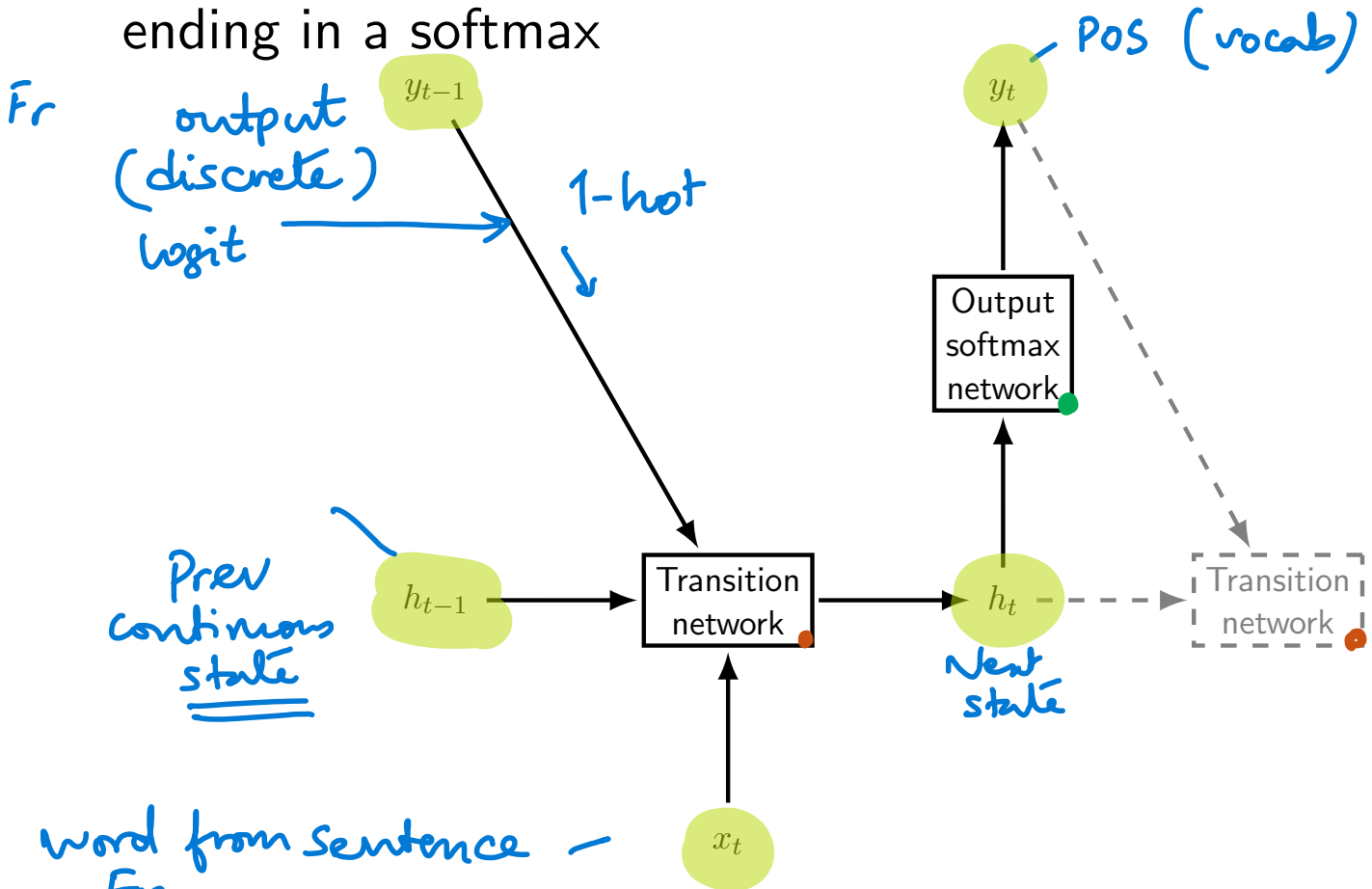
Embeddings and neural text analysis

Basic sequence network template

- Discrete state y_t replaced with continuous vector states h_t; usually $h_0 = \vec{0}$
- The transition network inputs previous state h_{t-1} and current input x_t (also continuous form) and outputs current state h_t
- y_t for end application derived from h_t using an output network ending in a softmax



Basic recurrent network (RNN)

- The transition network and the output network are standard multi-layer neural networks with linear combinations and nonlinearities at every hidden layer

  Elman network:
  $$h_t = \sigma(x_t W_h + h_{t-1} U_h + b_h)$$
  $$y_t = \sigma(h_t W_y + b_y)$$

  Jordan network:
  $$h_t = \sigma(x_t W_h + y_{t-1} U_h + b_h)$$
  $$y_t = \sigma(h_t W_y + b_y)$$

- Weights W_h, U_h, b_h of transition networks at all positions are tied; same for weights W_y, b_y in the output network
- RNNs are Turing complete
- Stream of consciousness
- Potential problems: vanishing or exploding gradient
  - I grew up in France ... I speak fluent French

Gated recurrent unit (GRU) [2]

- Logistic gates that decide what fraction of input and previous state to retain/forget

$$
\begin{aligned}
\text{Reset} \quad & r_t = \sigma(x_t W_r + h_{t-1} U_r + b_r) \\
\text{Combine} \quad & c_t = \tanh(x_t W_h + (r_t \circ h_{t-1}) U_h + b_h) \\
\text{Update} \quad & z_t = \sigma(x_t W_z + h_{t-1} U_z + b_z) \\
\text{Output} \quad & h_t = (1 - z_t) \circ h_{t-1} + z_t \circ c_t
\end{aligned}
$$

- σ is sigmoid; other nonlinearities specified explicitly
- r_t decides if c_t is computed as in a basic RNN, or h_{t-1} is discarded and only x_t is used
- z_t decides how to blend c_t and h_{t-1} as-is
- If we set by force $r_t \approx 1$ and $z_t \approx 1$, equivalent to basic RNN
- https://arxiv.org/abs/1412.3555

