Neural Language Models & Attention

Lara J. Martin (she/they)

https://laramartin.net/interactive-fiction-class

Slides modified from Dr. Daphne Ippolito & Dr. Frank Ferraro

Learning Objectives

Discover the basic function of language models

Determine why sequence-to-sequence models emerged from the regular RNN model

Explore the components of RNNs and seq2seq models

Understand the utility of attention mechanisms

Old-School
Interactive Fiction

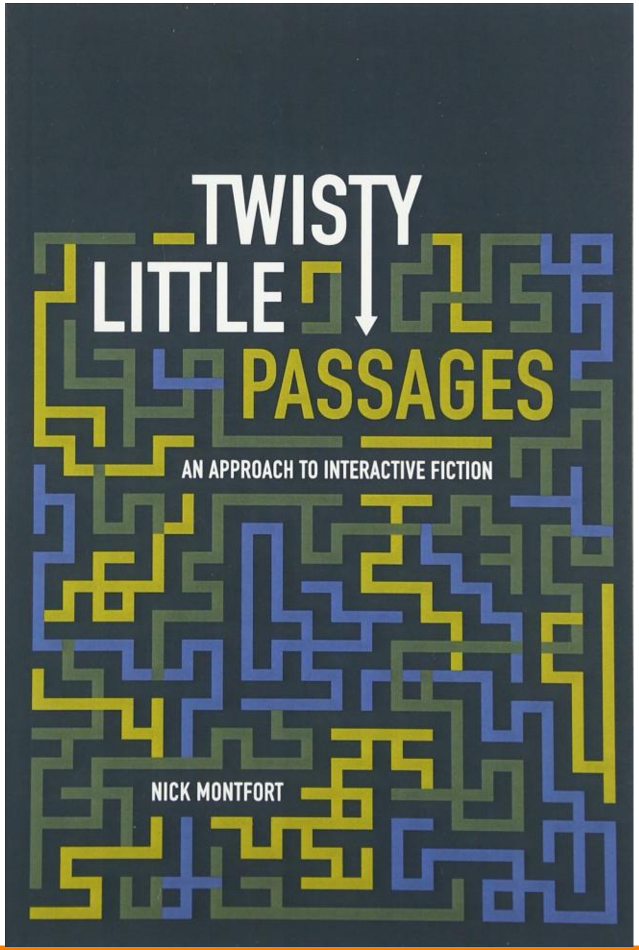
Full Action Space

Limited Action Space

Tabletop Roleplaying Games

Choose-Your-Own Adventures

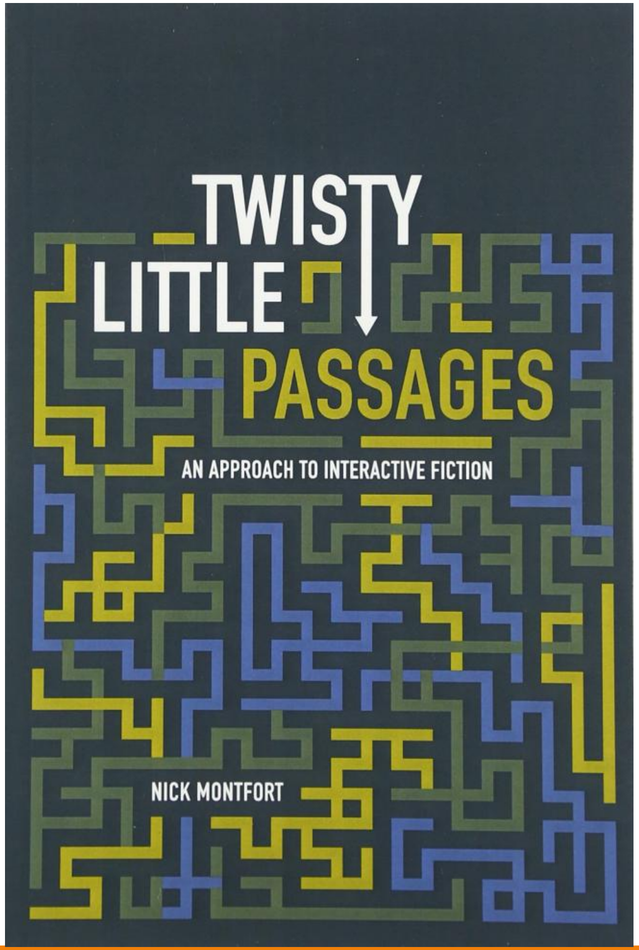
Review: Components of Interactive Fiction Games



The **parser**, which is the component that analyzes natural language input in an interactive fiction work.

