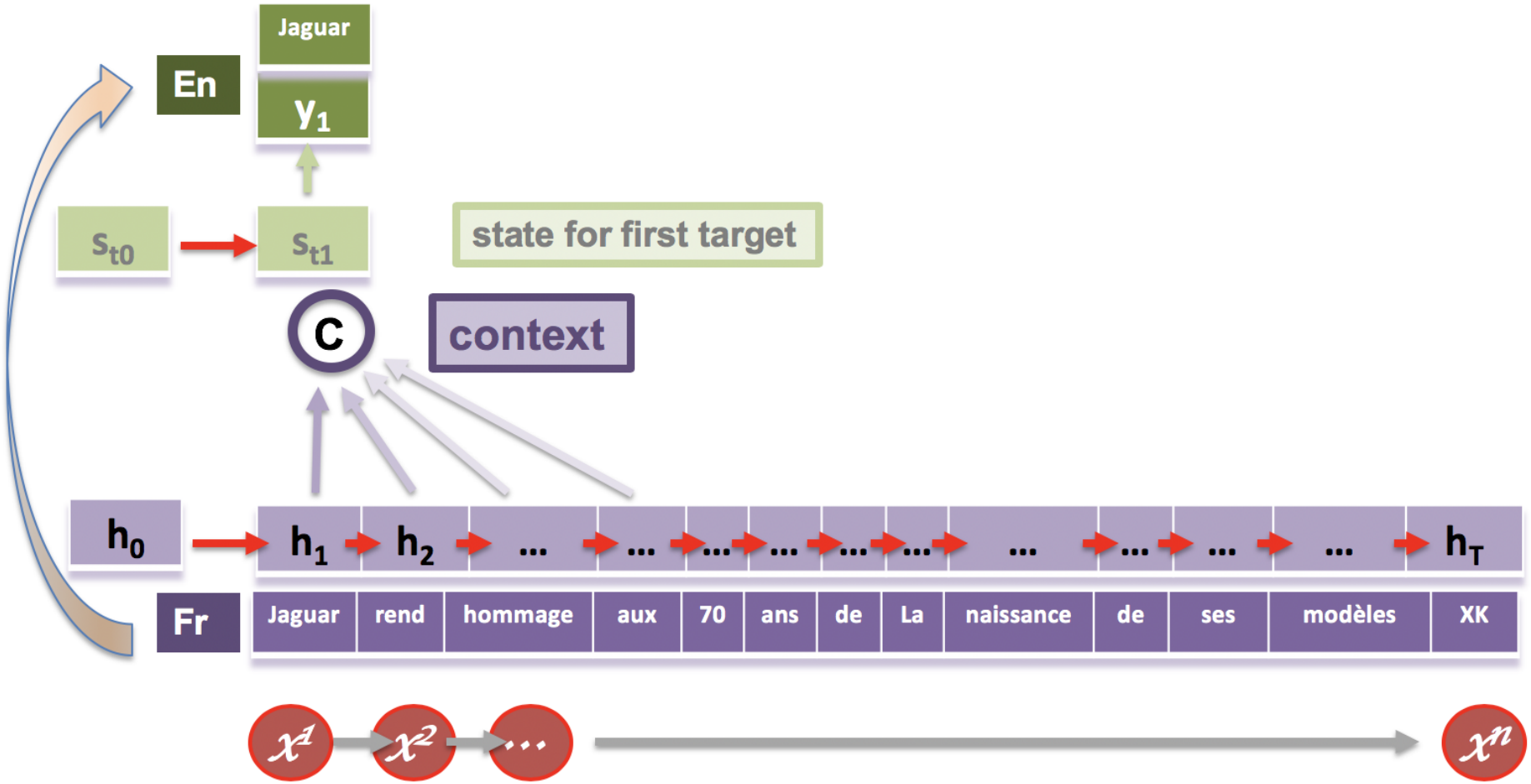
Source Language Activations



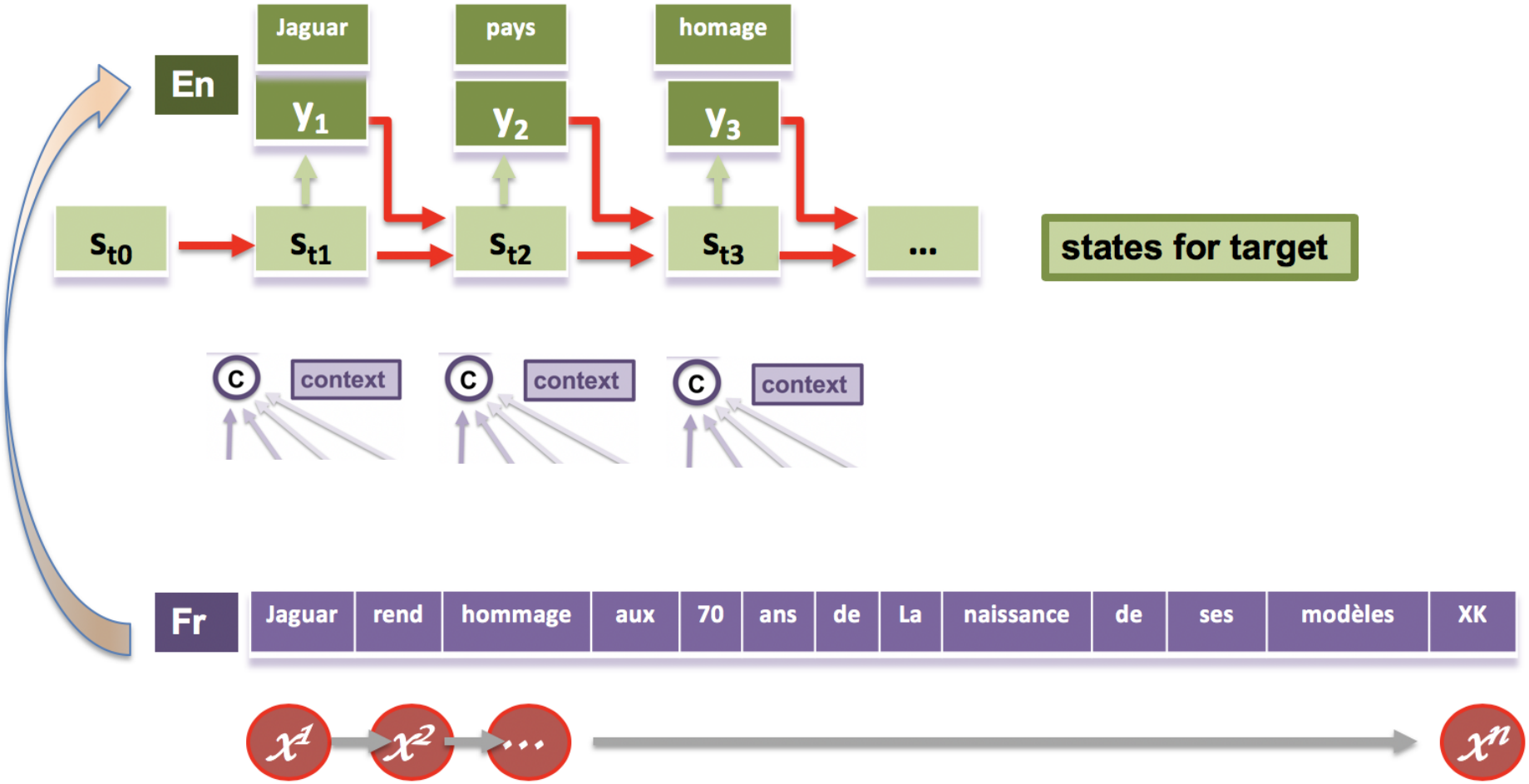
Target (Language) States



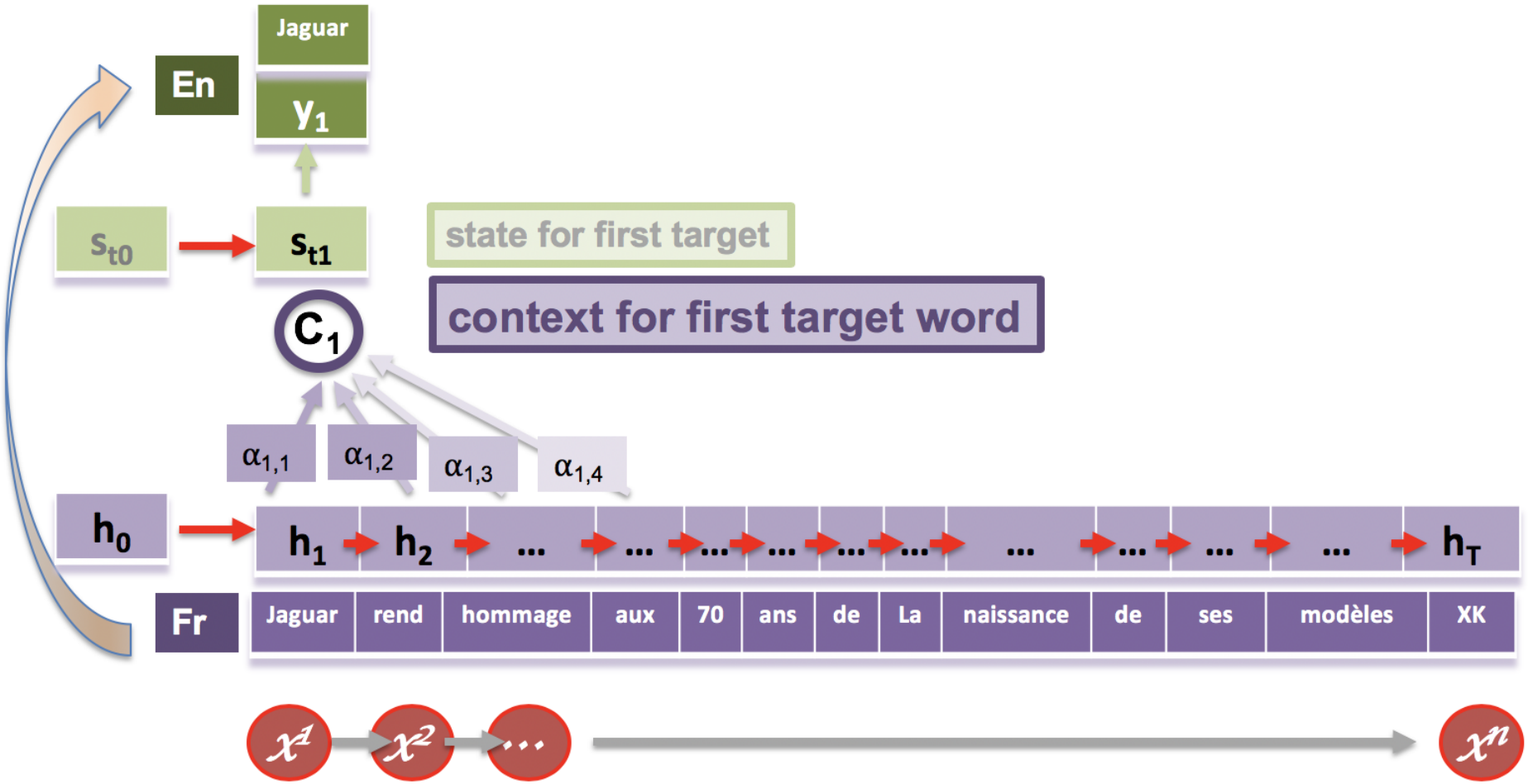
Context


En
Jaguar
y_1
s_{t0}
s_{t1}
state for first target
C
context
h_0
h_1
h_2
...
h_T
Fr
Jaguar
rend
hommage
aux
70
ans
de
La
naissance
de
ses
modèles
XK
x^1
x^2
...
x^n

Context Per Every Target State



Context For State One



What is in Context C1?