Long short-term memory (LSTM) [3]

- Slightly different arrangement of gates

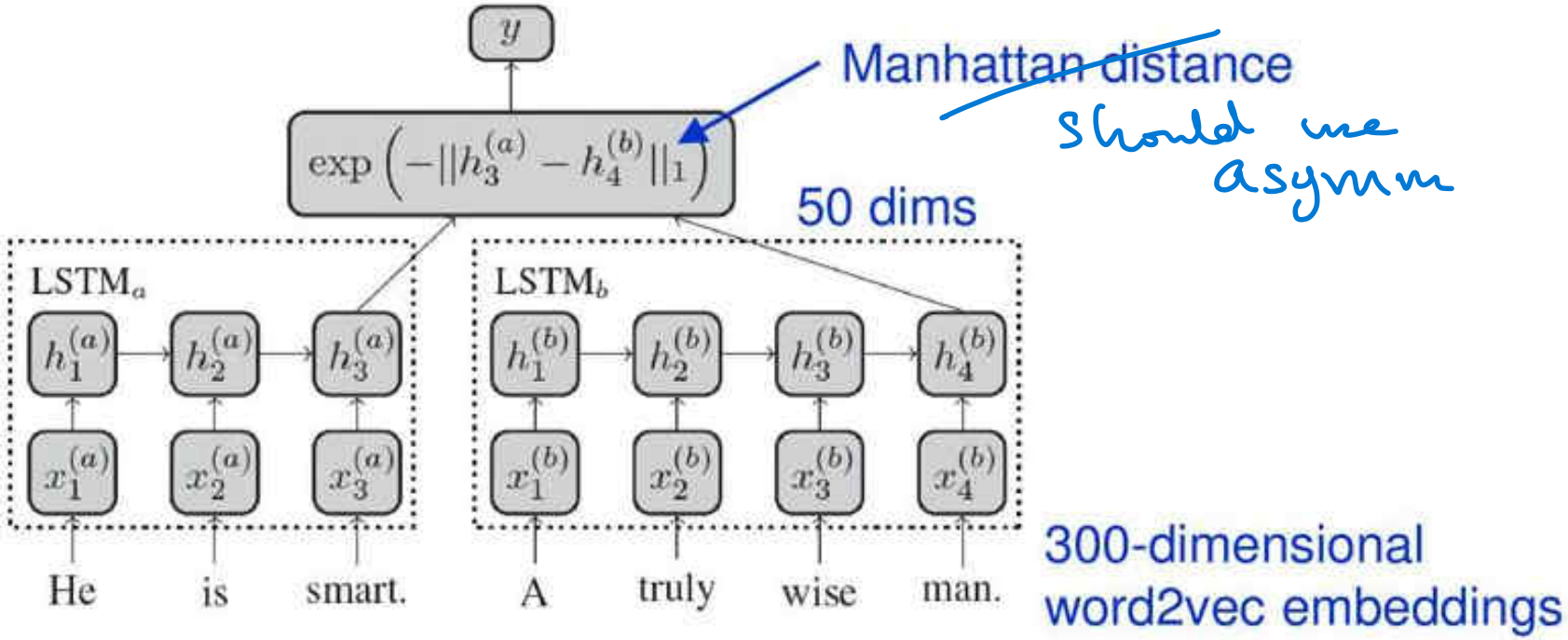
In	$i_t = \sigma(x_t U_i + h_{t-1} W_i + b_i)$
Forget	$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f)$
Out	$o_t = \sigma(x_t U_o + h_{t-1} W_o + b_o)$
RNN	$g_t = \tanh(x_t U_g + h_{t-1} W_g + b_g)$
Memory	$c_t = c_{t-1} \circ f_t + g_t \circ i_t$
State	$h_t = \tanh(c_t) \circ o_t$

- Accuracy comparable to GRUs
- Widely used for sequence labeling, sequence-to-sequence comparison, sequence-to-sequence translation, etc.
- Usually left-to-right and right-to-left LSTMs in tandem as a bi-LSTM; captures context from both sides
- Also see understanding LSTMs, unreasonable effectiveness

How the states h_t are used (c_t usually not)

- h_t is a compressed representation of the sequence $x_{\leq t}$
- I.e., the left context of position t
- Sometimes written $\overrightarrow{h_t}$
- To get compressed representation of right context, use a separate right-to-left LSTM
- Its states are called $\overleftarrow{h_t}$
- Together, $\overleftrightarrow{h_t} = [\overrightarrow{h_t}, \overleftarrow{h_t}]$ give a compressed representation of the whole text tailored to position t
- Consider sequence-to-discrete label classification, e.g., spam, sentiment
- Feed "average state" $\sum_t \overleftrightarrow{h_t}/T$ as input to a feed-forward network ending in a softmax
- Given a query, infer if a sentence/passage is relevant
- Given two sentences, find if one implies the other

Does one sentence imply another?



Quotable expletives and discontent:

"You can't cram the meaning of a whole %&!$# sentence into a single $&!#* vector!" — Ray Mooney (2014)

"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster."
— Geoffrey Hinton (2014)