The **world model**, which is setting of an interactive fiction work.

Review: Components of Interactive Fiction Games



The **parser,** which is the component that analyzes natural language input in an interactive fiction work.

The **world model,** which is setting of an interactive fiction work.

You just started up a game
and now you're staring at
text and a *blinking cursor*
and you *don't know what to do*! (> |)

Don't panic kids—
Crazy Uncle Zarf is here to help you get started . . .

These commands are very common:

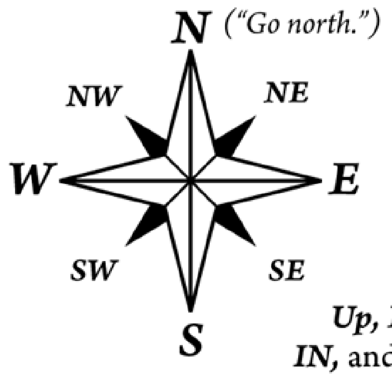
EXAMINE *it* — **PUSH** *it*
TAKE *it* — **PULL** *it*
DROP *it* — **TURN** *it*
OPEN *it* — **FEEL** *it*
PUT *it* **IN** *something*
PUT *it* **ON** *something*

When in doubt, examine more.

Does the game intro suggest **ABOUT, INFO, HELP**? *Try them first!*

You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox* here.

*Try opening!



Also: ***Up, Down, IN,*** and ***OUT***

You can try all sorts of commands on the things you see.
Try the commands that make sense!
Doors are for opening; buttons are for pushing; pie is for eating. *(Mmm, pie.)*

◇◇◇◇

If you meet a person, these should work:

TALK TO *name*
ASK *name* **ABOUT** *something*
TELL *name* **ABOUT** *something*
GIVE *something* **TO** *name*
SHOW *something* **TO** *name*

Each game has slightly different commands, but they all look ***pretty much like these.***

You could also try:

EAT *it* — **CLIMB** *it*
DRINK *it* — **WAVE** *it*
FILL *it* — **WEAR** *it*
SMELL *it* — **TAKE** *it* **OFF**
LISTEN TO *it* — **TURN** *it* **ON**
BREAK *it* — **DIG IN** *it*
BURN *it* — **ENTER** *it*
LOOK UNDER *it* — **SEARCH** *it*
UNLOCK *it* **WITH** *something*

Or even:

LISTEN — **JUMP**
SLEEP — **PRAY**
WAKE UP — **CURSE**
UNDO† — **SING**

†*Take back one move — handy!*

"What if I only want to type one or two letters?"

◇◇◇◇

N/E/S/W/NE/SE/NW/SW: GO *in the indicated compass direction.*
L: LOOK *around to see what is nearby.*
X: EXAMINE *a thing in more detail.*
I: *take* **INVENTORY** *of what you possess.*
Z: WAIT *a turn without doing anything.*
G: *do the same thing* **AGAIN**

◇◇◇◇

A service of the People's Republic of Interactive Fiction: ***http://pr-if.org***

Review: Why were parsers so bad?

