

Natural Language Processing with Recurrent Neural Networks

2023–24

Ion Androutsopoulos

<http://www.aueb.gr/users/ion/>

Contents

- Recurrent neural networks (RNNs), GRUs/LSTMs.
- Applications in POS tagging and named entity recognition.
- RNN language models.
- RNNs with self-attention and applications in text classification.
- Bidirectional and stacked RNNs.
- Obtaining word embeddings from character-based RNNs.
- Hierarchical RNNs for text classification and sequence labeling.
- Sequence-to-sequence RNN models with attention, and applications in machine translation.
- Universal sentence encoders, LASER.
- Variational dropout.
- Pretraining language models, ELMo.

Extracting contract elements

THIS AGREEMENT is made the 15th day of October 2009
(The “Effective Date”) BETWEEN:

- (1) Sugar 13 Inc., a corporation whose office is at James House, 42-50 Bond Street, London, EW2H TL (“Sugar”);
- (2) E2 UK Limited, a limited company whose registered office is at 260 Bathurst Road, Yorkshire, SL3 4SA (“Provider”).

RECITALS:

- A. The Parties wish to enter into a framework agreement which will enable Sugar, from time to time, to [...]
- B. [...]

NO THEREFORE IT IS AGREED AS FOLLOWS:

ARTICLE I - DEFINITIONS

- “Sugar” shall mean: Sugar 13 Inc.
“Provider” shall mean: E2 UK Limited
“1933 Act” shall mean: Securities Act of 1933

ARTICLE II - TERMINATION

The Service Period will be for five (5) years from the Effective Date (The “Initial Term”). The agreement is considered to be terminated in October 16, 2014.

ARTICLE III - PAYMENT - FEES

During the service period monthly payments should occur. The estimated fees for the Initial Term are £154,800.

ARTICLE IV - GOVERNING LAW

This agreement shall be governed and construed in accordance with the Laws of England & Wales. Each party hereby irrevocably submits to the exclusive jurisdiction of the courts sitting in Northern London.

IN WITNESS WHEREOF, the parties have caused their respective duly authorized officers to execute this Agreement.

BY: George Fake
Authorized Officer
Sugar 13 Inc.

BY: Olivier Giroux
CEO
E2 UK LIMITED

Identify start/end dates,
duration, contractors, amount,
legislations refs, jurisdiction
etc. Similar to Named Entity
Recognition (NER).

Reminder: window-based NER

i -th word of the text being classified

3-word **window** (often larger)

yesterday language **tech** announced that...

1-hot vectors ($|V| \times 1$) of the words in the **window**. ($|V|$ is the **vocabulary size**).

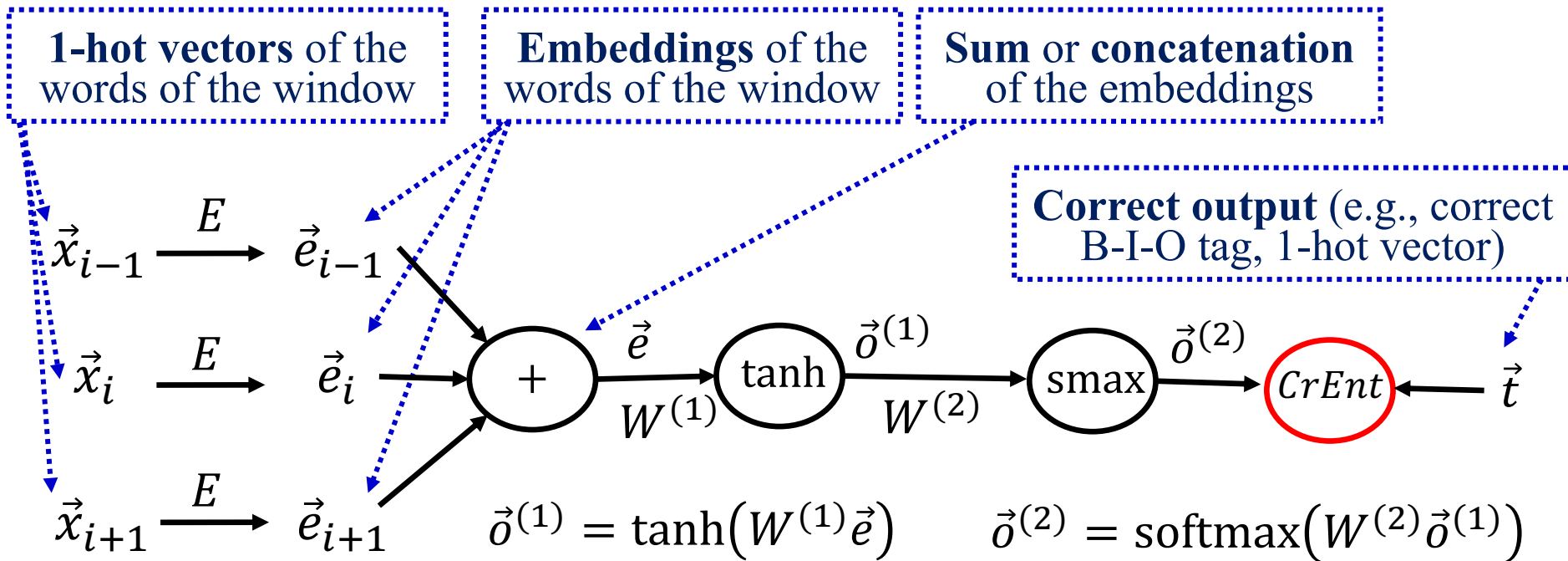
$$\vec{x}_{i-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \vec{x}_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \quad \vec{x}_{i+1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 1 \end{bmatrix}$$

Embeddings ($d \times 1$) of the words in the **window**. (d is the **dimensionality** of the word embeddings).

$$\vec{e}_{i-1} = \begin{bmatrix} 1.8 \\ 2.3 \\ -1.4 \\ 3.7 \\ \dots \\ -1.1 \end{bmatrix} \quad \vec{e}_i = \begin{bmatrix} 2.4 \\ -3 \\ 9.3 \\ 5.1 \\ \dots \\ 3.9 \end{bmatrix} \quad \vec{e}_{i+1} = \begin{bmatrix} 2.2 \\ 3.8 \\ 1.2 \\ -6.4 \\ \dots \\ 7.1 \end{bmatrix}$$

Let E be a matrix ($d \times |V|$) that **contains all the embeddings** of the **vocabulary as columns**. Then:
 $\vec{e}_{i-1} = E\vec{x}_{i-1}$, $\vec{e}_i = E\vec{x}_i$, ...

Reminder: window-based NER



We learn $W^{(1)}$, $W^{(2)}$ with **backpropagation**. We can also learn (or modify) the **word embeddings E** during **backpropagation**! But when we don't have large training datasets (e.g., corpus manually annotated with B-I-O tags), it may be better to use **pre-trained embeddings**, which can be obtained from large non-annotated corpora (e.g., via Word2Vec, GloVe).

We can use the same window-based approach for **POS-tagging**, **chunking**, ...

Reminder: cross-entropy loss

Word being classified.

3-word window (often larger).

yesterday language **tech** announced that...

$$\vec{o} = \begin{bmatrix} P_m(C = c_1) \\ P_m(C = c_2) \\ P_m(C = c_3) \\ \dots \\ P_m(C = c_k) \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.12 \\ 0.08 \\ \dots \\ 0.14 \end{bmatrix}$$

$$\vec{t} = \begin{bmatrix} P(C = c_1) \\ P(C = c_2) \\ P(C = c_3) \\ \dots \\ P(C = c_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

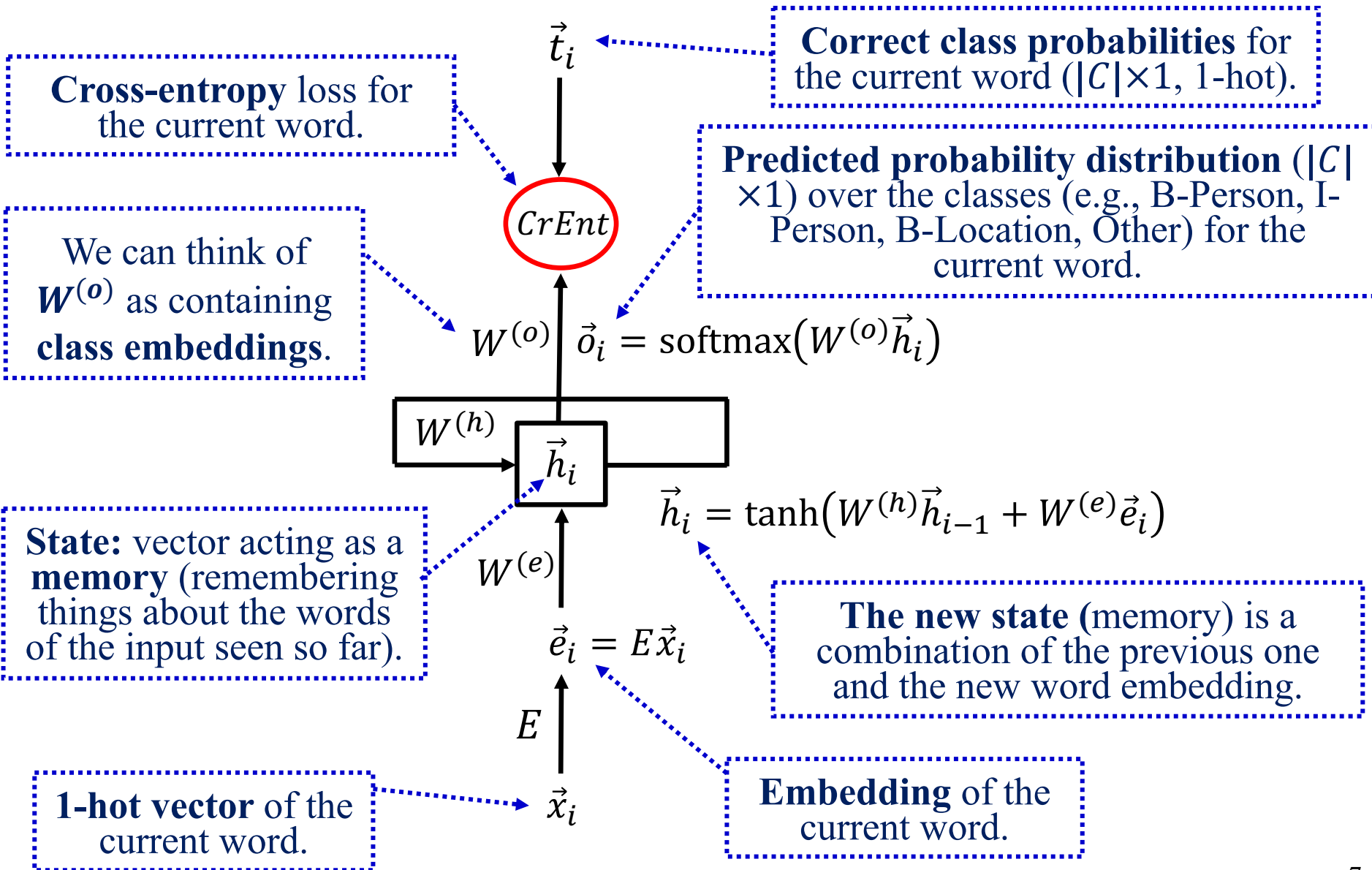
Probability estimates produced by the classifier for the class of the word “tech”.

The correct “probabilities” for the class of “tech”. A 1-hot vector.

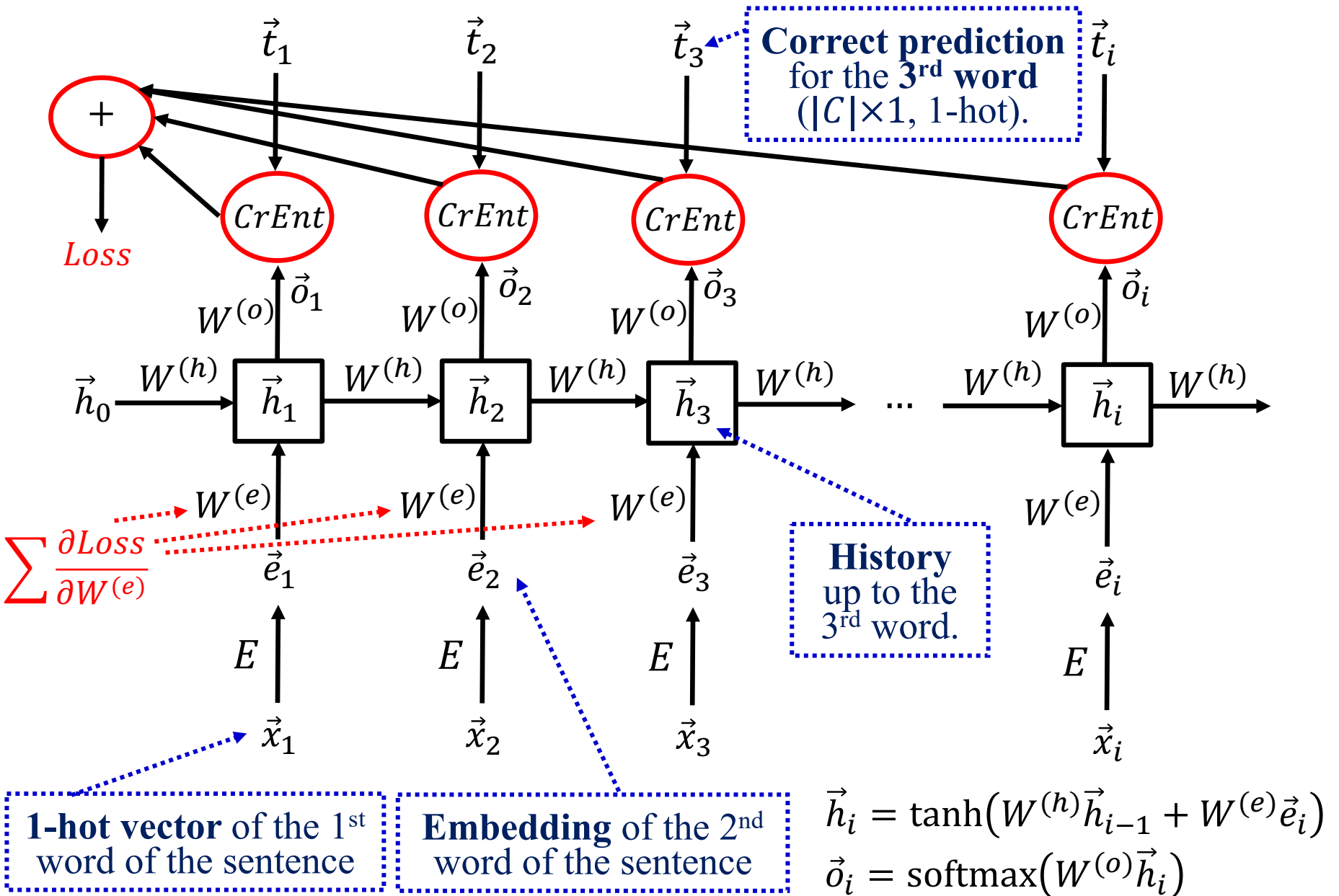
The log-likelihood of the correct class according to the classifier (with a minus sign).