RNNs as language models and translators

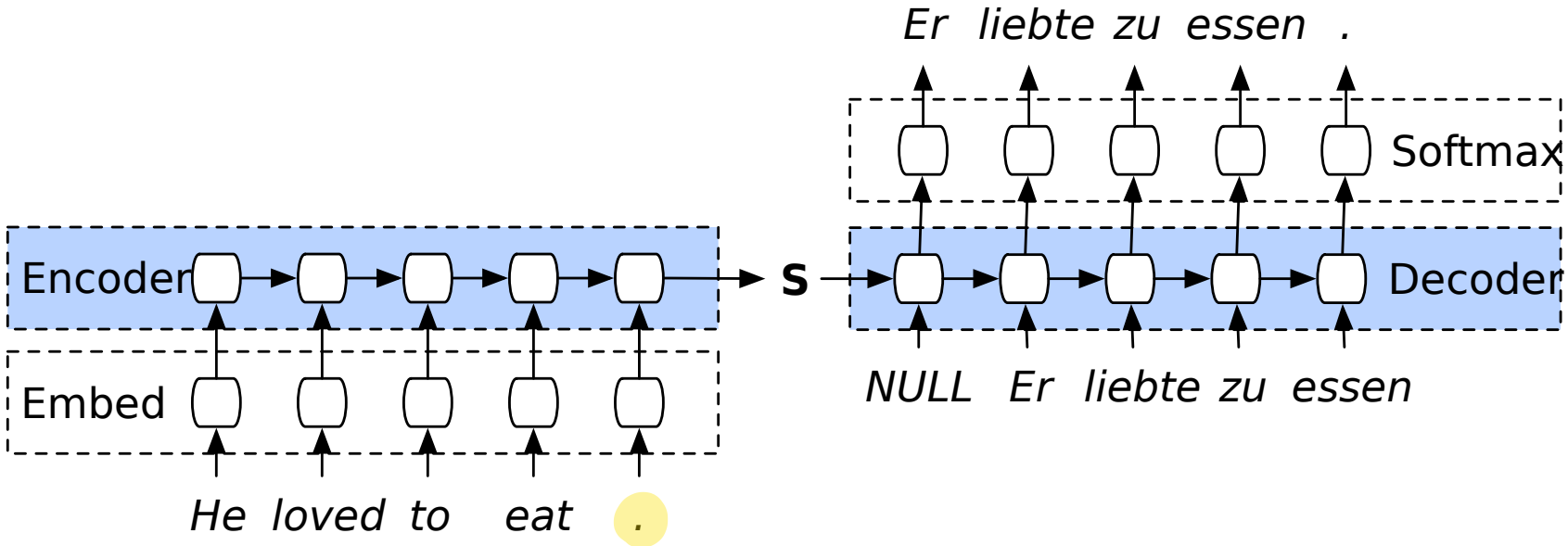
- Suppose input is word sequence (x_t)
- Whose embeddings are inputs to an RNN
- The label sequence (y_t) can be defined as looking ahead one step in the same word sequence, i.e., (x_{t+1})
- E.g., having read "I speak fluent_____", what next word do we expect?
- Thus, an unsupervised corpus can lead to loads of supervised "y_t" label data
- Word embeddings plus LSTM cell model weights constitute "language model"
- Can hallucinate synthetic text passages
- Can estimate (comparative) probabilities of given sentences

RNNs as language models and translators (2)

- Instead of word2vec or GloVE style word embeddings ...
- ... where 'bank' or 'interest' has only one embedding each
- Suppose I publish both the LSTM model weights and the word embeddings as trained through the LSTMs
- Then you run the LSTM through sentences like
  - He swam close to the river bank
  - The bank was giving low interest loans
  - Her interest in music was minimal
- Then $\overleftrightarrow{h}_{\text{bank}}$ and $\overleftrightarrow{h}_{\text{interest}}$ will be quite different in various sentences
- I.e., context-sensitive word embeddings

RNNs as language models and translators (3)

- Use of RNNs in translation (sequence to sequence)

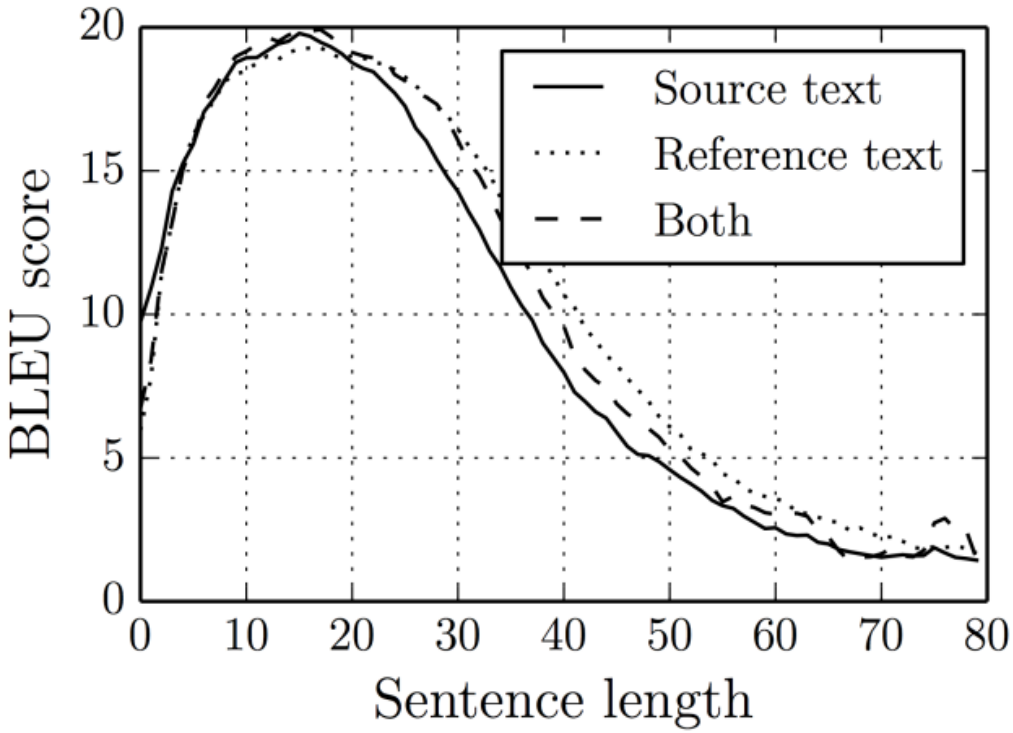


- First encode the English sentence
- Ending with a special marker '.'
- "Meaning" of English sentence captured in vector $\mathbf{S}$
- Send in *NULL*, starting the German decoder