4: Context C1

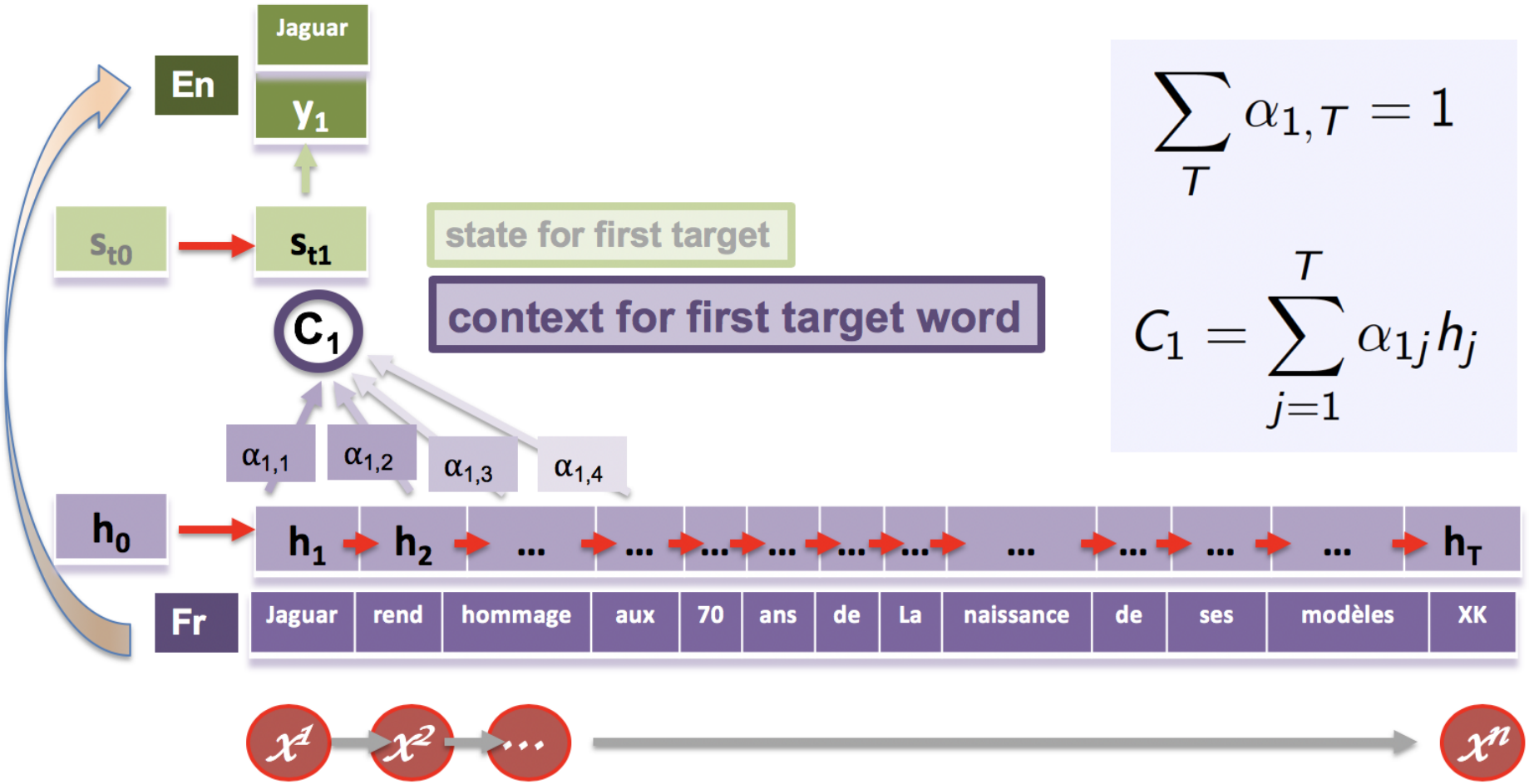
$$\sum_T \alpha_{1,T} = 1$$

$$C_1 = \sum_{j=1}^{T} \alpha_{1j} h_j$$

Context Ingredients

- α: attention weights
- h: activation from source

Context For State One Enhanced



$$\sum_T \alpha_{1,T} = 1$$

$$C_1 = \sum_{j=1}^{T} \alpha_{1j} h_j$$

What is in Context C2?