$$H_{P_m}(C) = - \sum_{i=1}^k P(C = c_i) \log_2 P_m(C = c_i) = - \log_2 P_m(C = c_2)$$

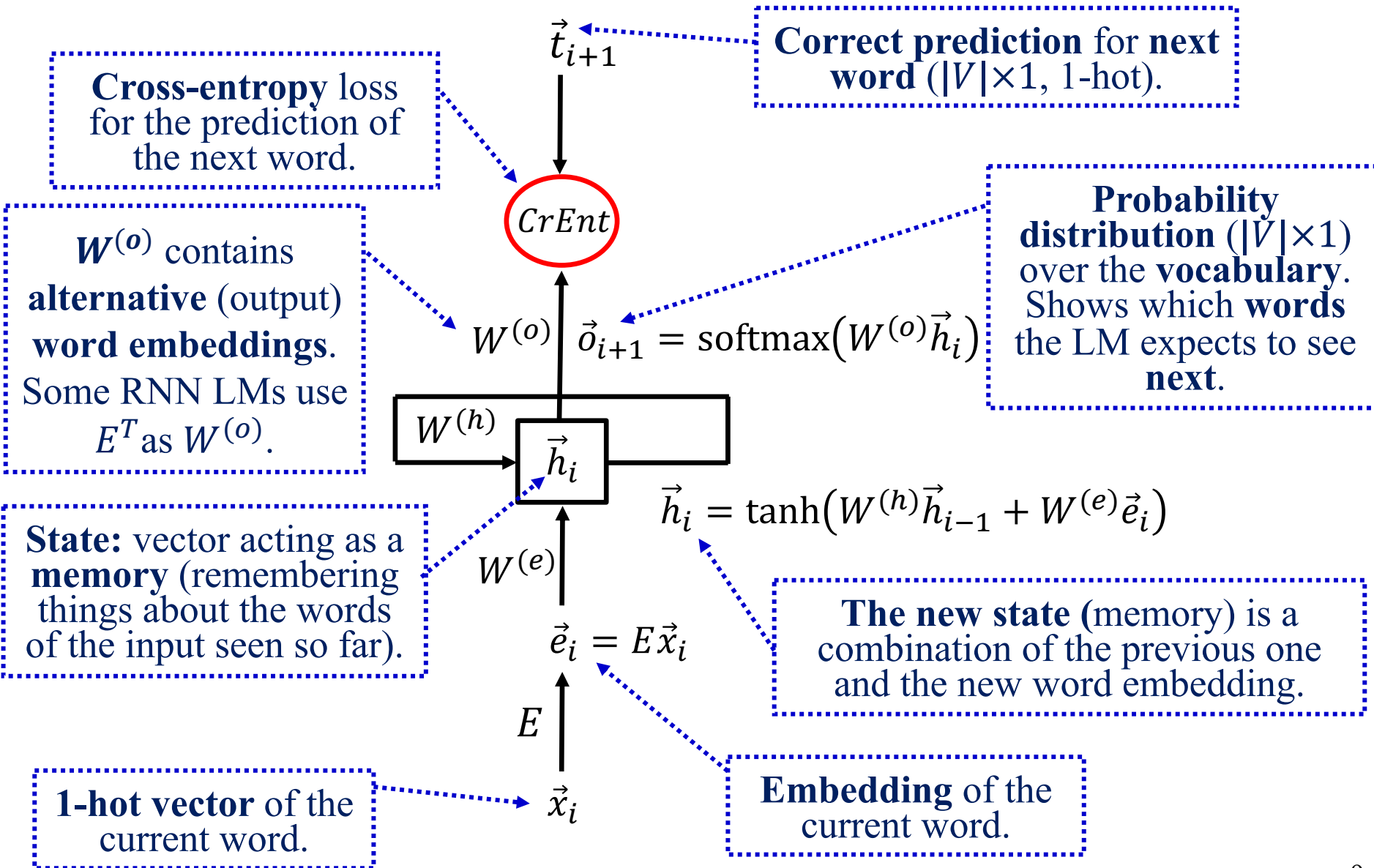
RNN-based Named Entity Recognizer



Unrolled RNN



RNN language model



Reminder: LMs as next word predictors

- **Sequence probability** using a bigram LM:

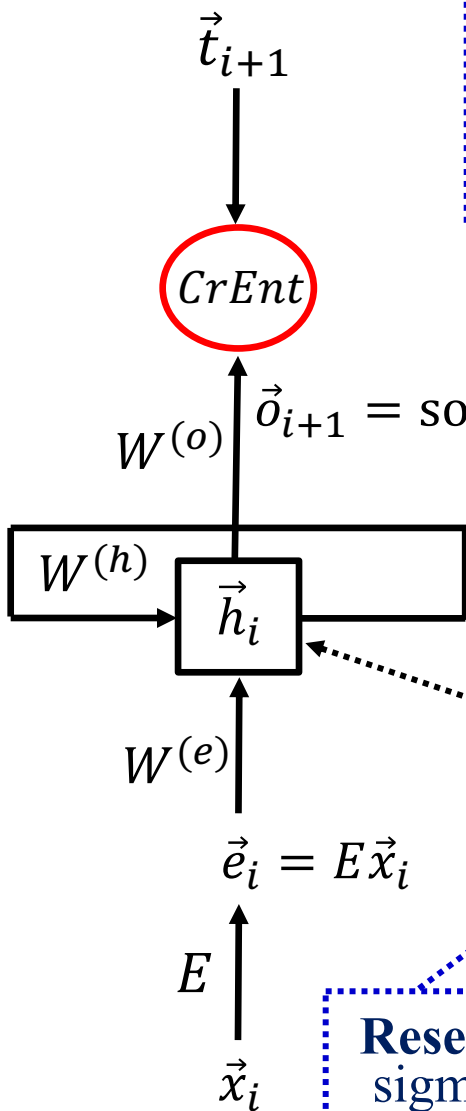
$$P(w_1^k) = P(w_1, \dots, w_k) = P(w_1) \cdot P(w_2 | w_1) \cdot$$

$$P(w_3 | w_1, w_2) \cdot P(w_4 | w_1^3) \cdots P(w_k | w_1^{k-1}) \simeq$$

$$P(w_1 | start) \cdot P(w_2 | w_1) \cdot P(w_3 | w_2) \cdots P(w_k | w_{k-1})$$

- We can think of the **LM** as a system that **provides the probabilities** $P(w_i | w_{i-1})$, which we then multiply.
 - Or the probabilities $P(w_i | w_{i-2}, w_{i-1})$ for a **trigram LM**.
 - Or the probabilities $P(w_i | h_{i-1})$ for an LM that considers all the “**history**” (previous words) h_{i-1} , e.g., in an **RNN LM**.
- An **LM** typically provides a **distribution** $P(w | h)$ showing how probable it is for **every word** $w \in V$ to be the next one.

RNN LM with GRU cells



Candidate new history (\circ denotes pairwise multiplication). For $\vec{r}_i \approx \vec{1}$, same as the \vec{h}_i of a simple RNN cell. For $\vec{r}_i \approx \vec{0}$, **forgets** \vec{h}_{i-1} and considers only the current word embedding.

New history. For $\vec{z}_i \approx \vec{0}$, same as $\tilde{\vec{h}}_i$. For $\vec{z}_i \approx \vec{1}$, ignores $\tilde{\vec{h}}_i$ and **maintains** \vec{h}_{i-1} as \vec{h}_i .

GRU cell:

$$\tilde{\vec{h}}_i = \tanh(\vec{r}_i \circ W^{(h)}\vec{h}_{i-1} + W^{(e)}\vec{e}_i)$$

$$\vec{h}_i = \vec{z}_i \circ \tilde{\vec{h}}_i + (\vec{1} - \vec{z}_i) \circ \vec{h}_{i-1}$$

$$\vec{r}_i = \sigma(W^{(r)}\vec{h}_{i-1} + U^{(r)}\vec{e}_i)$$

$$\vec{z}_i = \sigma(W^{(z)}\vec{h}_{i-1} + U^{(z)}\vec{e}_i)$$

Reset gate (σ is the sigmoid function).

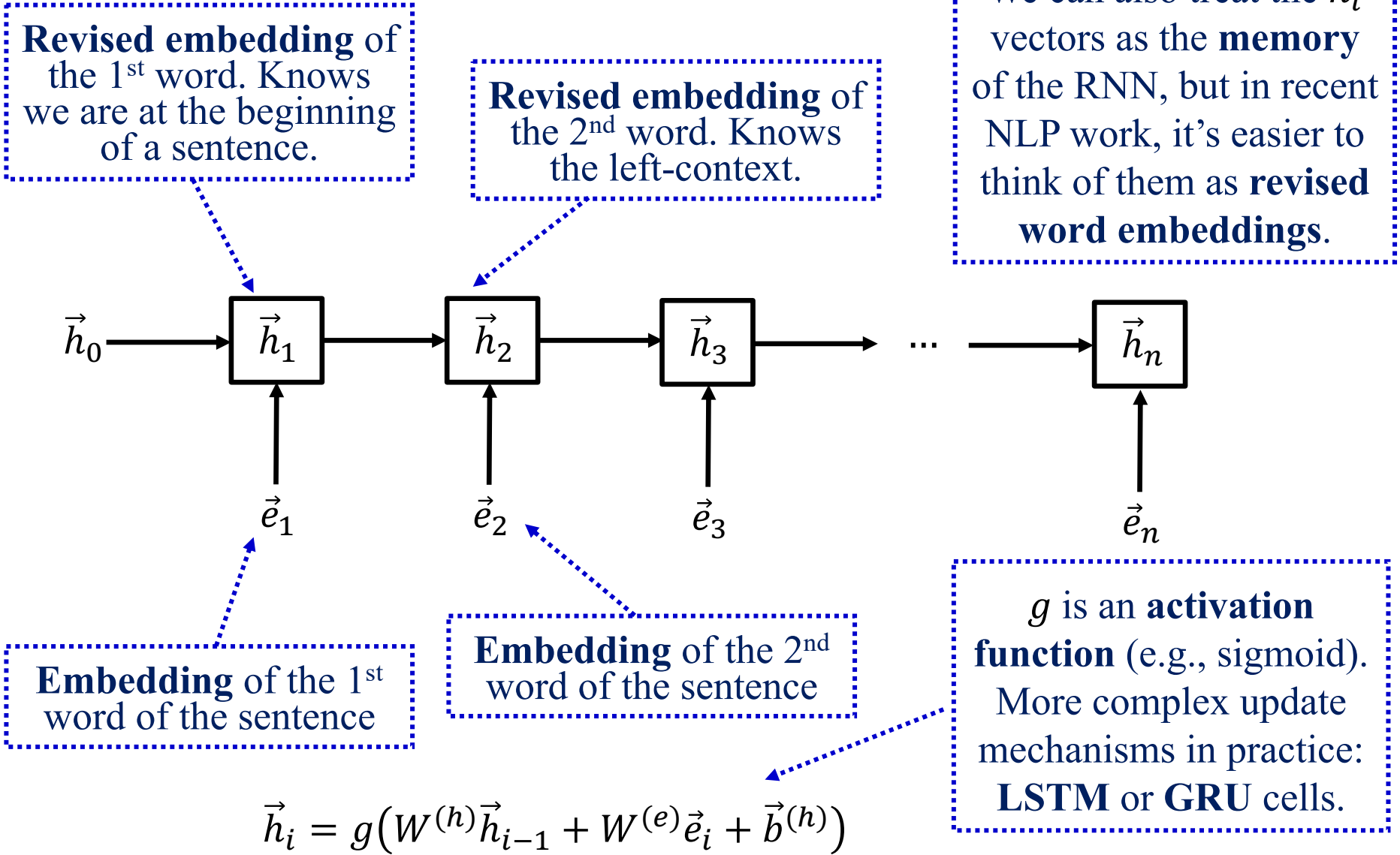
Update gate.