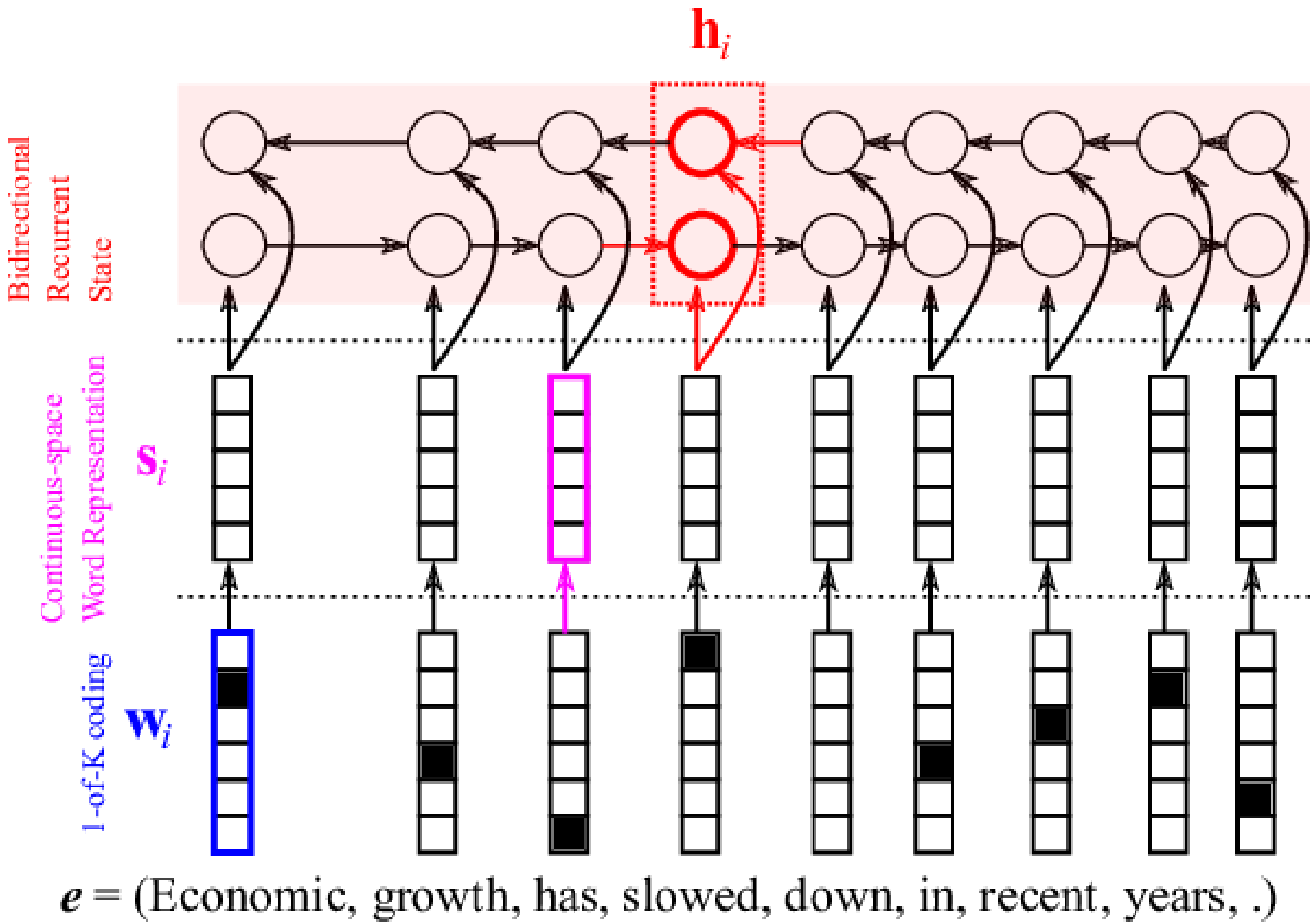
Fixed size passage encoding

- Fixed passage encoding dimension independent of passage length is a problem
- Attention to the rescue

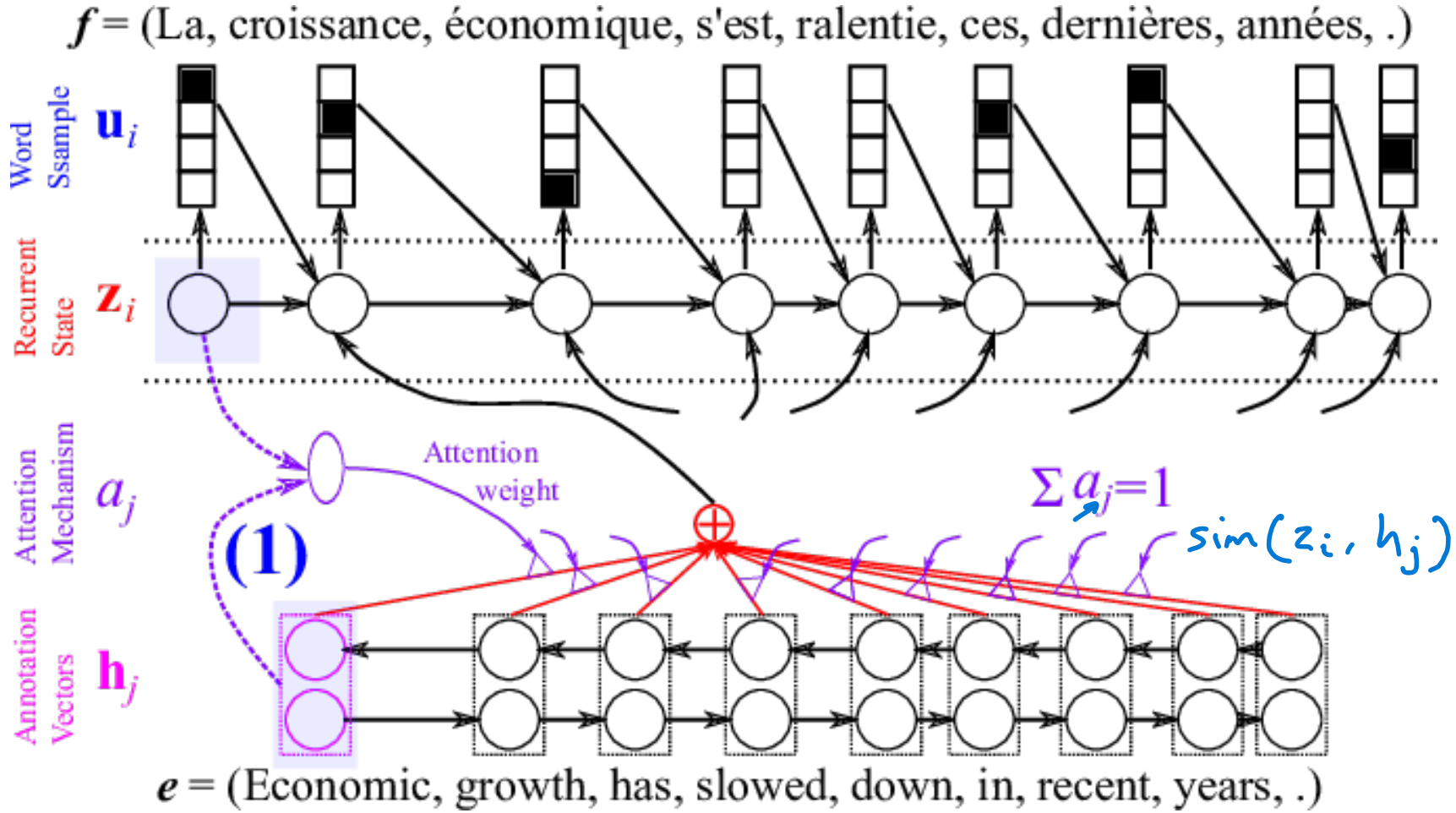
BLEU(gold Fr sent., system Fr output)