Limited computational resources. Computers had ≤128 KB of memory

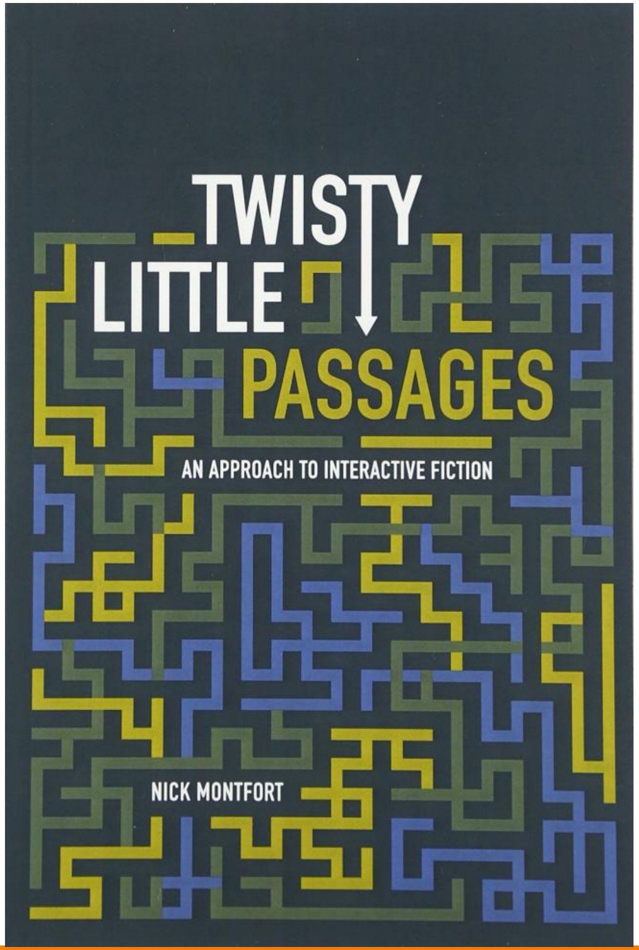
Language is difficult. There are many things that make human languages genuinely challenging for a computer to process.

Keyword-based commands. Only exact matches worked properly. No synonyms, no paraphrases.

Everything was manual. Game developers had to anticipate all possible commands, and manually code the responses.

No machine learning. This was prior to the advent of machine learning based natural language processing

Review: Components of Interactive Fiction Games



The **parser,** which is the component that analyzes natural language input in an interactive fiction work.

The **world model,** which is setting of an interactive fiction work.

Review: World Model

It represents the physical environment, and things like
- Settings or locations
- Physical objects in each setting
- The player’s character
- Non-player characters

It also represents and simulates the physical laws of the environment.

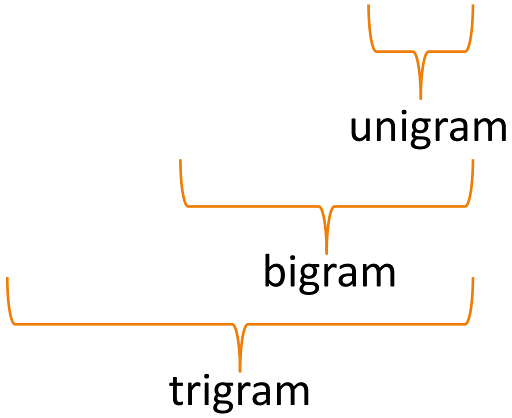
Language Models

What is a language model?

A model, given a history of words, that outputs likely next words.

Originally, they were statistical **n-gram models**.

The quick brown fox jumps over the lazy dog.



What is a language model?

A model, given a history of words, that outputs likely next words.

Originally, they were statistical **n-gram models**.

trigram

The quick brown fox jumps over the lazy dog.

trigram

trigram

n-gram: any consecutive n # of words, treated as a single unit

What is a language model?

A model, given a history of words, that outputs **likely** next words.