LSTM cells are similar, but with **more gates**. See <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

More about RNNs

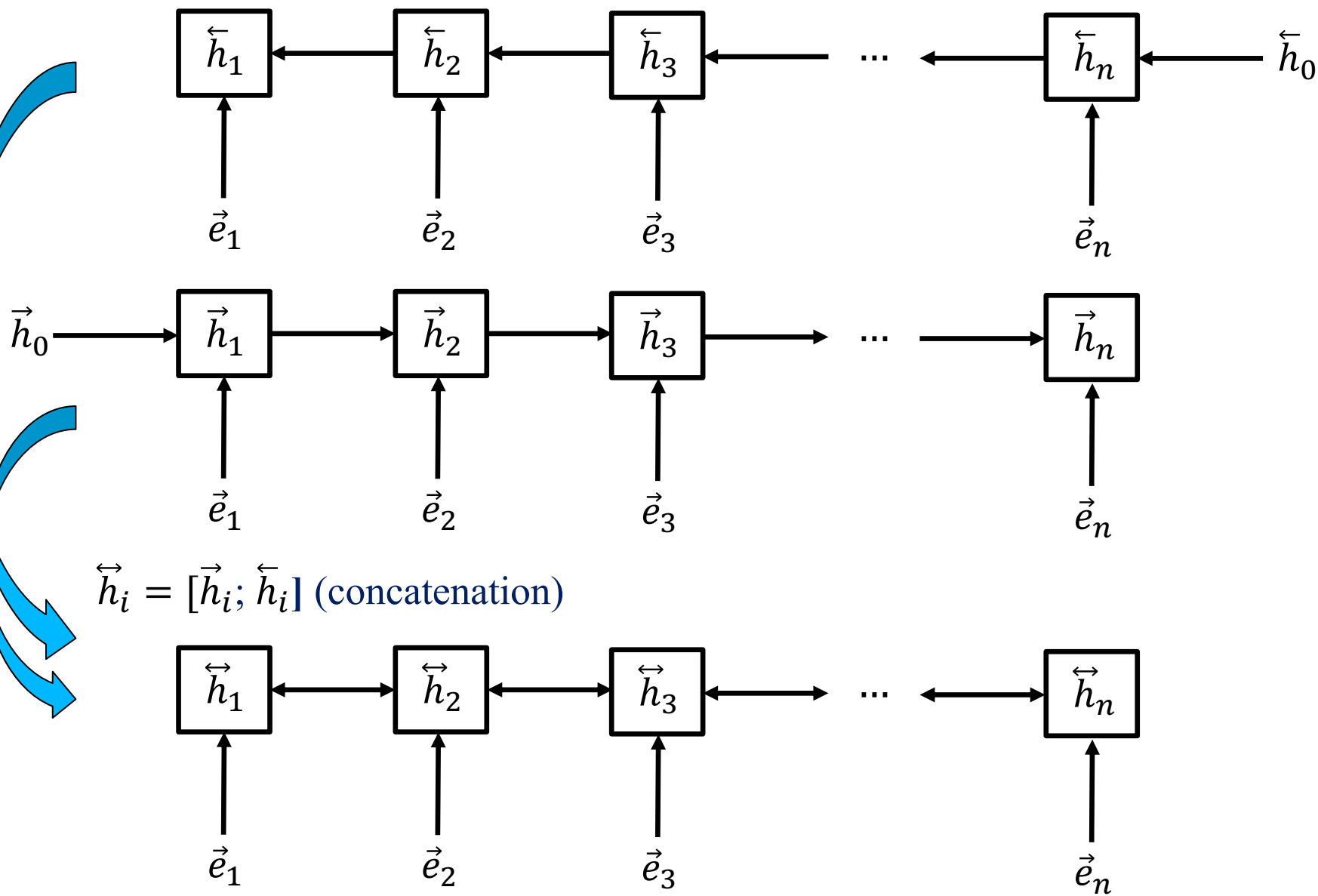
- Trained by **backpropagation** (with **unrolled** view).
 - For **each sentence (or window)**, feed it to the **unrolled RNN**, compute the **loss** and **backpropagate**, adding gradients obtained for the **same matrix** (e.g., same $W^{(h)}$ at each cell).
 - **GRU** or **LSTM** cells help avoid **vanishing gradients**.
 - The norms of the **gradients** can be **clipped** (when larger than a max value) to avoid **exploding gradients**.
 - Use **layer normalization**, not batch normalization in RNNs.
- We can also **learn the word embeddings (E)** with an RNN LM. Billions of **free training examples!**
 - We can **re-use the word embeddings** in **other NLP tasks**.
 - With a **large vocabulary**, **softmax** is too **slow** (alternatives: small vocabulary, hierarchical softmax, negative sampling).

What about the right-context of each token?

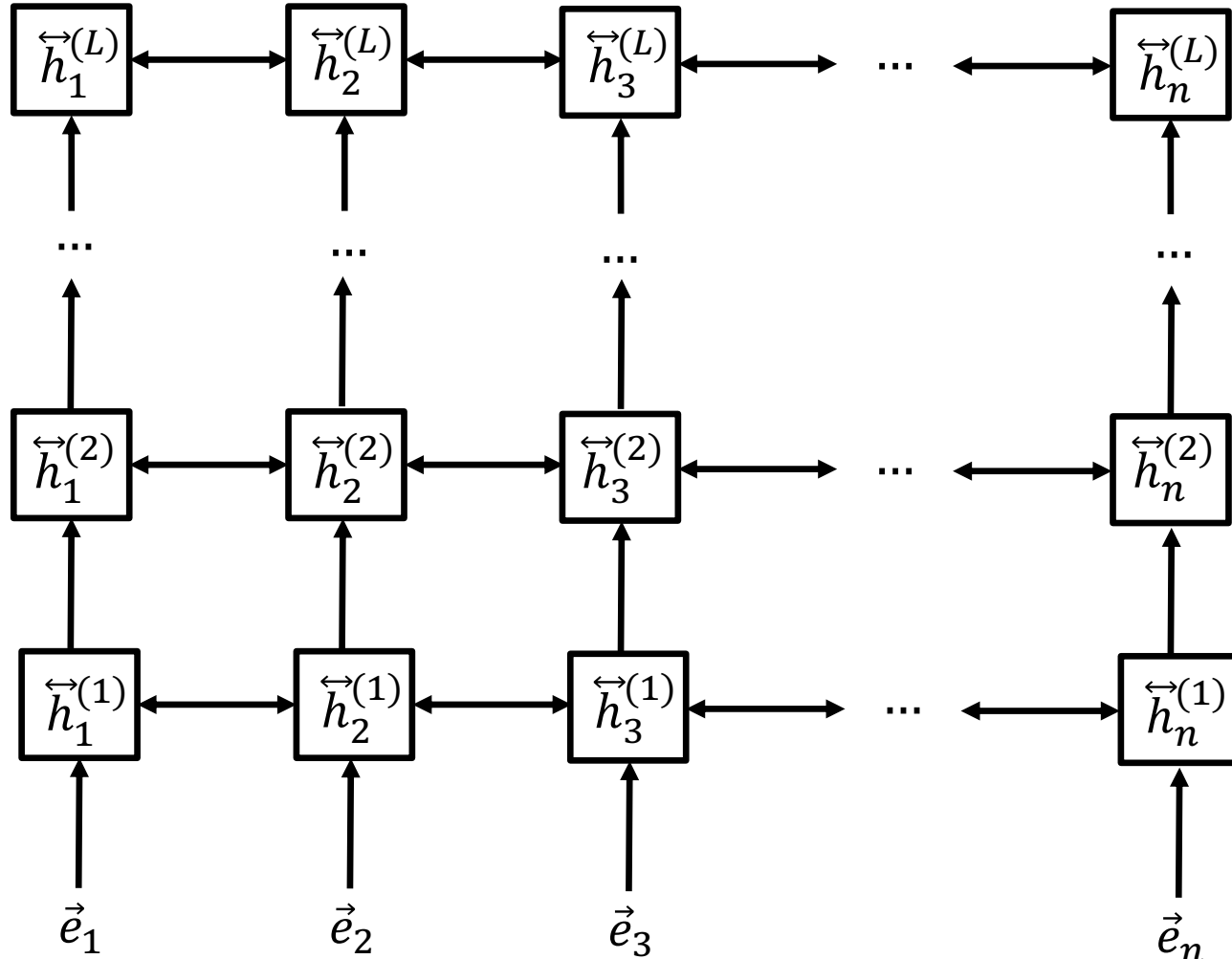


$$\vec{h}_i = g(W^{(h)}\vec{h}_{i-1} + W^{(e)}\vec{e}_i + \vec{b}^{(h)})$$

Bidirectional RNN (biRNN)



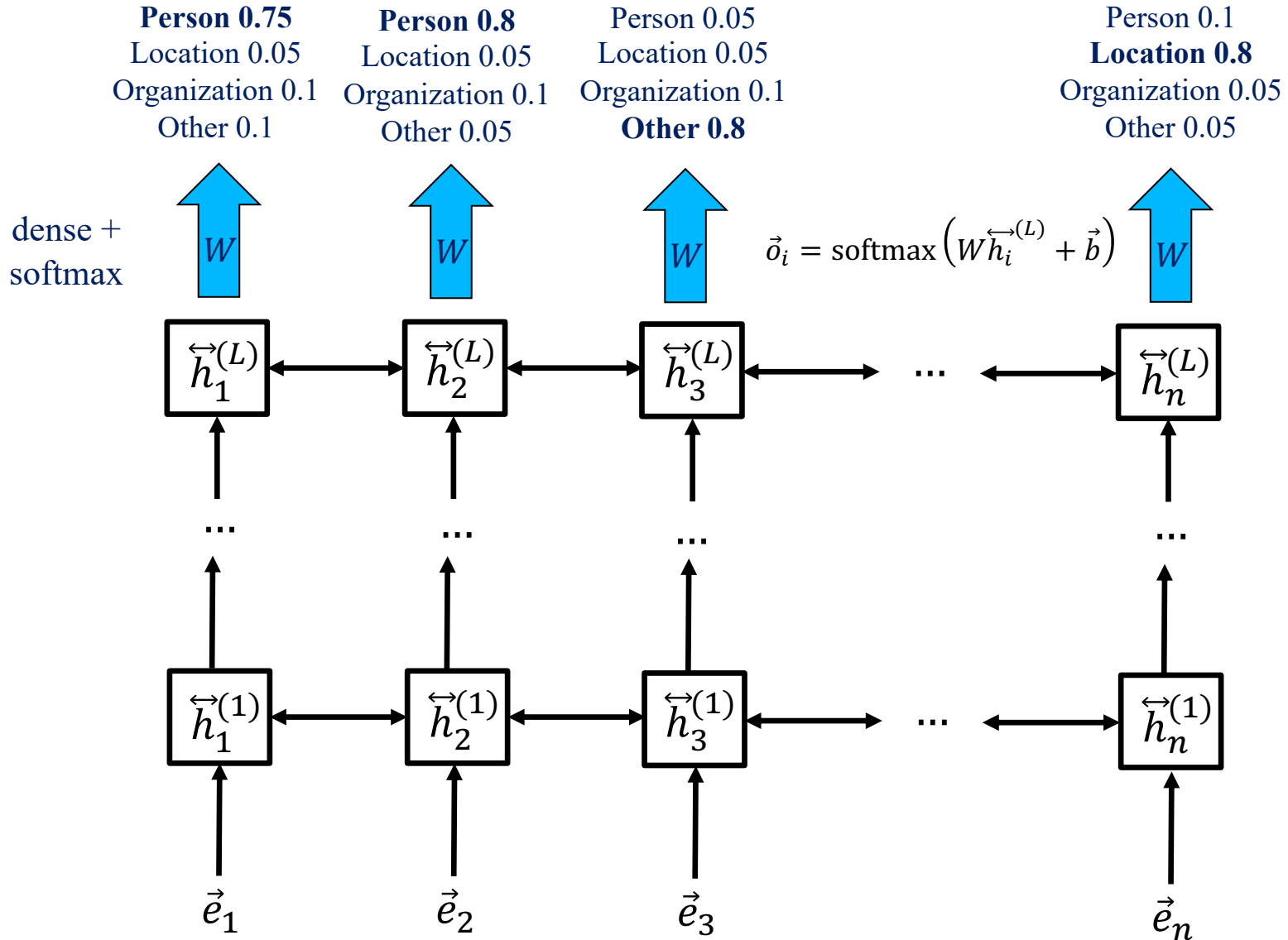
Stacked bidirectional RNN



Each layer revises the word embeddings of the previous (lower) layer. The **embeddings** become **increasingly more context-aware** and also **increasingly more appropriate** for the **particular task** we address...

Token classification with a stacked biRNN

Compare to the correct predictions (sum the cross-entropy loss for all token positions) and backpropagate to adjust all the weights, including the weights of the stacked biRNN.

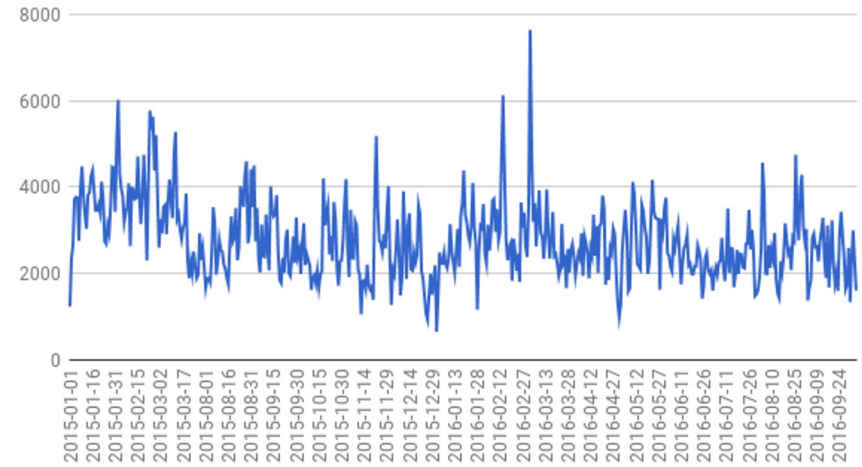


User comment moderation

A moderation panel assists the moderators to detect abusive comments, and leads to quicker publication of non-abusive comments.

Highlighting suspicious words using an RNN with self-attention.

Number of comments per day



Moderation Panel

Go	and	hang	yourself	!	85%						
You	are	ignorant	and	vandal	!	Stop	it	!	88%		
Hello	there	try	to	relax					0%		
Thanks	.	Please	go	f#\$@	yourself	.	Ty	!	85%		

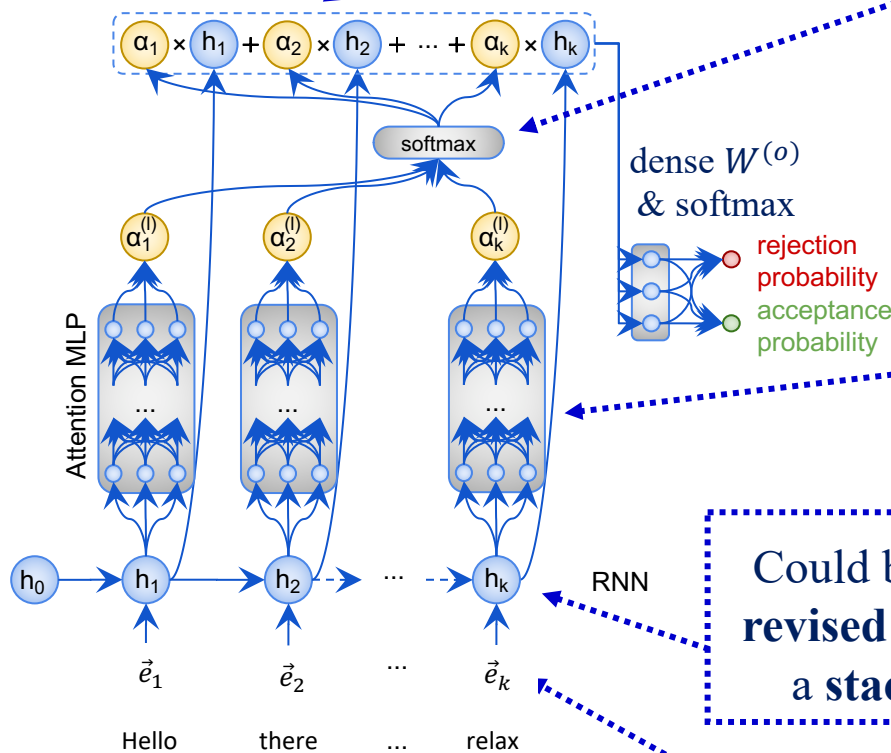
J. Pavlopoulos, P. Malakasiotis and I. Androutsopoulos, “Deeper Attention to Abusive User Content Moderation”, EMNLP 2017, <http://nlp.cs.aueb.gr/pubs/emnlp2017.pdf>.