Encoding the English sentence

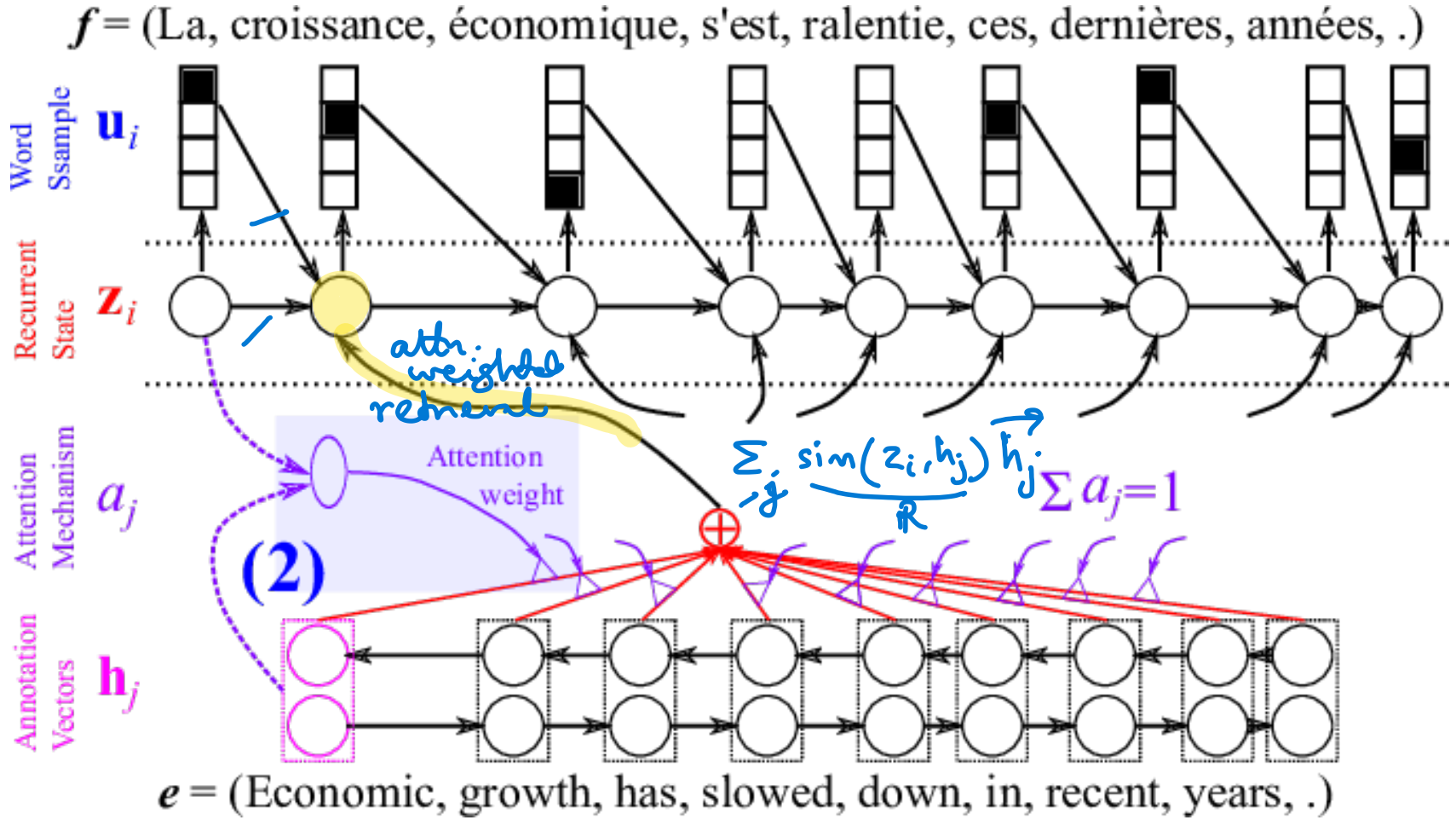


Evaluating attention from target position [1]