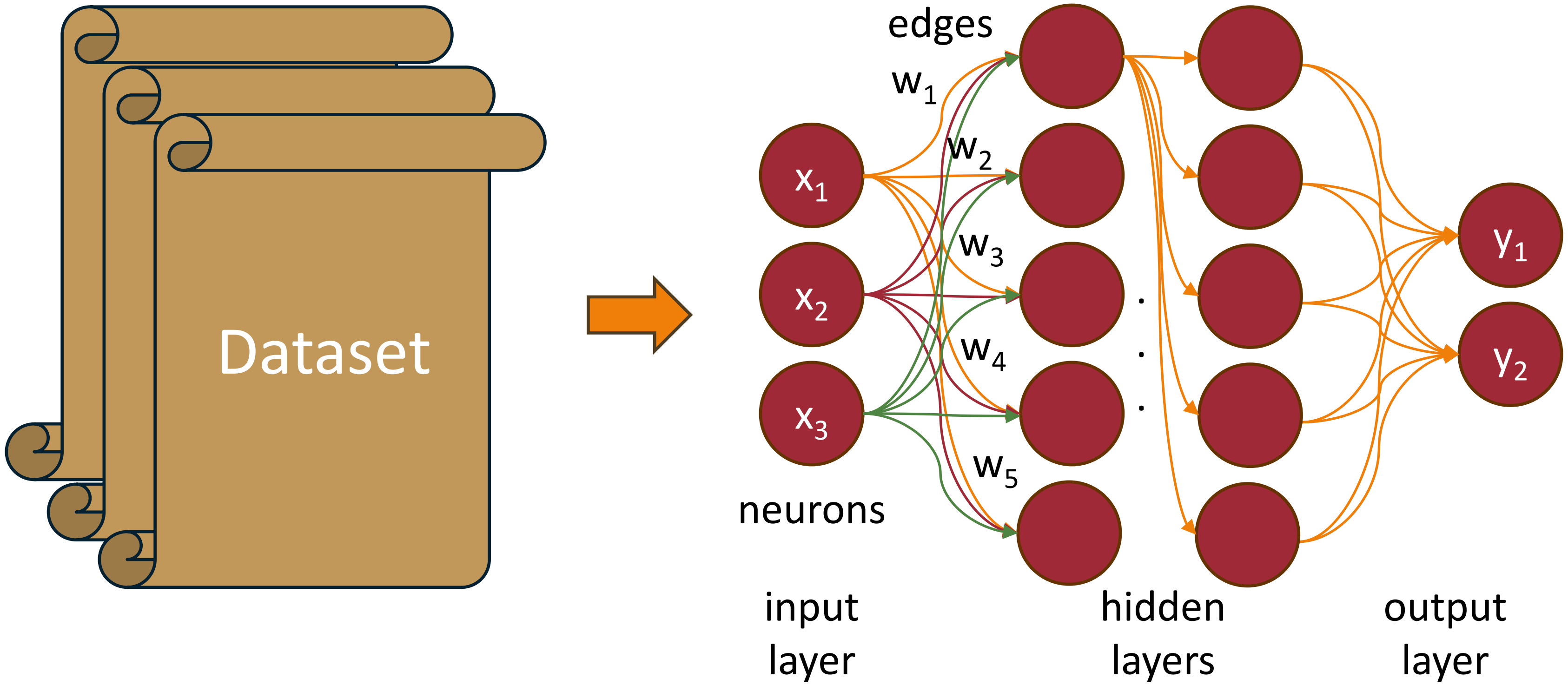
P(“apple” | “eat the”) = 0.02

P(“pineapple” | “eat the”) = 0.01

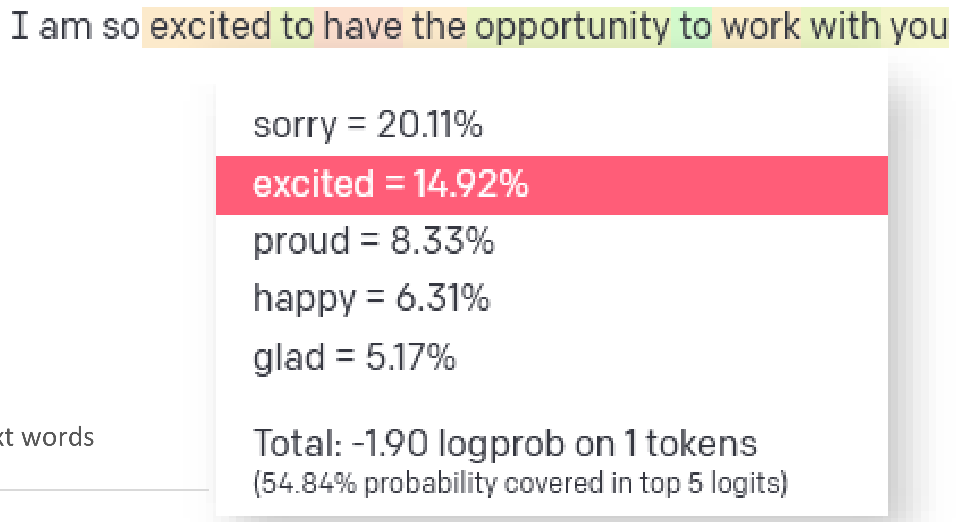
P(“suitcase” | “eat the”) = 0.0001

What is a *neural* language model?

What’s a neural network?

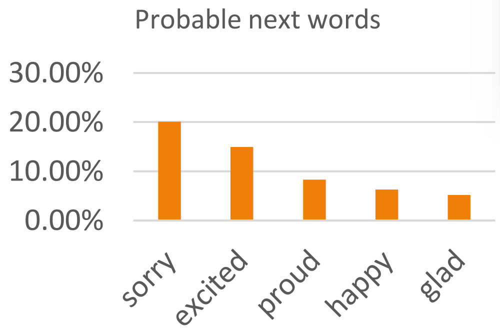


Using a neural language model



I am so

excited



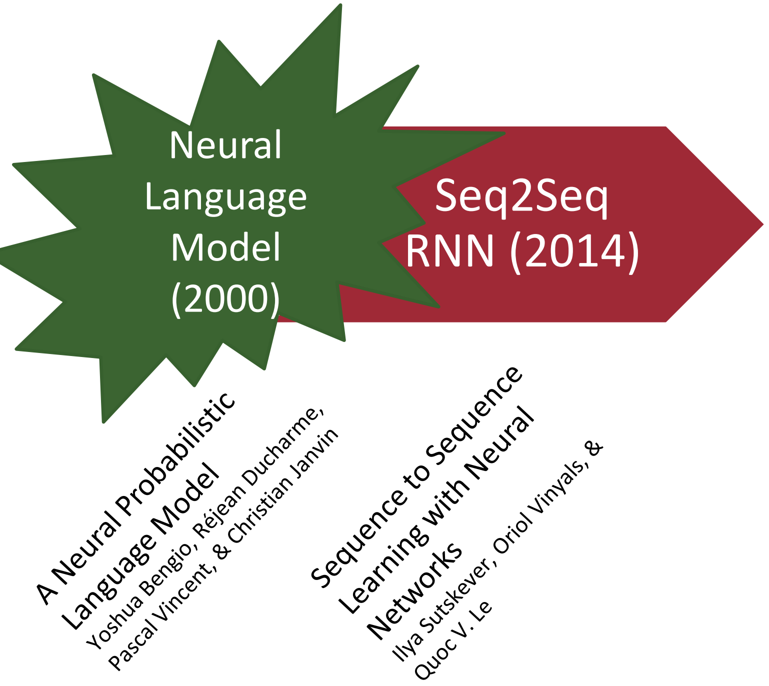
Example created with text-davinci-003 on openai.com in Feb. 2023

Neural Language Model Timeline

Neural Language Model (2000)

A Neural Probabilistic Language Model
Yoshua Bengio, Réjean Ducharme, Pascal Vincent, & Christian Janvin

Neural Language Model Timeline



Sequence-to-Sequence RNNs

Up until 2017 or so, neural language models were mostly built using recurrent neural networks.