RNN with deep self-attention

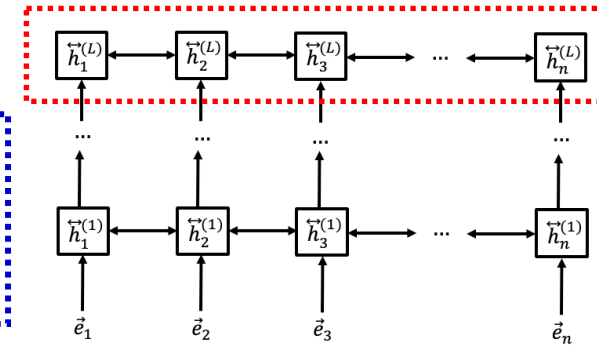
The **entire input text** is now represented by the **weighted (by a_i scores) sum of the revised embeddings** of its words.

The **softmax** ensures all the a_i scores are between 0 and 1, and that they sum to 1.

We use an **MLP** (the same at all time-steps) to obtain an **attention score** (importance) a_i for each word from its revised embedding h_i . We could also use a **single dense layer**: $a_i = W^{(a)} h_i$.



Could be the **top-level revised embeddings of a stacked biRNN**.

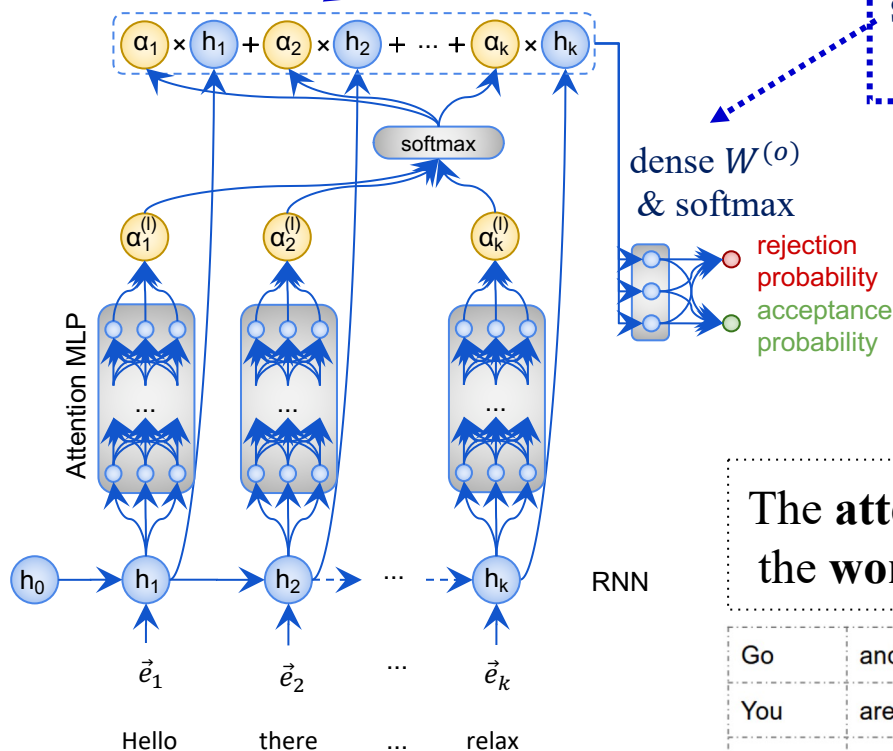


Initial word embeddings (e.g., via Word2Vec).

RNN with deep self-attention

The **entire input text** is now represented by the **weighted (by a_i scores) sum** of the **revised embeddings** of its words.

We pass the **weighted sum vector** (point) through another **dense layer and softmax** to obtain a **probability score** for **each class** (here accept, reject).

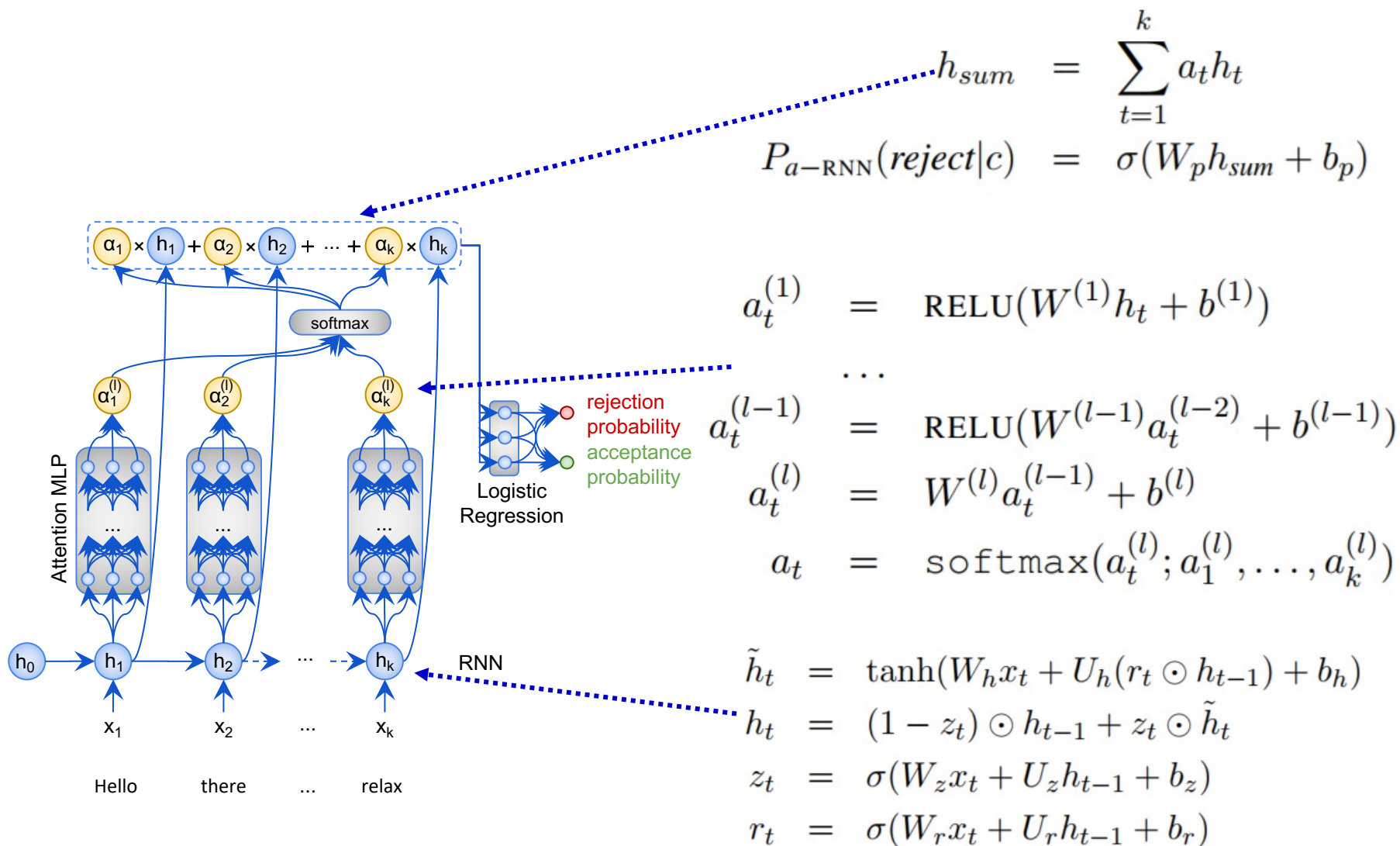


Compare to the correct predictions with a **cross-entropy loss** and **backpropagate** to **adjust the weights** of the **entire neural net**, including the MLP and RNN(s).

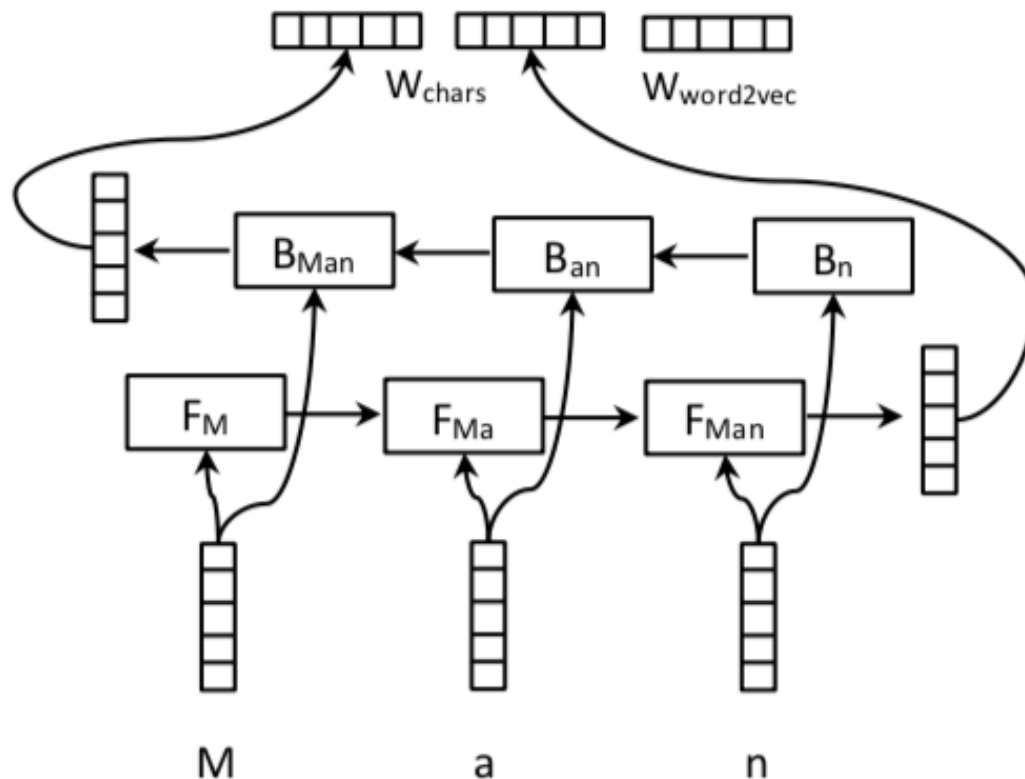
The **attention scores a_i** can also be used to **highlight** the **words** that influence the system's decision most.

Go	and	hang	yourself	!				
You	are	ignorant	and	vandal	!	Stop	it	!
Thanks	.	Please	go	!	yourself	.	ty	!

RNN with deep self-attention



RNNs that produce word embeddings from character embeddings



Word embedding layer,
part of a larger network.

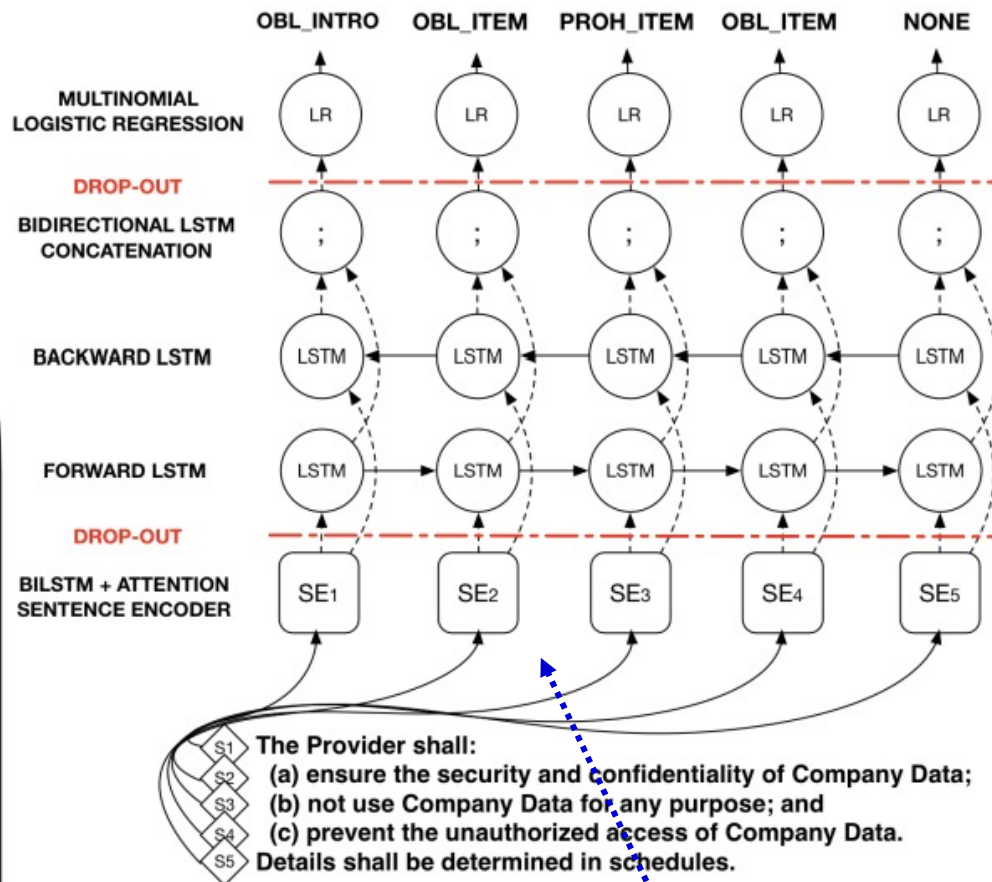
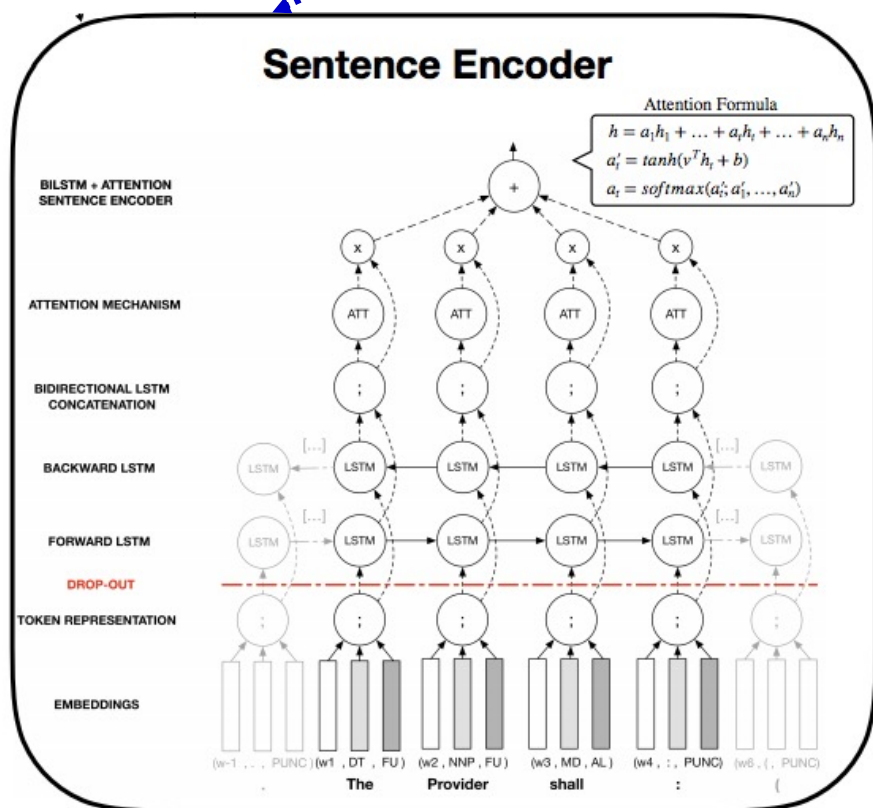
We **concatenate** the
word embedding we get
from the **character-level**
biLSTM with the
Word2Vec embedding.