5: Context C2

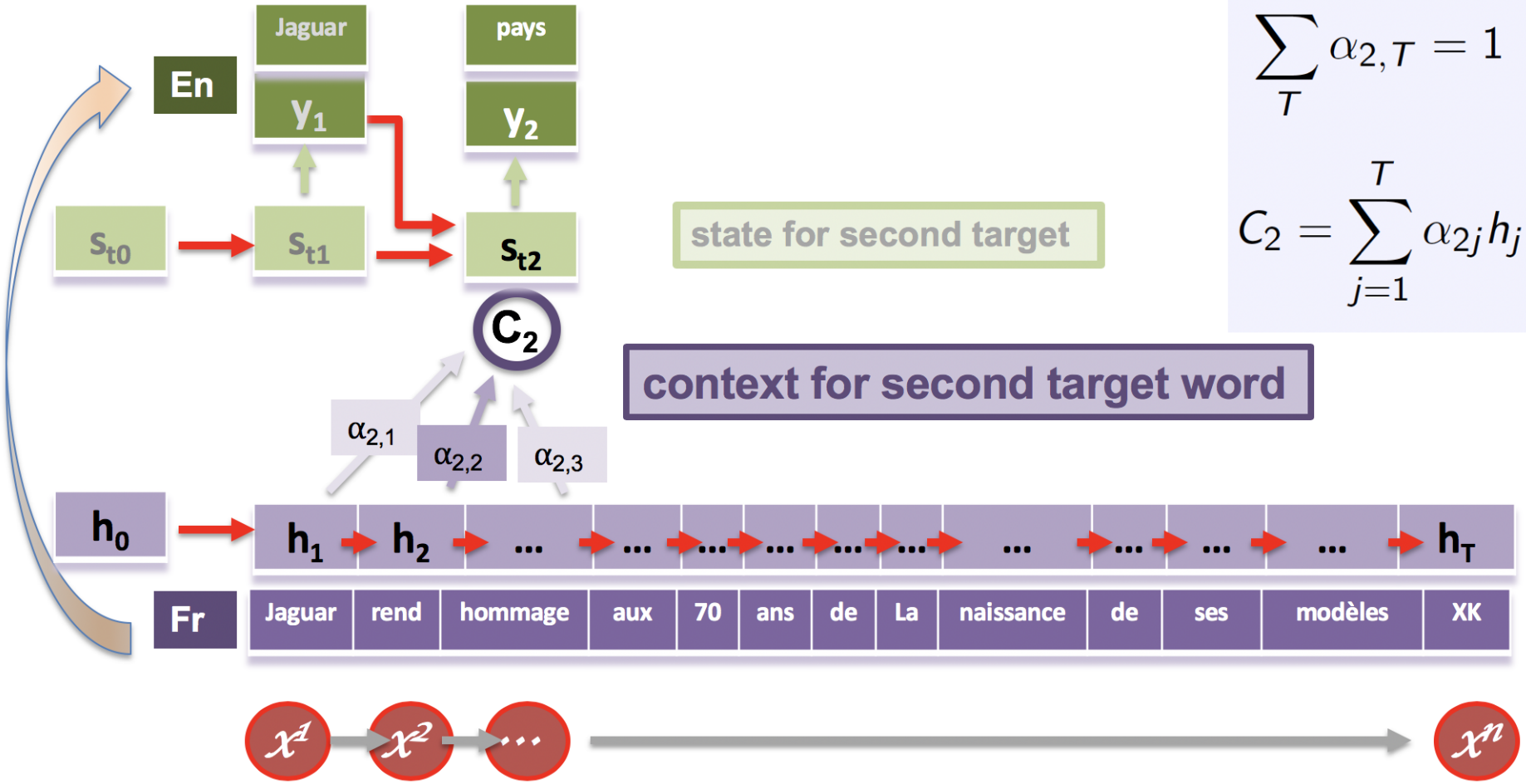
$$\sum_T \alpha_{2,T} = 1$$

$$C_2 = \sum_{j=1}^{T} \alpha_{2j} h_j$$

Context Ingredients

- α: attention weights
- h: activation from source

Context For State Two



state for second target

$$\sum_T \alpha_{2,T} = 1$$

$$C_2 = \sum_{j=1}^{T} \alpha_{2j} h_j$$

context for second target word

What is in Context C3?

6: Context C3

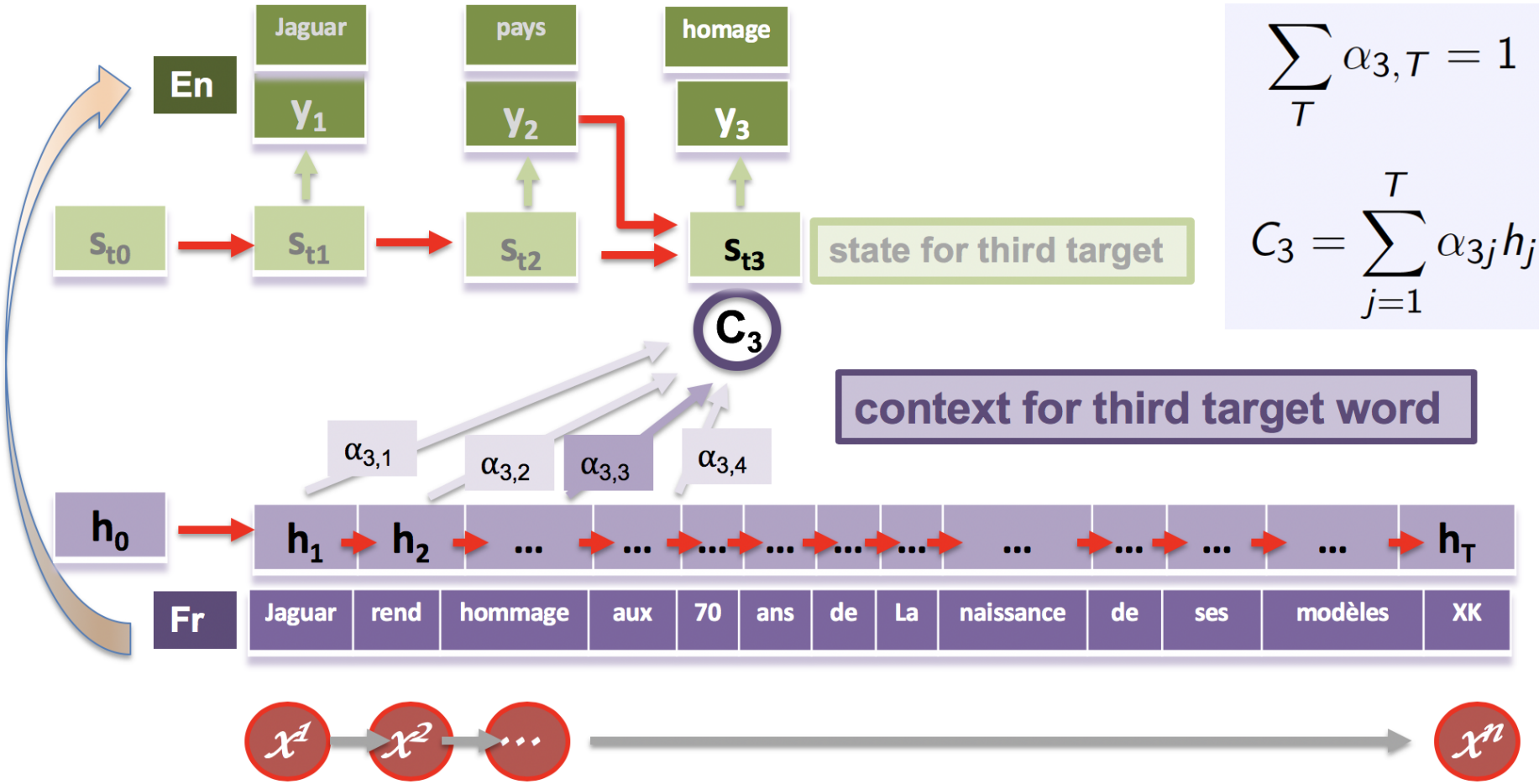
$$\sum_{T} \alpha_{3,T} = 1$$

$$C_3 = \sum_{j=1}^{T} \alpha_{3j} h_j$$

Context Ingredients

- α: attention weights
- h: activation from source

Context For State Three



$$\sum_{T} \alpha_{3,T} = 1$$

$$C_3 = \sum_{j=1}^{T} \alpha_{3j} h_j$$

Decoder: Conditioning on Distinct C_i For Each y_i

7: New Decoder

Recall, original decoder:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | \{y_1, \ldots, y_{t-1}\}, c)$$