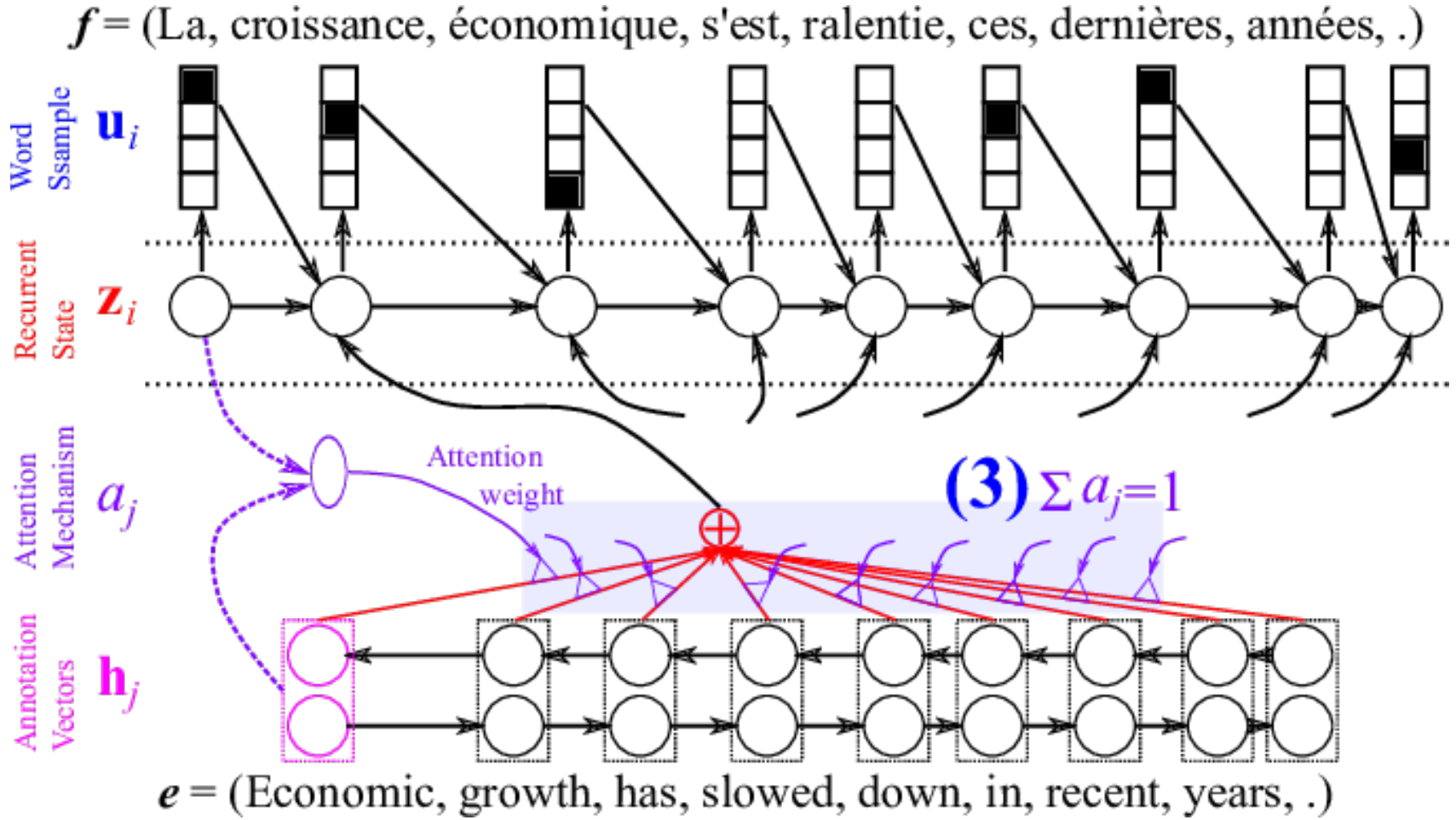
- For each target position i
- Choose one source position j
- Whose state h_j, along with z_{i-1} informs decoder

Computing attention-weighted average English state



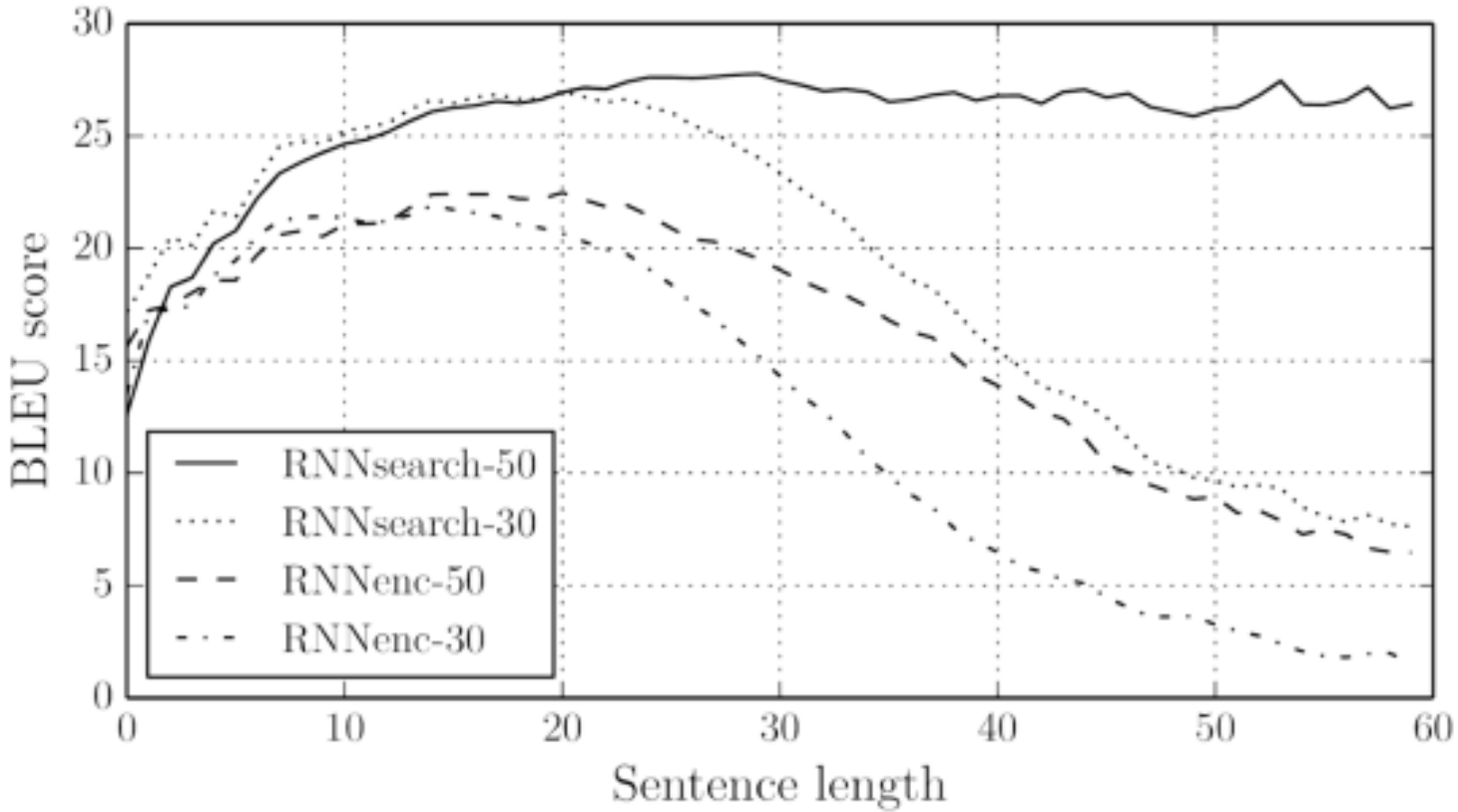
- Choose h_j in a soft manner to support loss backprop
- Compute a_j as some network function of h_j and z_i
- Normalize using softmax to add up to 1 and increase skew
- Compute $\sum_j a_j h_j$ and feed into decoder's $(i+1)$th cell