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever Google ilyasu@google.com

Oriol Vinyals Google vinyals@google.com

Quoc V. Le Google qvl@google.com

v:1409.3215v3 [cs.CL] 14 Dec 2014

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

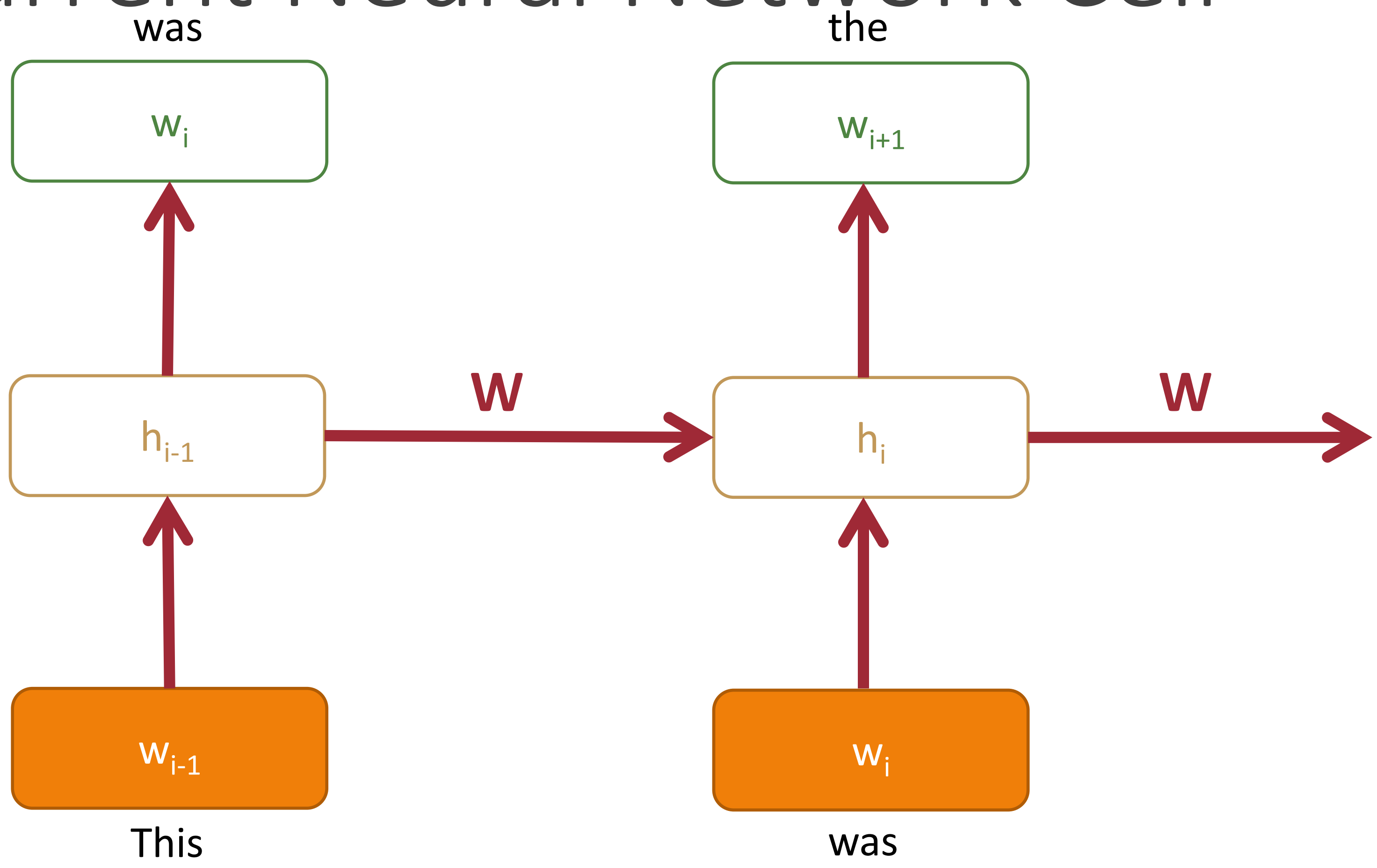
Generating Sequences With Recurrent Neural Networks

Alex Graves
Department of Computer Science
University of Toronto
graves@cs.toronto.edu