For new decoder, each conditional probability defined as:

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x} = g(y_{i-1}, s_i, c_i)$$

where s_i is an RNN hidden state for time i, computed by:

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

What is in Context C?

Context Ingredients

- The context vector c_i **depends on a sequence of annotations** $(h_1, \ldots, h_{T_1})$

8: Context C

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

Context Ingredients

- α: attention weights
- h: activation from source

Weight α_{ij}

9: Weight α_{ij} for each annotation h_j

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

where

$$e_{ij} = a(s_{i-1}, h_j)$$

is an alignment model based on the RNN hidden state s_{i-1} (just before emitting y_i) and the j-th annotation h_j of the input sentence

Alignment Model

- **alignment model** a scores how well the inputs around position j and the output at position i match.

Computing Soft Alignment for Model α

Alignment Model

- The **alignment model** *a* **directly computes a soft alignment**, which allows the gradient of the cost function to be backpropagated through
- This **gradient can be used to train the alignment model as well as the whole translation model jointly**

Role of Attention

- "By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixedlength vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly" (Bahdanau et al., 2015)

Bahdanau et al. (2015) Attention Model Visualization

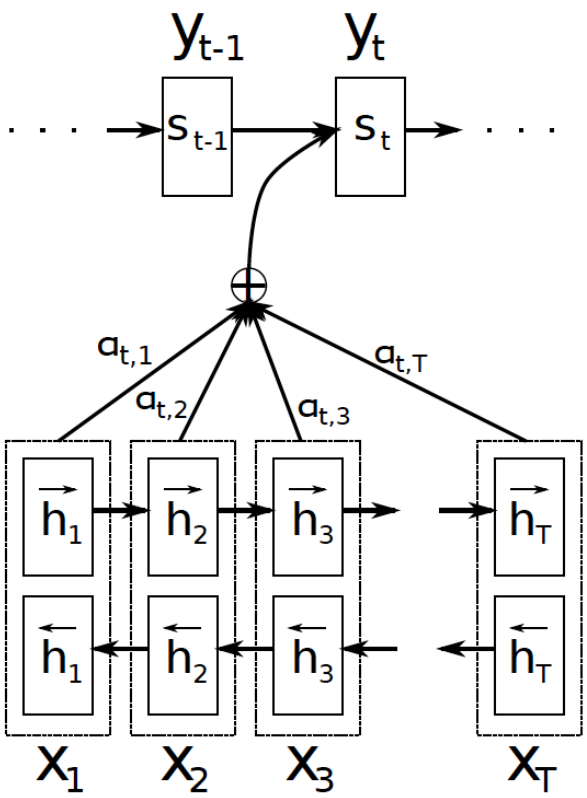


Figure: Bahdanau et al. (2015) attention model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$