The **character**
embeddings are **learned**
during back-propagation.

G, Bekoulis, J, Deleu, T, Demeester, C. Develder, “Joint entity recognition and relation extraction as a multi-head selection problem”, Expert Systems with Applications, Vol, 114, pp. 34-45, 2018. Figure from the pre-print <https://arxiv.org/abs/1804.07847>.

Sequence labeling with a Hierarchical RNN

The **lower RNN** reads the words of each sentence and converts the sentence to a **sentence embedding**.



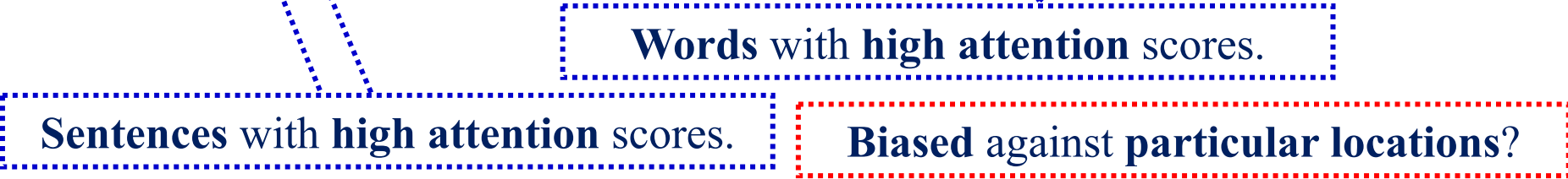
The **upper RNN** reads a sequence of sentence embeddings and **classifies each sentence**.

Legal judgment prediction for ECHR cases

Case ID: 001-148227 Violated Articles: Article 3 Predicted Violation: YES (0.97%)

- 1. The applicant was born in 1955 and lives in Kharkiv .
- 2. On 5 May 2004 the applicant was arrested by four police officers on suspicion of bribe - taking . The police officers took him to the Kharkiv Dzerzhynskyy District Police Station , where he was held overnight . According to the applicant , the police officers beat him for several hours , forcing him to confess .
- 3. On 6 May 2004 the applicant was taken to the Kharkiv City Prosecutor 's Office . He complained of ill-treatment to a senior prosecutor from the above office . The prosecutor referred the applicant for a forensic medical examination .
- 4. On 7 May 2004 the applicant was diagnosed with concussion and admitted to hospital .
- 5. On 8 May 2004 the applicant underwent a forensic medical examination , which established that he had numerous bruises on his face , chest , legs and arms , as well as a damaged tooth .
- 6. On 11 May 2004 criminal proceedings were instituted against the applicant on charges of bribe-taking . They were eventually terminated on 27 April 2007 for lack of corpus delicti .
- 7. On 2 June 2004 the applicant lodged another complaint of ill - treatment with the Kharkiv City Prosecutor 's Office .

Figure 1: Attention over words (colored words) and facts (vertical heat bars) as produced by HAN.



RNNs for Machine Translation

From the slides of R. Socher's course "Deep Learning for NLP", 2015. <http://cs224d.stanford.edu/>

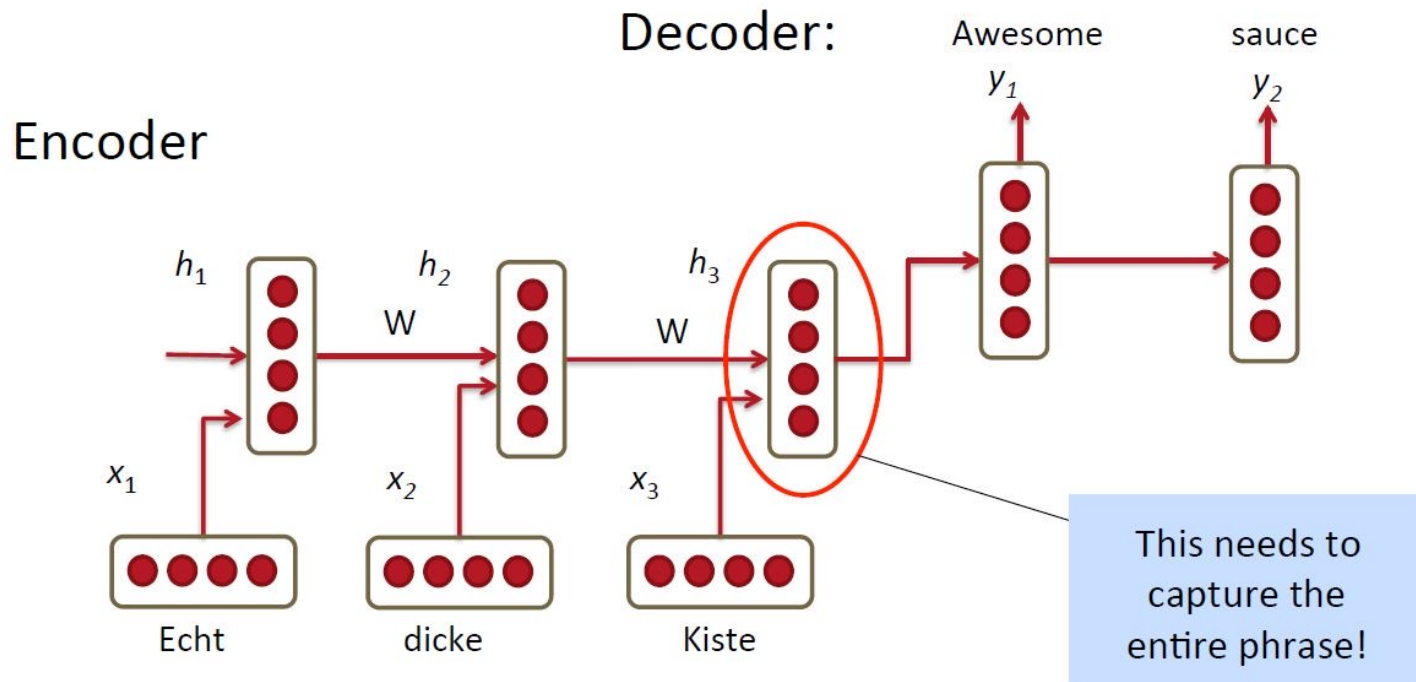
Encoder: $h_t = \phi(h_{t-1}, x_t) = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$

Decoder: $h_t = \phi(h_{t-1}) = f(W^{(hh)}h_{t-1})$

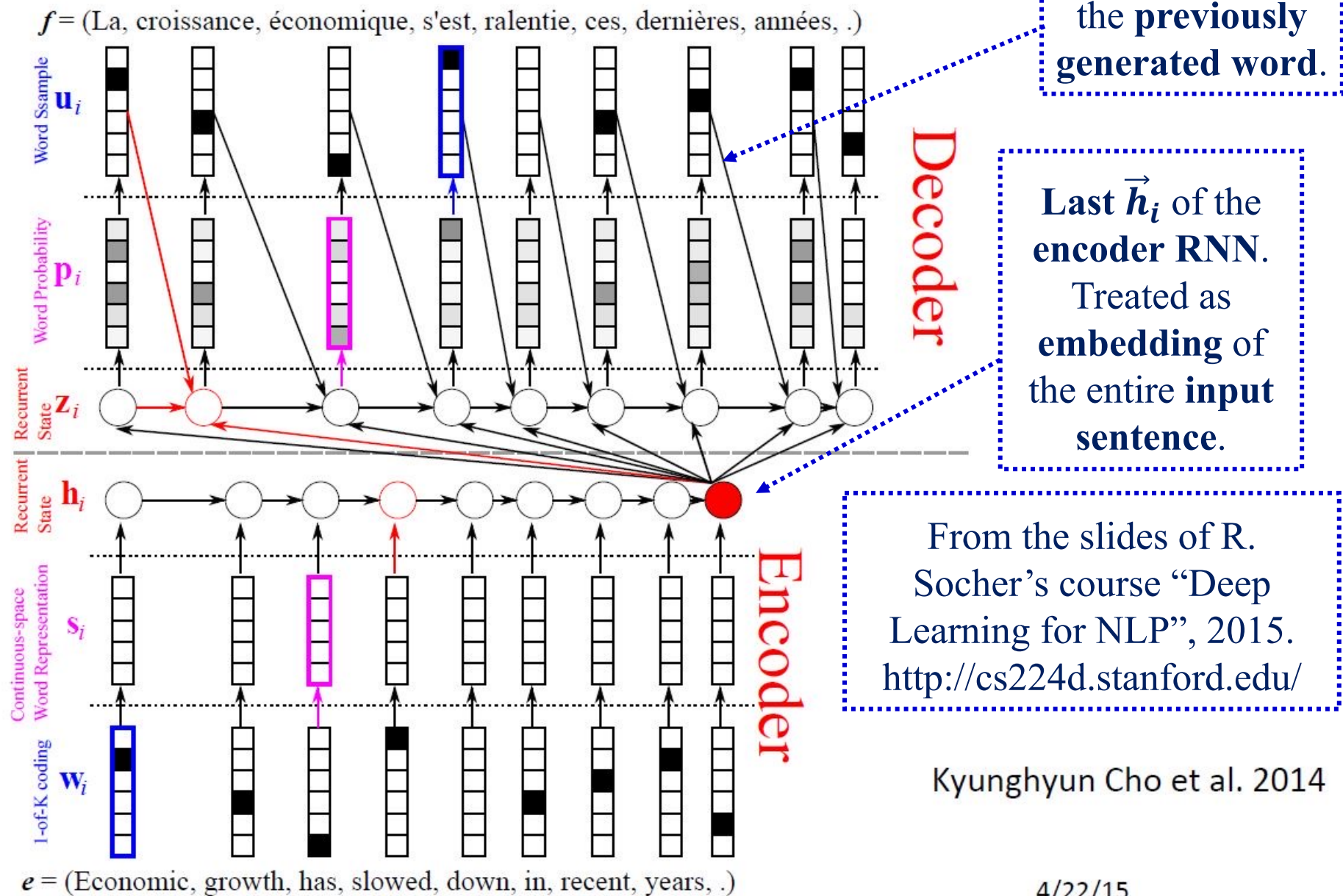
$$y_t = \text{softmax}(W^{(S)}h_t)$$

Minimize cross entropy error for all target words conditioned on source words

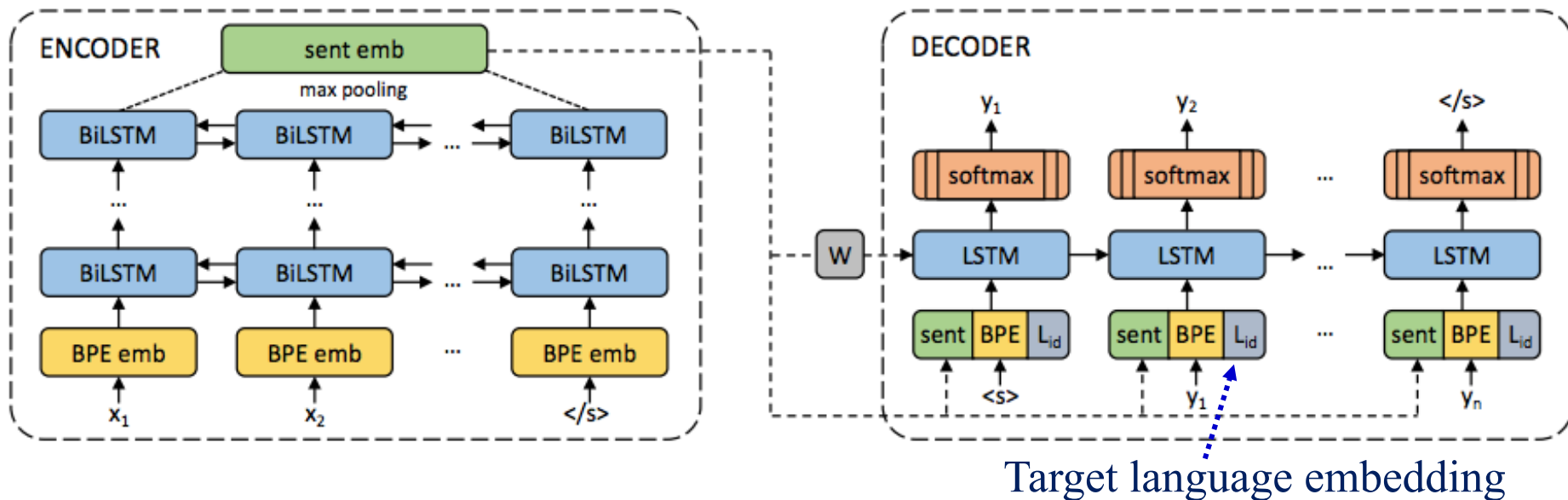
$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log p_{\theta}(y^{(n)} | x^{(n)})$$