Emitting next French word



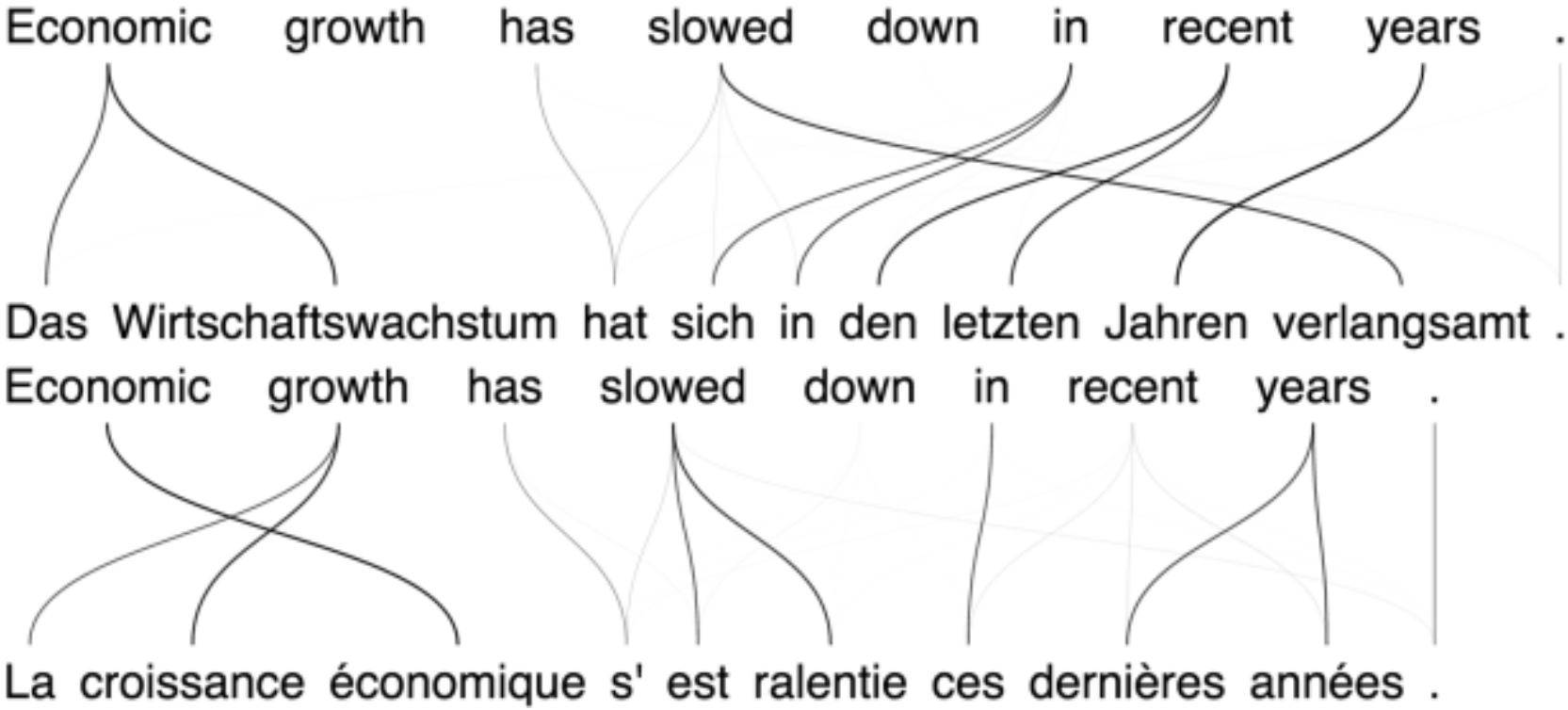
- Decoder network itself is not bidirectional
- Both i and i' can choose a common $j*$
- But each i can (soft-) select only one j

Improved BLEU score



Paper

Attention strength examples



- *Wirtschafts* means *economic*
- *Wirtschaftswachstum* means *economic growth*
- Note the missing line from *growth*
- Also, *dernières* means *latest*, faint link to *recent*
- Should allow many-to-many 'fractional' links in general [4]