Bahdanau et al. (2015) Weight Visualization

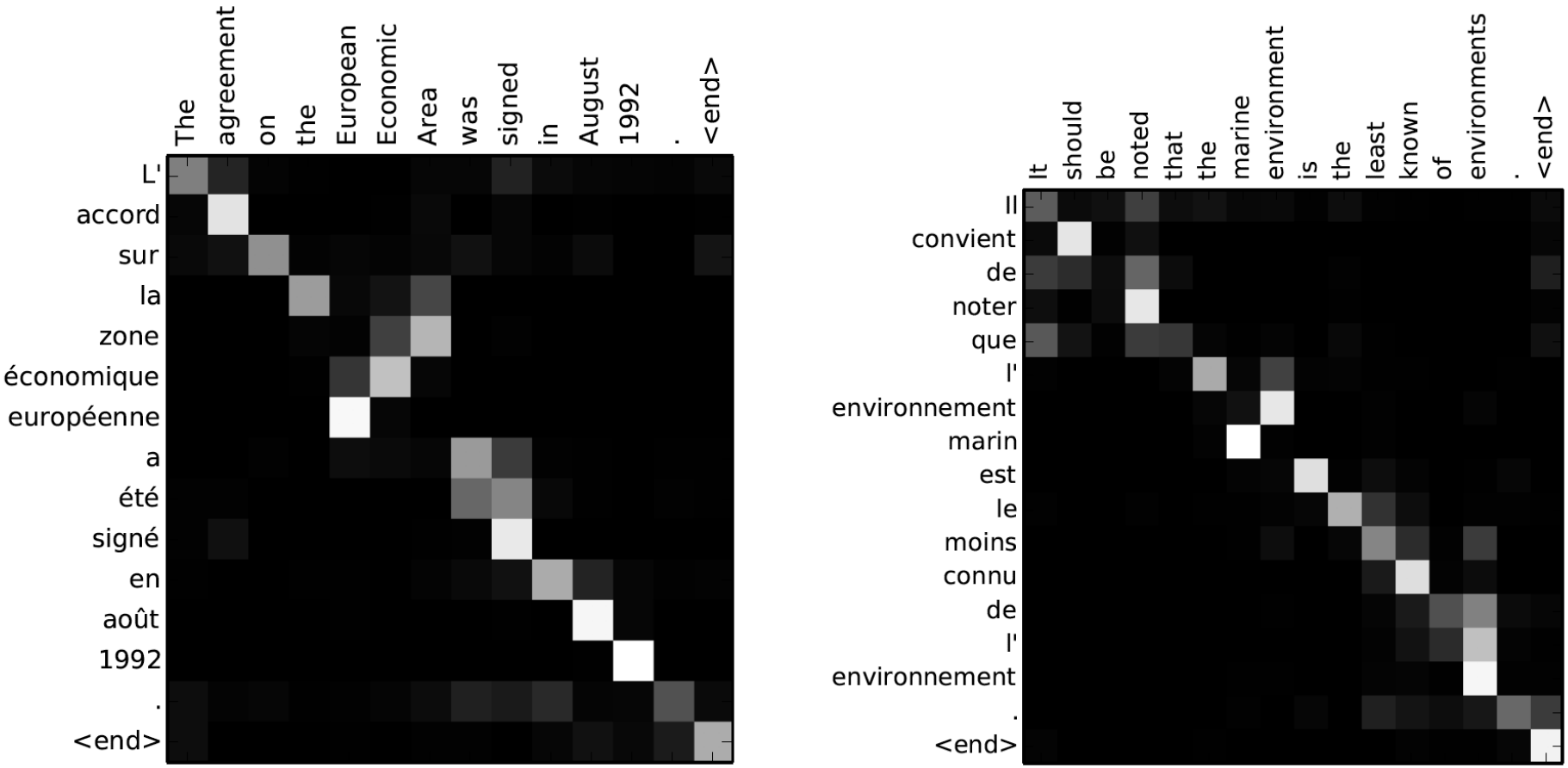


Figure: Each pixel shows the weight ij of the annotation of the j-th source word for the i-th target word (Bahdanau et al., 2015).

Image Caption Generation (With Attention)