Different picture, same idea



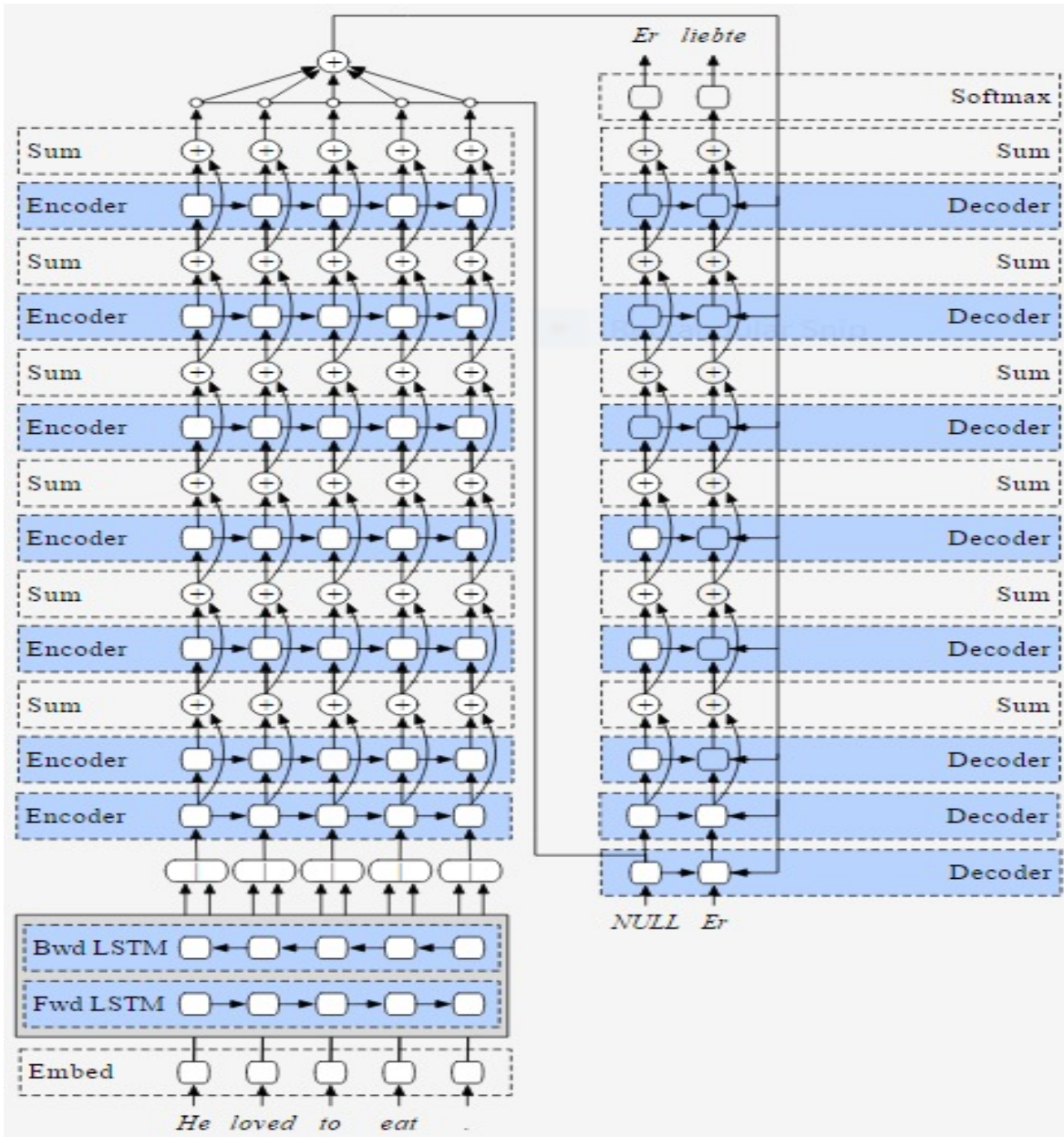
Universal sentence encoders



- **Laser:** Trained on parallel corpora of 93 languages.
 - Using the **same encoder** and **decoder** for all languages.
 - **Shared vocabulary** of sub-word units (BPEs).
- E.g., we can **train a classifier on English tweets**, and use the **same trained classifier on Greek tweets**.
 - In **both languages**, we use the **same encoder** to convert each **tweet to a feature vector**.

M. Artetxe and H. Schwenk, “Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond”. <https://arxiv.org/abs/1812.10464>
<https://code.fb.com/ai-research/laser-multilingual-sentence-embeddings/>

RNN-based Machine Translation

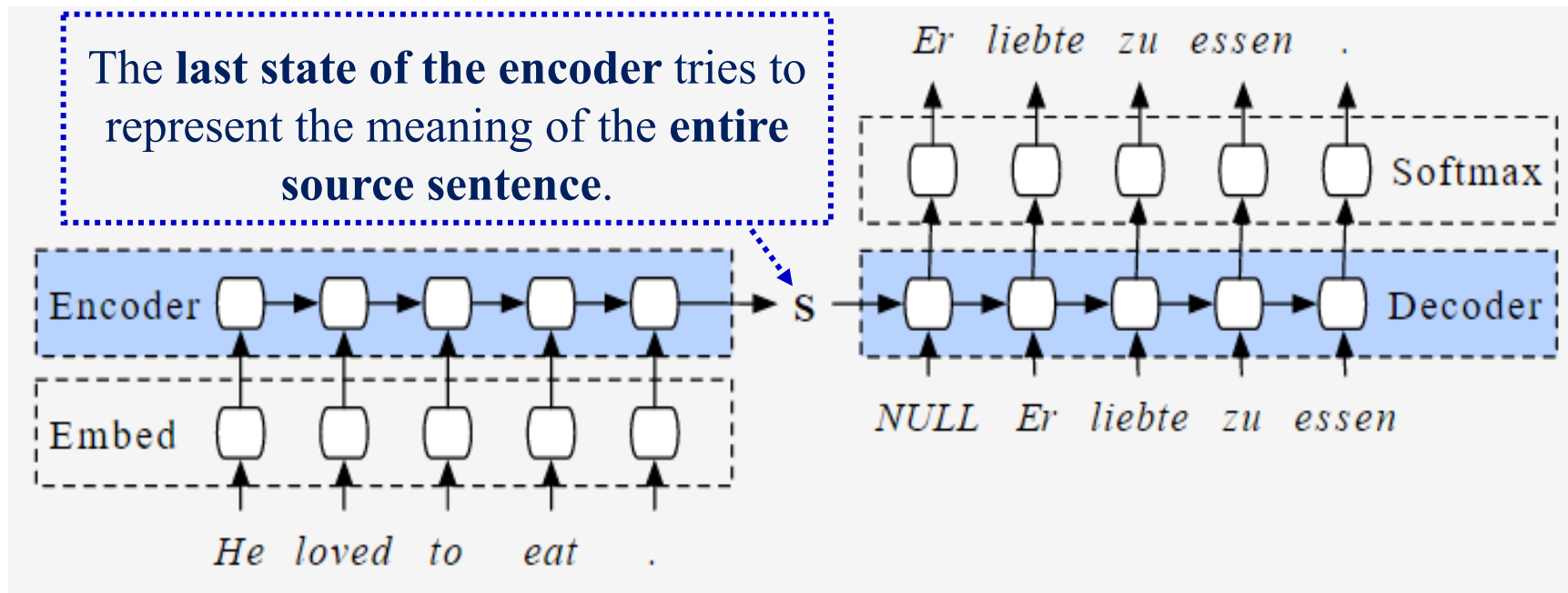


Google's paper:
<https://arxiv.org/abs/1609.08144>

Images from Stephen Merity's
http://smerity.com/articles/2016/google_nmt_arch.html

Easier to explain step by step...

Basic Encoder-Decoder NMT



During training, at each time-step of the **decoder**, we can use the **correct previous word** of the human translation (**teacher forcing**), or we can **randomly use the correct or the predicted** previous word (**scheduled sampling**).

During testing (inference), we always use the **predicted previous word**; and we either **greedily select the most probable next word**, or we use **beam search** to find the translation y_1^m of x_1^n with the highest probability:

$$p(y_1|x_1^n) p(y_2|y_1, x_1^n) p(y_3|y_1^2, x_1^n) \dots p(y_m|y_1^{m-1}, x_1^n)$$

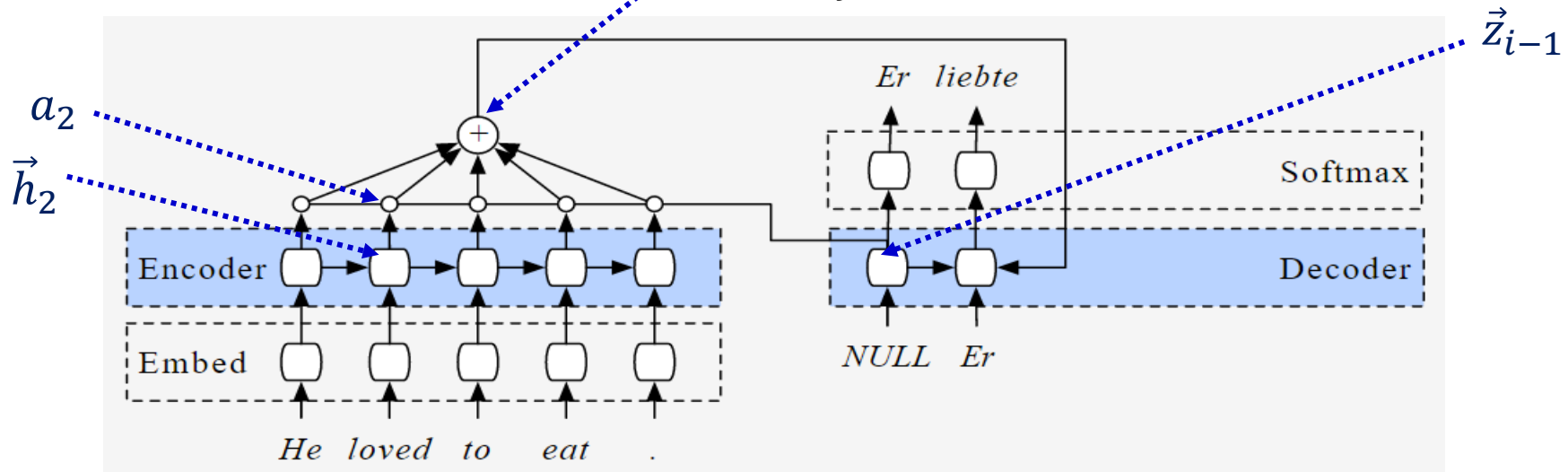
Google's paper: <https://arxiv.org/abs/1609.08144>

Images from Stephen Merity's http://smerity.com/articles/2016/google_nmt_arch.html

Encoder-Decoder with attention

The **source sentence** is now represented by the **weighted sum** of the **encoder states**:

$$\vec{h}_{sum} = \sum_j a_j \vec{h}_j$$



For each German word, the **attention scores** over the English words **change!** Each “**attention**” weight a_j is a **function** of the **corresponding encoder state** \vec{h}_j and the **previous state** \vec{z}_{i-1} of the **decoder** (memory of translation so far), e.g.:

$$\tilde{a}_j = v^T \cdot f(W^{(h)} \vec{h}_j + W^{(z)} \vec{z}_{i-1}) = v^T \cdot f(W[\vec{h}_j; \vec{z}_{i-1}]), \quad a_j = \text{softmax}(\tilde{a}_j)$$

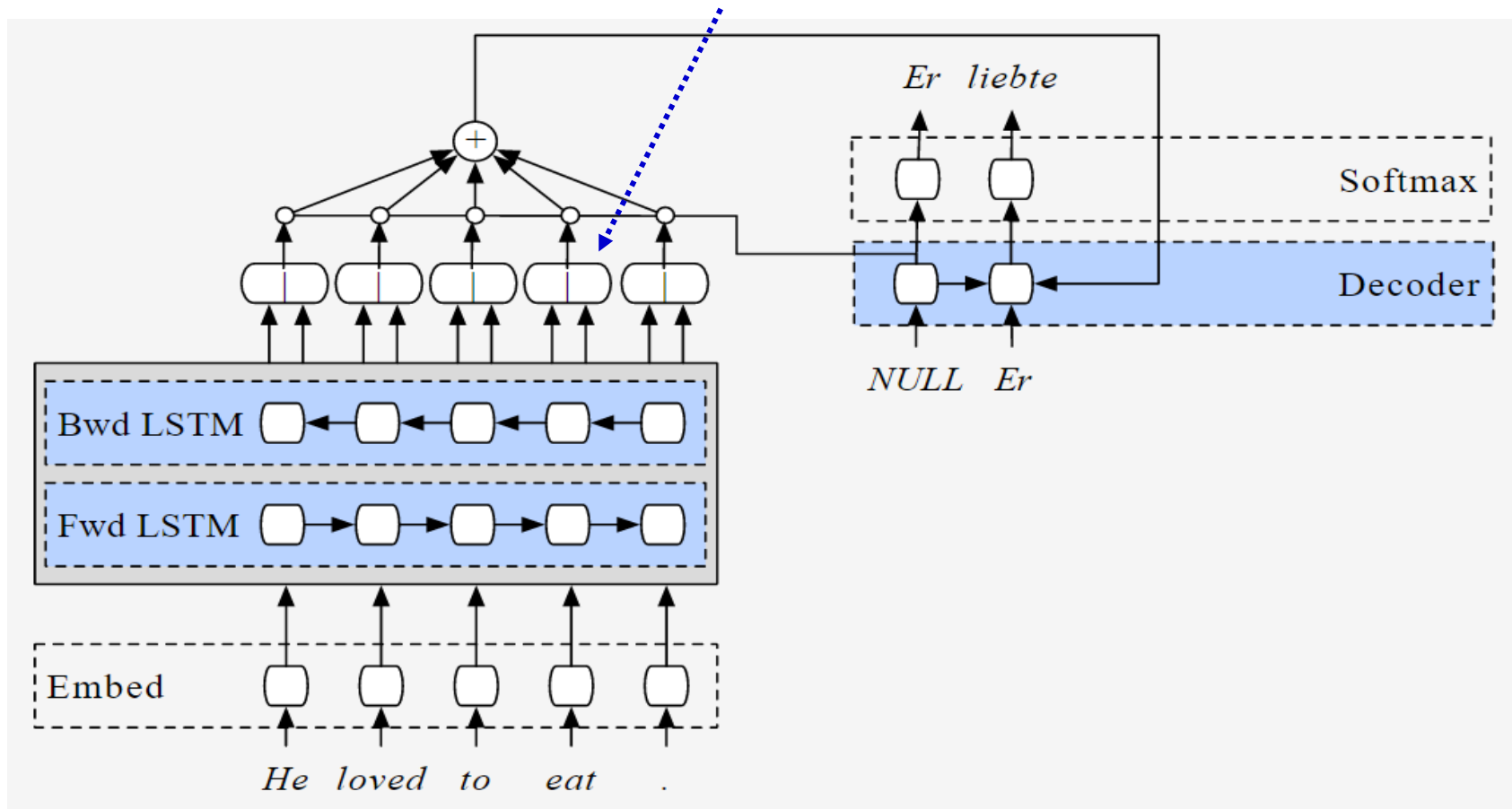
with a **softmax** to make the a_j weights sum to 1.