Self-attention

- Can we "fake recurrence" by replacing l-to-r or r-to-l recurrence with several layers of (self-)attention?
- Raw (embedding) inputs x_t
- Map these to keys k_t, queries q_t, and values v_t
- Via $k_t \leftarrow W_K x_t + b_K, \quad q_t \leftarrow W_Q x_t + b_Q, \quad v_t \leftarrow W_V x_t + b_V$
- For each position t, calculate attention logits over other positions τ via $e_{t,\tau} = q_t \cdot k_\tau$
- Note that $E = (e_{t,\tau})$ is a $T \times T$ matrix
- Take softmax as $a_{t,\tau} = \mathrm{softmax}_\tau(e_{t,\tau})$
- I.e., softmax each row of E: for each t, what is the affinity of the original token at t to each other token?
  - The bank is giving very low interest rates
  - The drama proved that comedy is alive and well as a genre
- Compute attention-weighted values as outputs $o_t = \sum_\tau a_{t,\tau} v_\tau$

Self-attention (2)

- One's output $o_t^{(\ell)}$ is another's input $x_t^{(\ell+1)}$ (rinse and repeat)
- How to train?
- Hide ('mask') some fraction of tokens and predict them from o_t
- (word2vec was doing something similar, predicting one token at a time)
- Add a gazillion bells and whistles to get ... Transformers!
- What do BERT, RoBERTa, ALBERT, SpanBERT, DistilBERT, SesameBERT, SemBERT, SciBERT, BioBERT, MobileBERT, TinyBERT and CamemBERT all have in common?

General notation for GRUs and LSTMs

- In general we can use 'RNN' as an abstract cell:
$$\boldsymbol{h}_t = \mathsf{RNN}(\boldsymbol{h}_{t-1}, x_{t-1})$$
- $\boldsymbol{h}_t$ is the recurrent state and x_t is the input
- We regard the input x_t as 1-of-Σ discrete, written as a 1-hot vector, and looked up in an embedding matrix to send a vector into the RNN cell
- A separate output network converts states to distributions over the output[1] vocabulary Σ, from which output symbol y_t can be sampled
$$y_t \sim \mathsf{OUT}(\boldsymbol{h}_t) \in \Delta^{\Sigma},$$
where Δ^{Σ} is the multinomial distribution over $|\Sigma|$ outcomes
- Standard RNNs and GRUs fit this exactly
- LSTMs pass *two* vectors $\boldsymbol{c}_t, \boldsymbol{h}_t$ from each position to next
- These can be notationally clubbed into a single $\boldsymbol{h}_t$

General notation for GRUs and LSTMs (2)

- And OUT ignores the $\boldsymbol{c}_t$ part
- Or, we can explicitly write

$$[\boldsymbol{c}_t, \boldsymbol{h}_t] = \text{LSTM}([\boldsymbol{c}_{t-1}, \boldsymbol{h}_{t-1}], x_{t-1})$$
$$y_t \sim \text{OUT}(\boldsymbol{h}_t)$$

- In language modeling applications we have 'gold' y_t^* available, which is just the next word x_t
- Our goal there is to maximize $\text{OUT}(\boldsymbol{h}_t)[x_t]$, or the overall log likelihood

$$\sum_t \log \text{OUT}(\boldsymbol{h}_t)[x_t]$$

- In a seq2seq translation network without attention, there is an encoder RNNe and a decoder RNNd
- During encoding there is no output
- Let the input sequence have length N and output sequence have length M

General notation for GRUs and LSTMs (3)

- The last encoder state is $\boldsymbol{h}_N$
- To avoid confusion name the states of the decoder $\boldsymbol{g}_m$; $\boldsymbol{g}_0 \equiv \boldsymbol{h}_N$
- And let the inputs and outputs of the decoder be y_m
- For the decoder, $y_0 =$ START is a special marker token
- The decoder does have an associated OUT network

$$\boldsymbol{g}_m = \text{RNNd}(\boldsymbol{g}_{m-1}, y_{m-1})$$

$$y_m \sim \text{OUT}(\boldsymbol{g}_m)$$

- Here we have a gold translated sequence $y^*_m, m = 1, \ldots, M$
- 0/1 loss is like $\sum_m [\![y_m \neq y^*_m]\!]$
- Is $\sum_m -\log \text{OUT}(\boldsymbol{g}_m)[y^*_m]$ an ok surrogate?
- In the basic attention model, every output position m is 'explained'/supported by an input position n, chosen in a soft manner

General notation for GRUs and LSTMs (4)

- Encoder RNNe works as before, but all $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ can be accessed by decoder RNNd
- Assume RNNd is a GRU decoder for simplicity
- Attention from output position m on input position n may be modeled as $a_{mn} \propto \exp(\boldsymbol{g}_m \cdot \boldsymbol{h}_n)$
- Normalize to multinomial distribution for each m: $a_{mn} = \exp(\boldsymbol{g}_m \cdot \boldsymbol{h}_n) / \sum_{n'} \exp(\boldsymbol{g}_m \cdot \boldsymbol{h}_{n'})$
- Attention-weighted source state selection for output position m will be $\hat{\boldsymbol{h}}_m = \sum_n a_{mn} \boldsymbol{h}_n$
- Now decoder recurrence is set up as $\boldsymbol{g}_m = \text{RNNd}(\boldsymbol{g}_{m-1}, \hat{\boldsymbol{h}}_m, y_{m-1})$
- And output $y_m \sim \text{OUT}(\boldsymbol{g}_m)$ is sampled as before
- (Mild variations used in the literature)

▸ HW Practice writing vectorized Tf/Pytorch code for above

[1] Input and output vocabularies may be same or different

Summary of using neural networks for text

- Start with individual word embeddings
- ConvNets provide somewhat position-sensitive aggregation of words of a sentence or passage, without demanding hard parses
- If parses are available, can compose word embeddings along the parse structure
- (Not all compositions are valid, e.g., "New York", "rat race", "red carpet", "snake oil")
- Counterpart to chain CRFs is the RNN family
- Exploding and vanishing gradients
- Addressed by GRU and LSTM
- Applied in parsing, entailment testing, machine translation, question answering, ...
- Various attention mechanisms boost performance