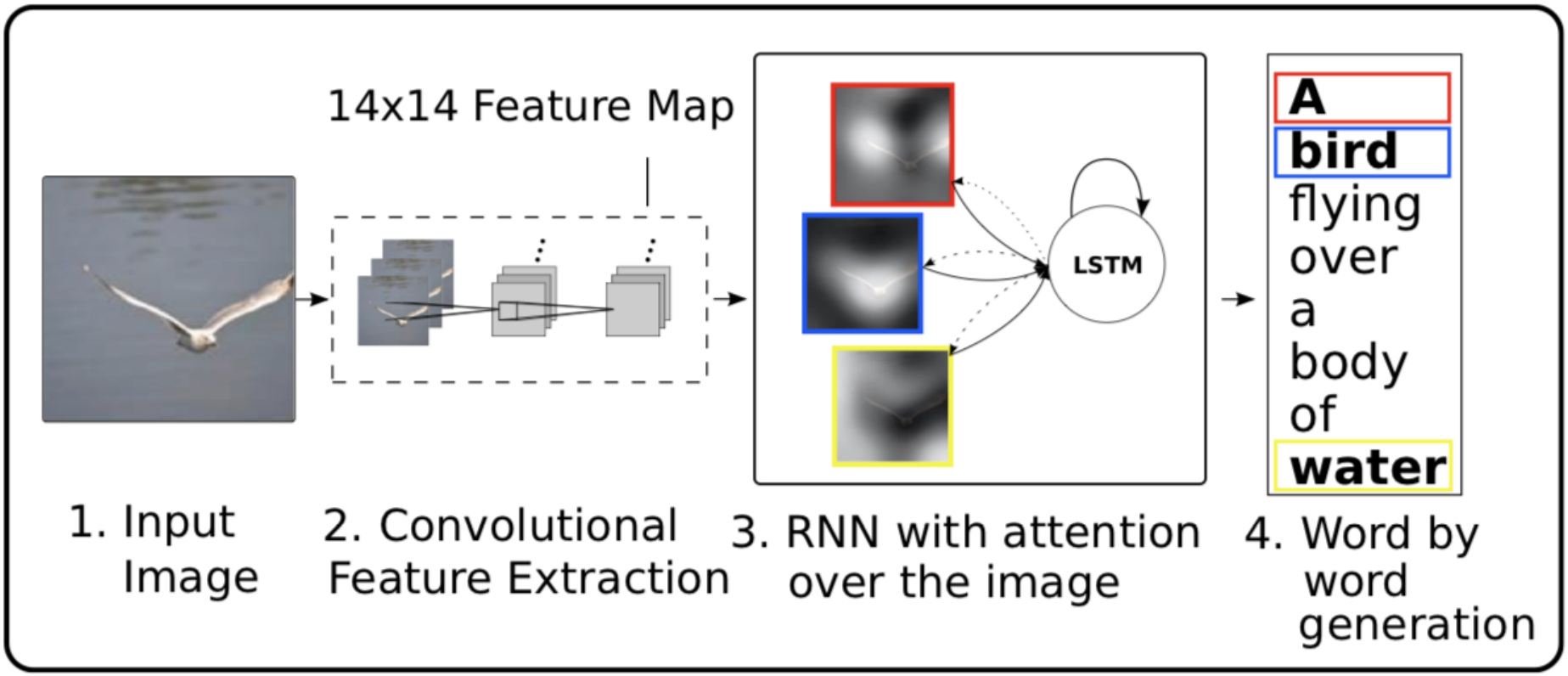


Figure: Xu et al. (2016) Show, Attend, and Tell . . . . A model that learns a words/image alignment.

Visual Attention

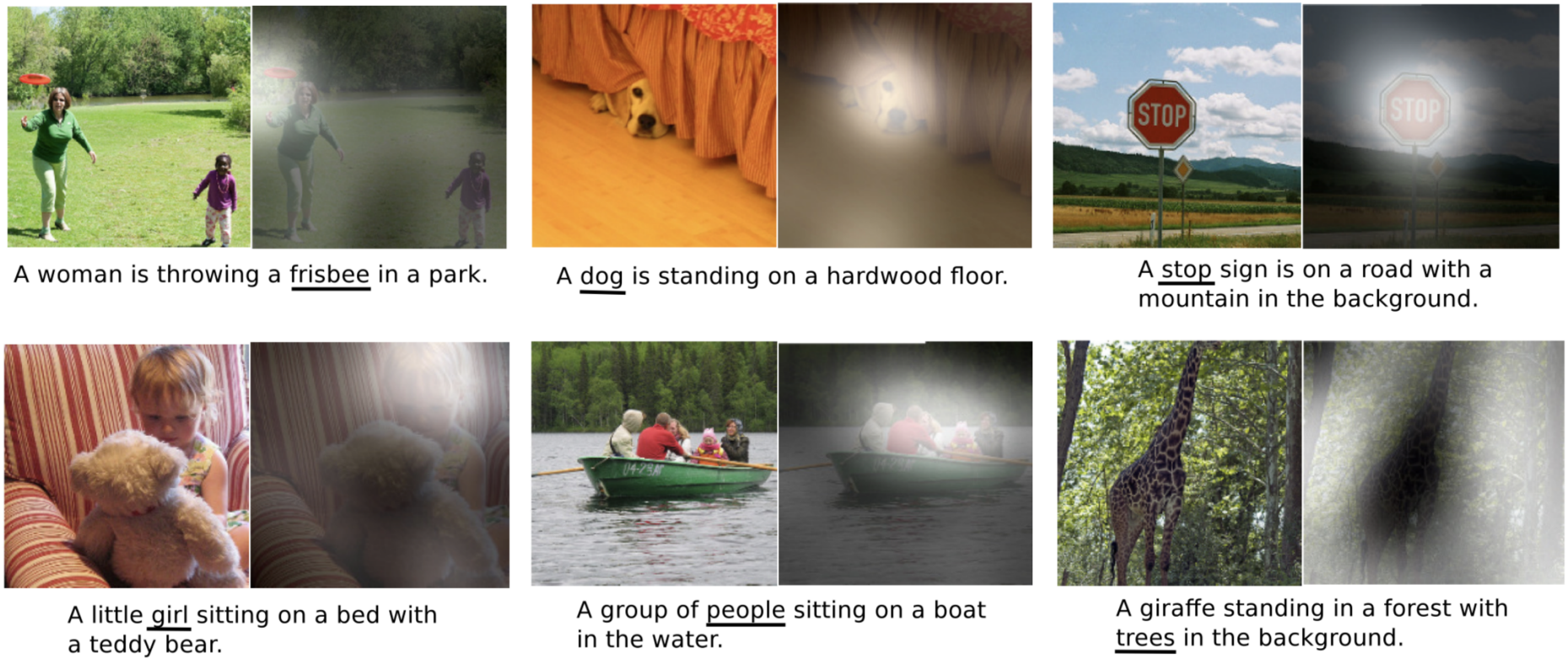


Figure: Xu et al. (2016): Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention

Visual Attention: Model Errors

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Figure: Errors from Xu et al. (2016)