Google’s paper: <https://arxiv.org/abs/1609.08144>

Images from Stephen Merity’s http://smerity.com/articles/2016/google_nmt_arch.html

Bidirectional LSTM encoder

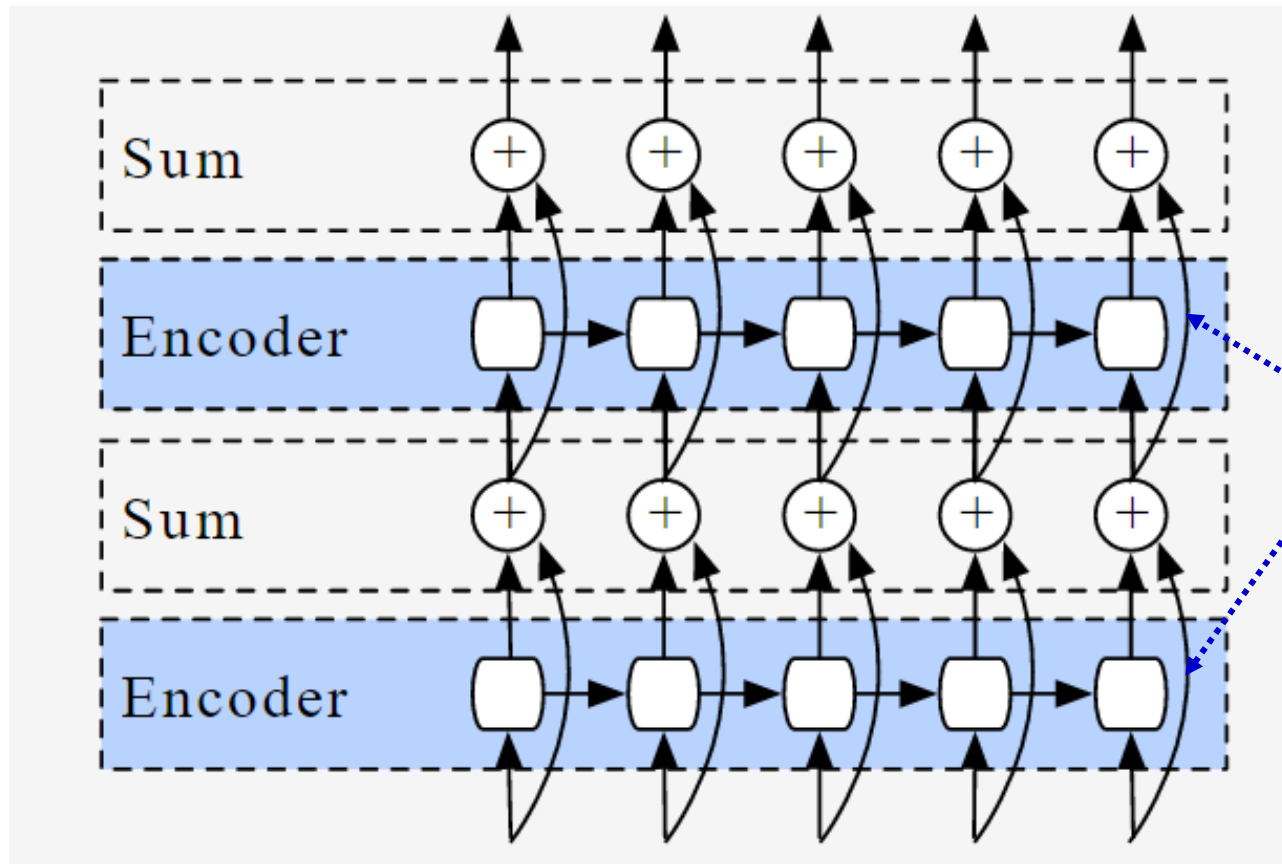
The encoder is now a **bidirectional LSTM**. The **encoder state** for the **j -th word of the source sentence** is the **concatenation** of the **corresponding states of the forward and backward LSTM**.



Google's paper: <https://arxiv.org/abs/1609.08144>

Images from Stephen Merity's http://smerity.com/articles/2016/google_nmt_arch.html

Stacking RNNs and residuals

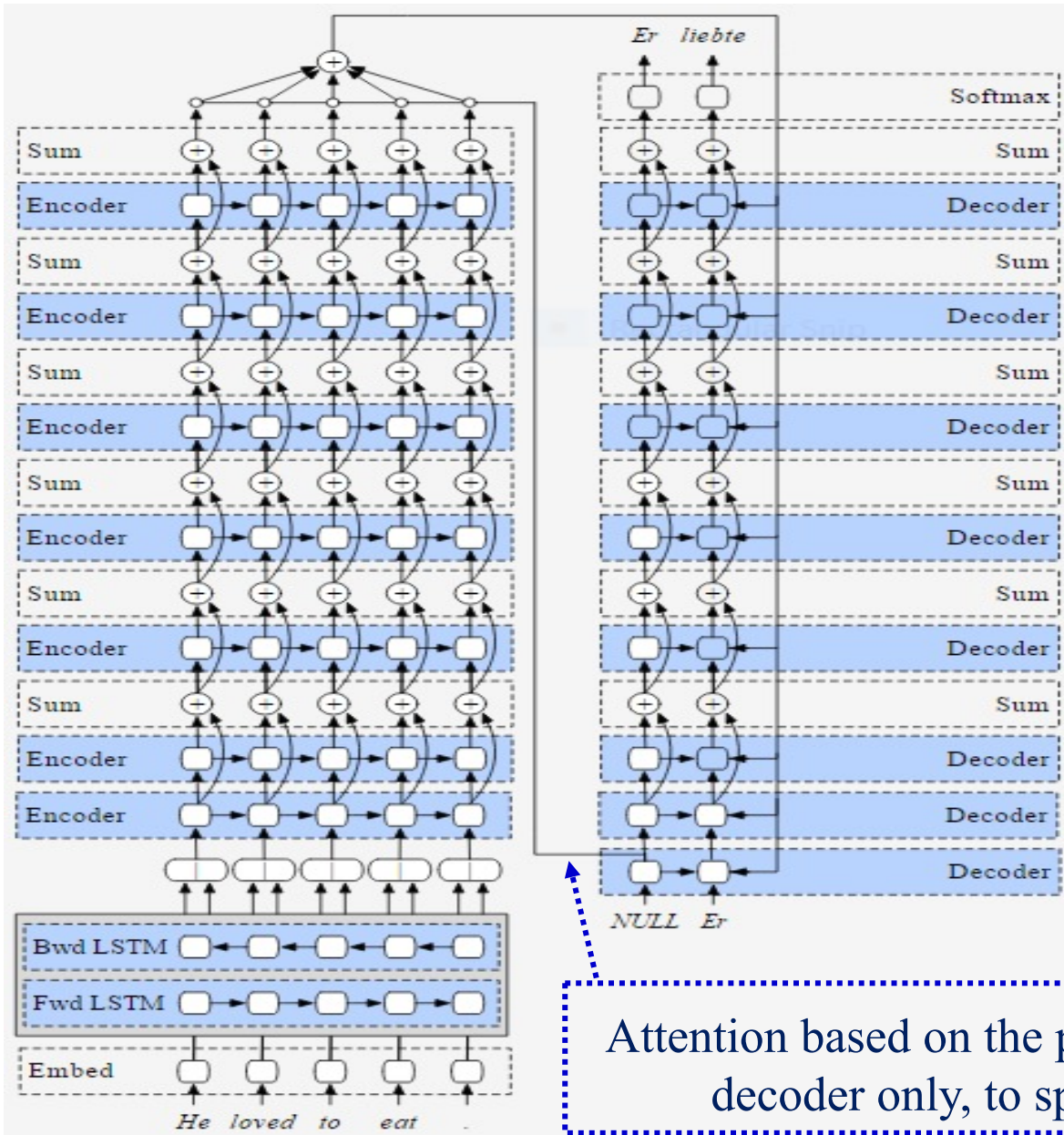


“Residual”
connections allow
skipping layers.
Helps fight
vanishing gradients
in backpropagation.

Google’s paper: <https://arxiv.org/abs/1609.08144>

Images from Stephen Merity’s http://smerity.com/articles/2016/google_nmt_arch.html

RNN-based Machine Translation



Google's paper:
<https://arxiv.org/abs/1609.08144>

Images from Stephen Merity's
http://smerity.com/articles/2016/google_nmt_arch.html

Attention based on the previous state of the bottom decoder only, to speed up computations.

Additional optional reading slides.

Dropout vs. Variational Dropout

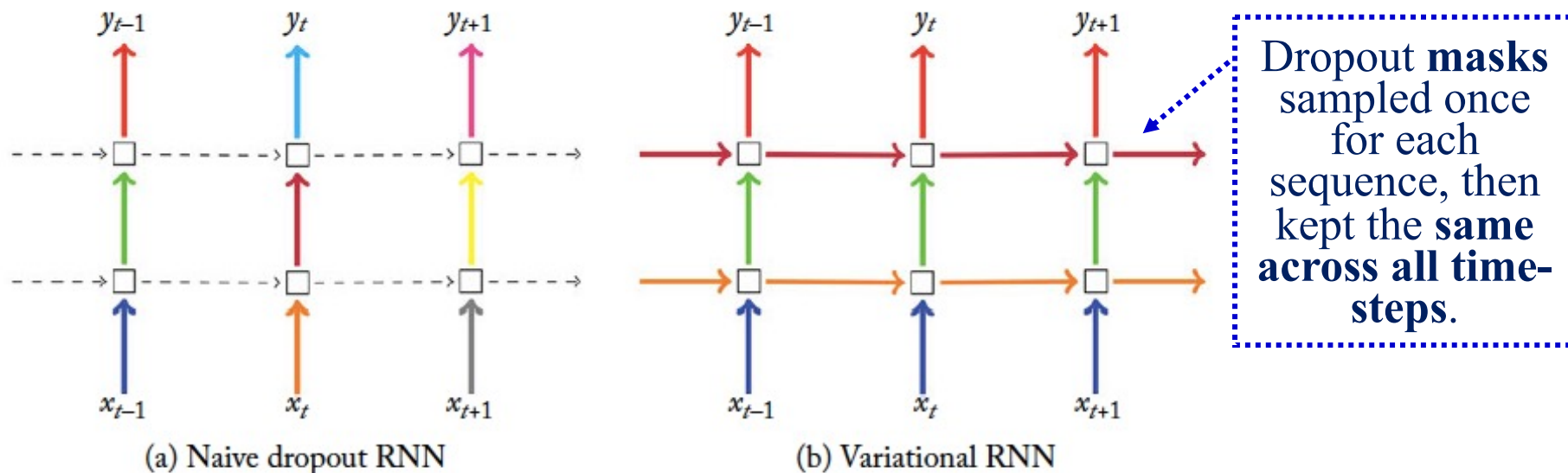
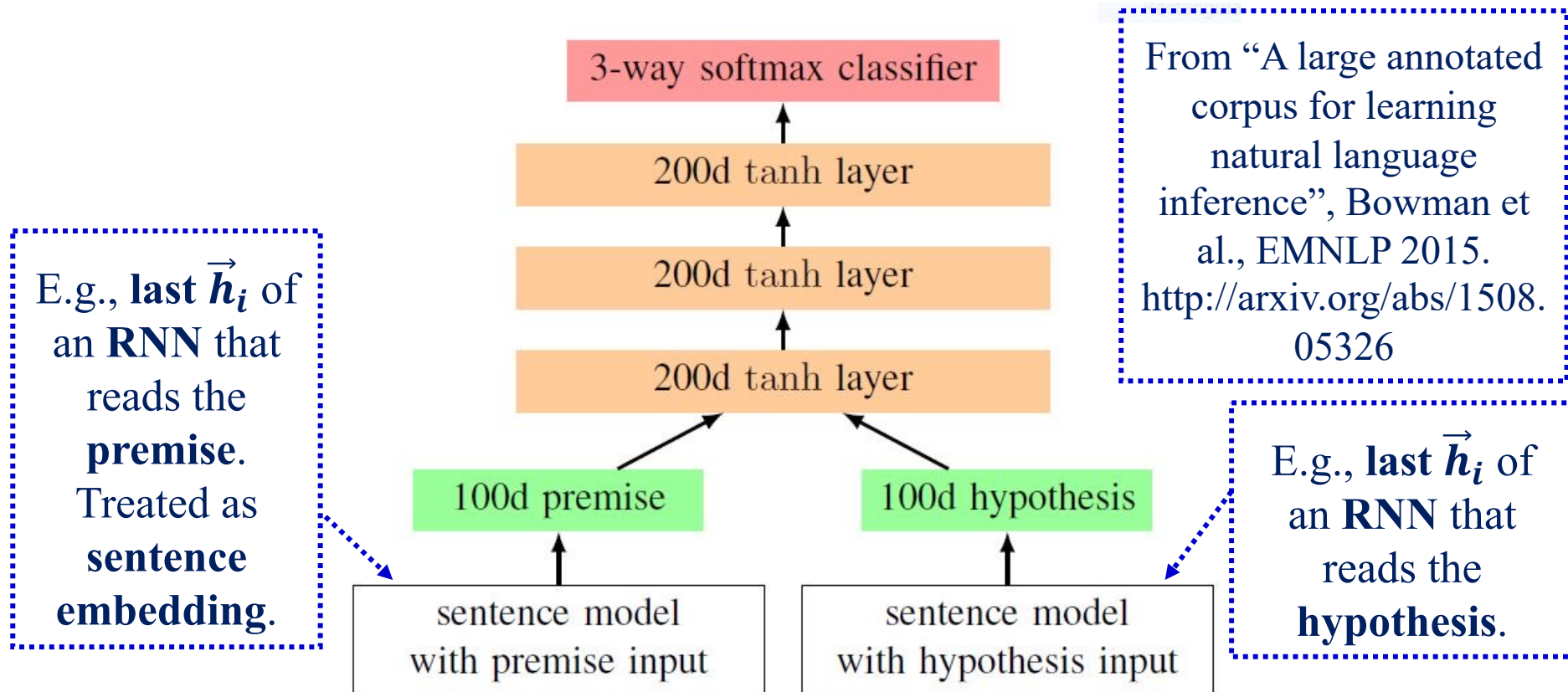


Figure 15.2: Gal's proposal for RNN dropout (b), vs. the previous suggestion by Pham et al. [2013], Zaremba et al. [2014] (a). Figure from Gal [2015], used with permission. Each square represents an RNN unit, with horizontal arrows representing time dependence (recurrent connections). Vertical arrows represent the input and output to each RNN unit. Colored connections represent dropped-out inputs, with different colors corresponding to different dropout masks. Dashed lines correspond to standard connections with no dropout. Previous techniques (naive dropout, left) use different masks at different time steps, with no dropout on the recurrent layers. Gal's proposed technique (Variational RNN, right) uses the same dropout mask at each time step, including the recurrent layers.

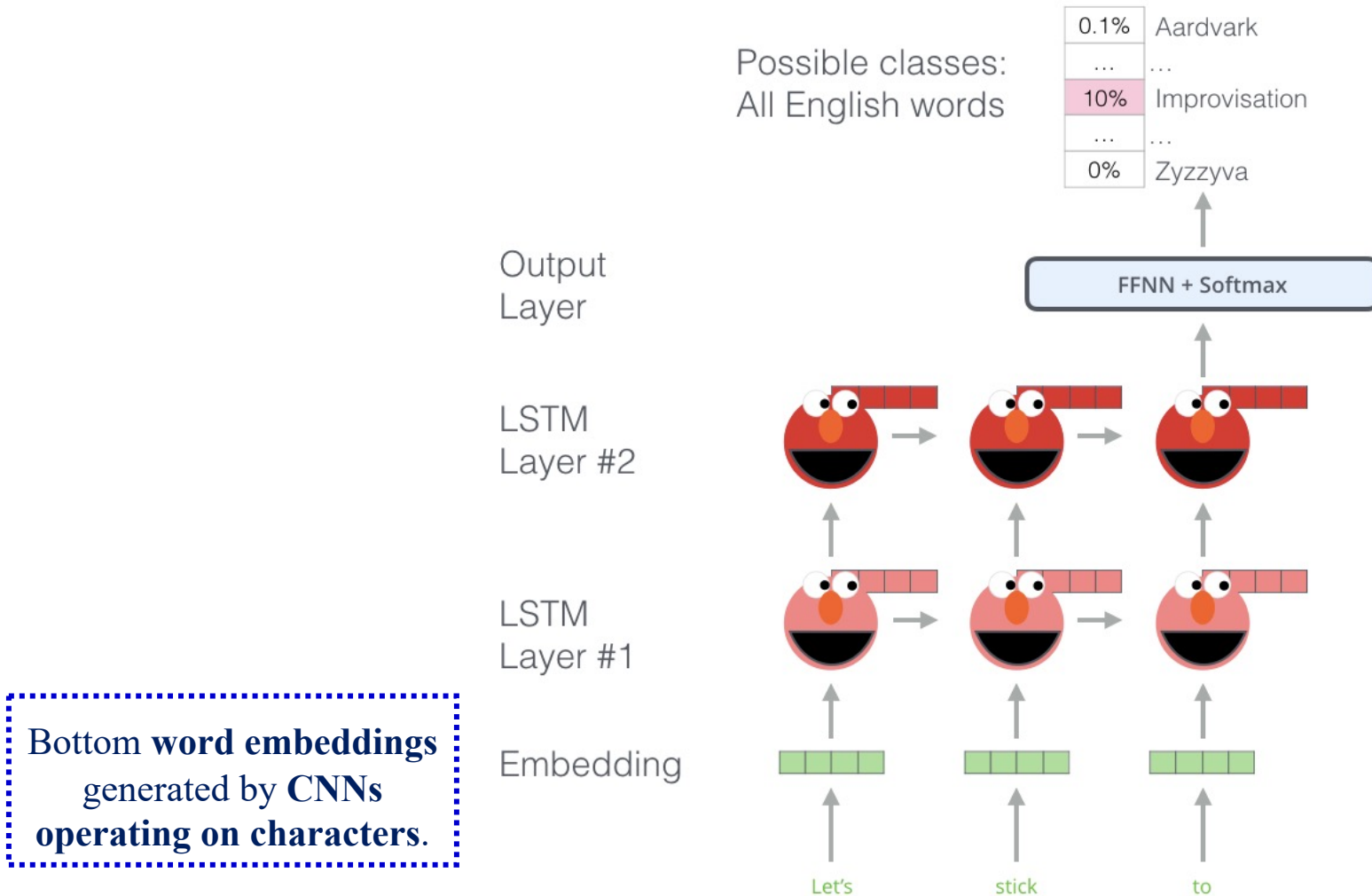
Figure from: Y. Goldberg, *Neural Network Models for Natural Language Processing*, Morgan & Claypool Publishers, 2017. See also <https://adriangcoder.medium.com/a-review-of-dropout-as-applied-to-rnns-72e79ecd5b7b>

Textual entailment with RNNs+MLP



A man inspects the uniform of a figure in some East Asian country.	contradiction C C C C C	The man is sleeping
An older and younger man smiling.	neutral N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	contradiction C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	entailment E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	neutral N N E C N	A happy woman in a fairy costume holds an umbrella.

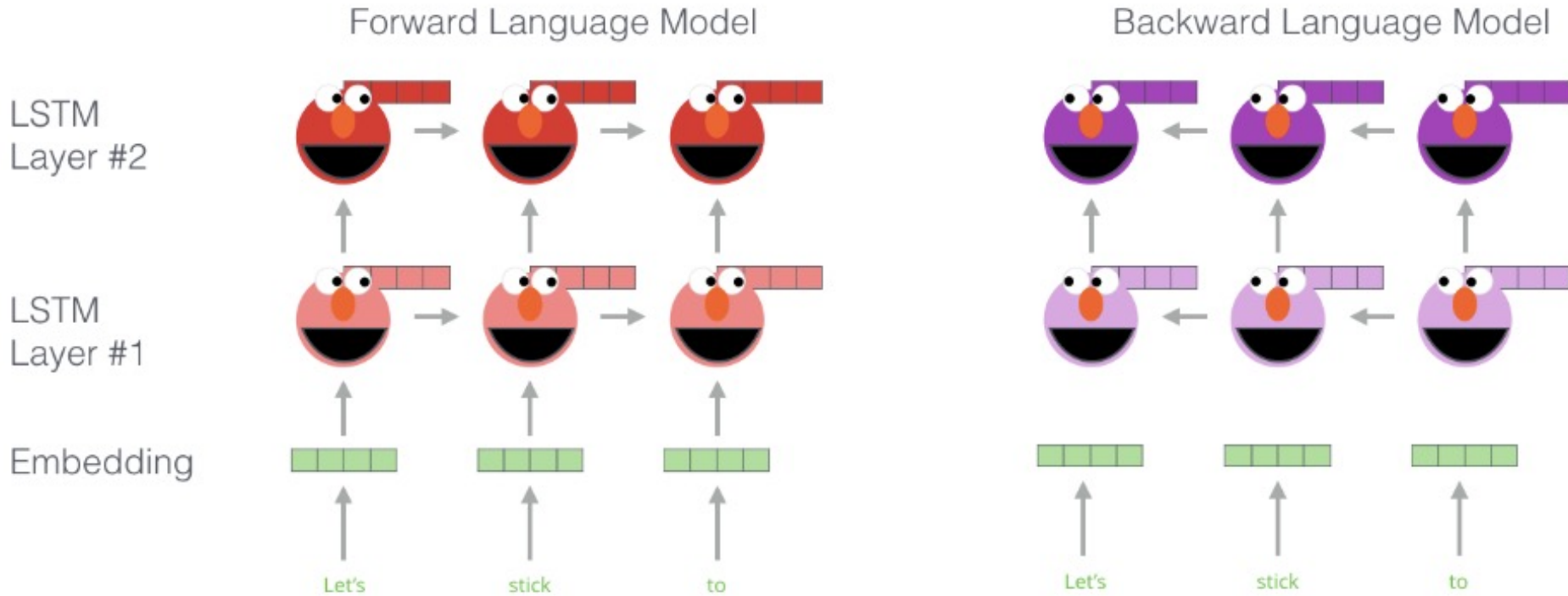
ELMo – Pretraining LMs to obtain context aware word embeddings



Figures from J. Alammari's "The Illustrated BERT, ELMo, and co."

<http://jalammar.github.io/illustrated-bert/>. ELMo paper: Peters et al. "Deep Contextualized Word Representations", NAACL-HLT 2018. <http://aclweb.org/anthology/N18-1202>

ELMo – Pretraining LMs to obtain context aware word embeddings



Figures from J. Alammari's "The Illustrated BERT, ELMo, and co."

<http://jalammar.github.io/illustrated-bert/>. ELMo paper: Peters et al. "Deep Contextualized Word Representations", NAACL-HLT 2018. <http://aclweb.org/anthology/N18-1202>

ELMo – Pretraining LMs to obtain context aware word embeddings

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers



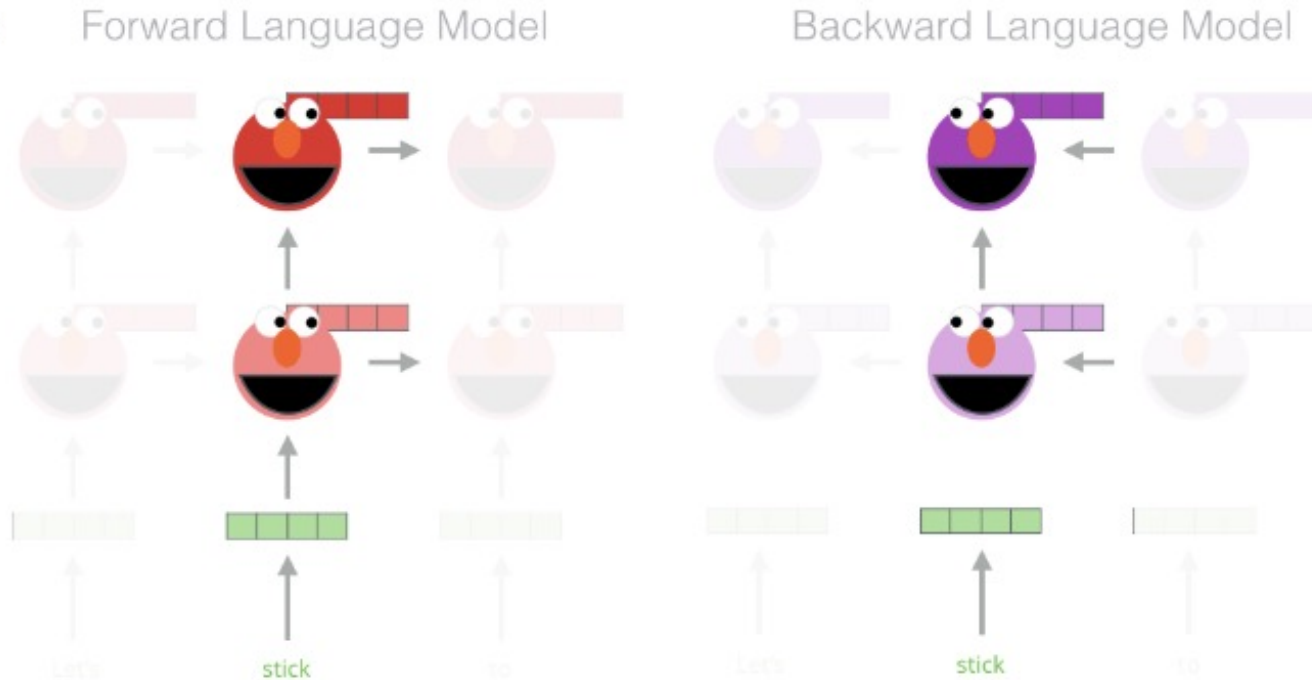
2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context



Figures from J. Alammr’s “The Illustrated BERT, ELMo, and co.”

<http://jalammr.github.io/illustrated-bert/>. ELMo paper: Peters et al. “Deep Contextualized Word Representations”, NAACL-HLT 2018. <http://aclweb.org/anthology/N18-1202>

Recommended reading

- Y. Goldberg, *Neural Network Models for Natural Language Processing*, Morgan & Claypool Publishers, 2017.
 - Mostly chapters 14–17.
- Jurafsky and Martin's, *Speech and Language Processing* is being revised (3rd edition) to include DL methods.
 - <http://web.stanford.edu/~jurafsky/slp3/>
- F. Chollet, *Deep Learning in Python*, 1st edition, Manning Publications, 2017.
 - 1st edition freely available (and sufficient for this course): <https://www.manning.com/books/deep-learning-with-python>
 - See mostly sections 6.1–6.3, section 8.1.
 - 2nd edition (2022) now available, requires payment. Highly recommended.
- See also the recommended reading and resources of the previous part (NLP with MLPs) of this course.