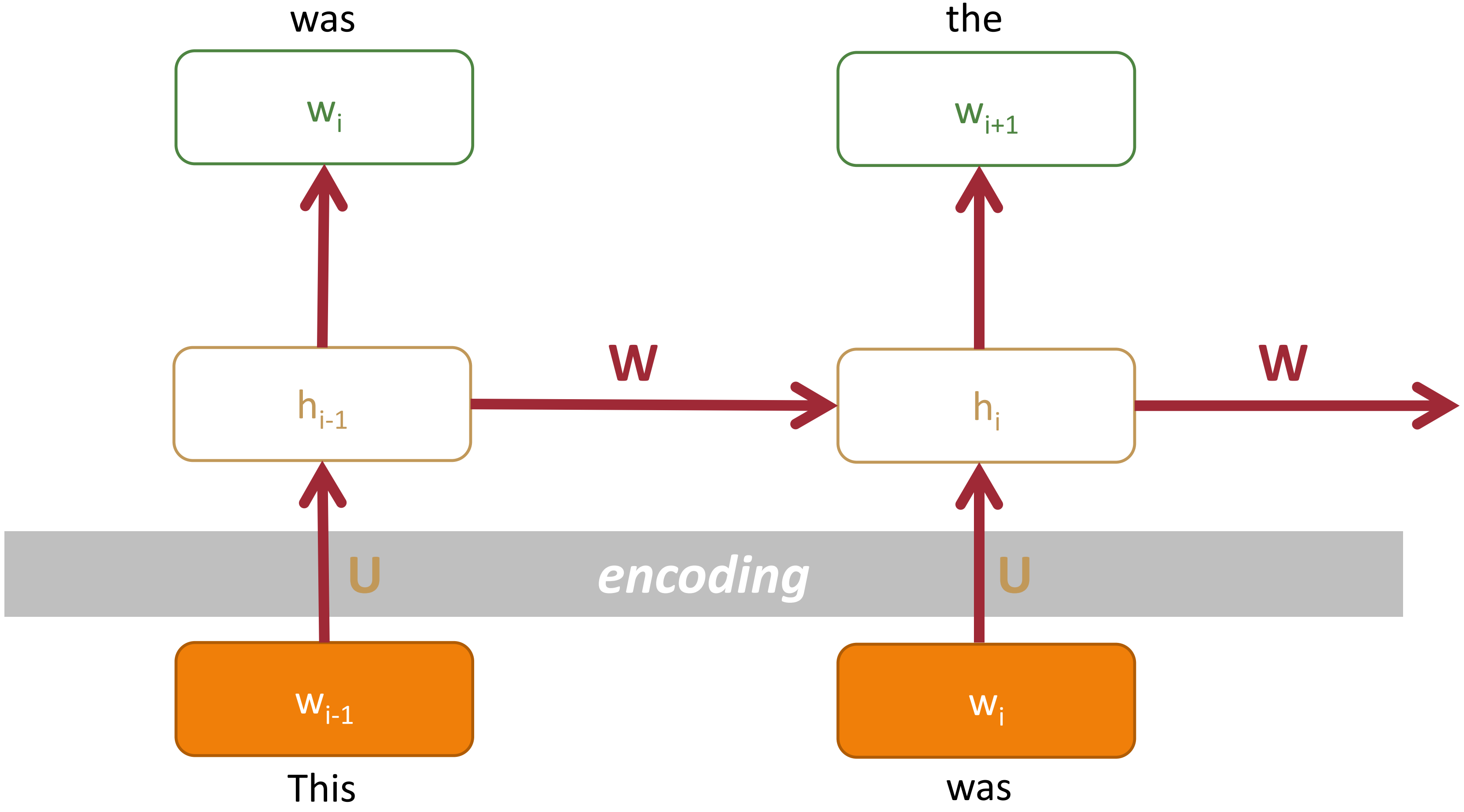
Abstract

This paper shows how Long Short-term Memory recurrent neural networks can be used to generate complex sequences with long-range structure, simply by predicting one data point at a time. The approach is demonstrated for text (where the data are discrete) and online handwriting (where the data are real-valued). It is then extended to handwriting synthesis by allowing the network to condition its predictions on a text sequence. The resulting system is able to generate highly realistic cursive handwriting in a wide variety of styles.

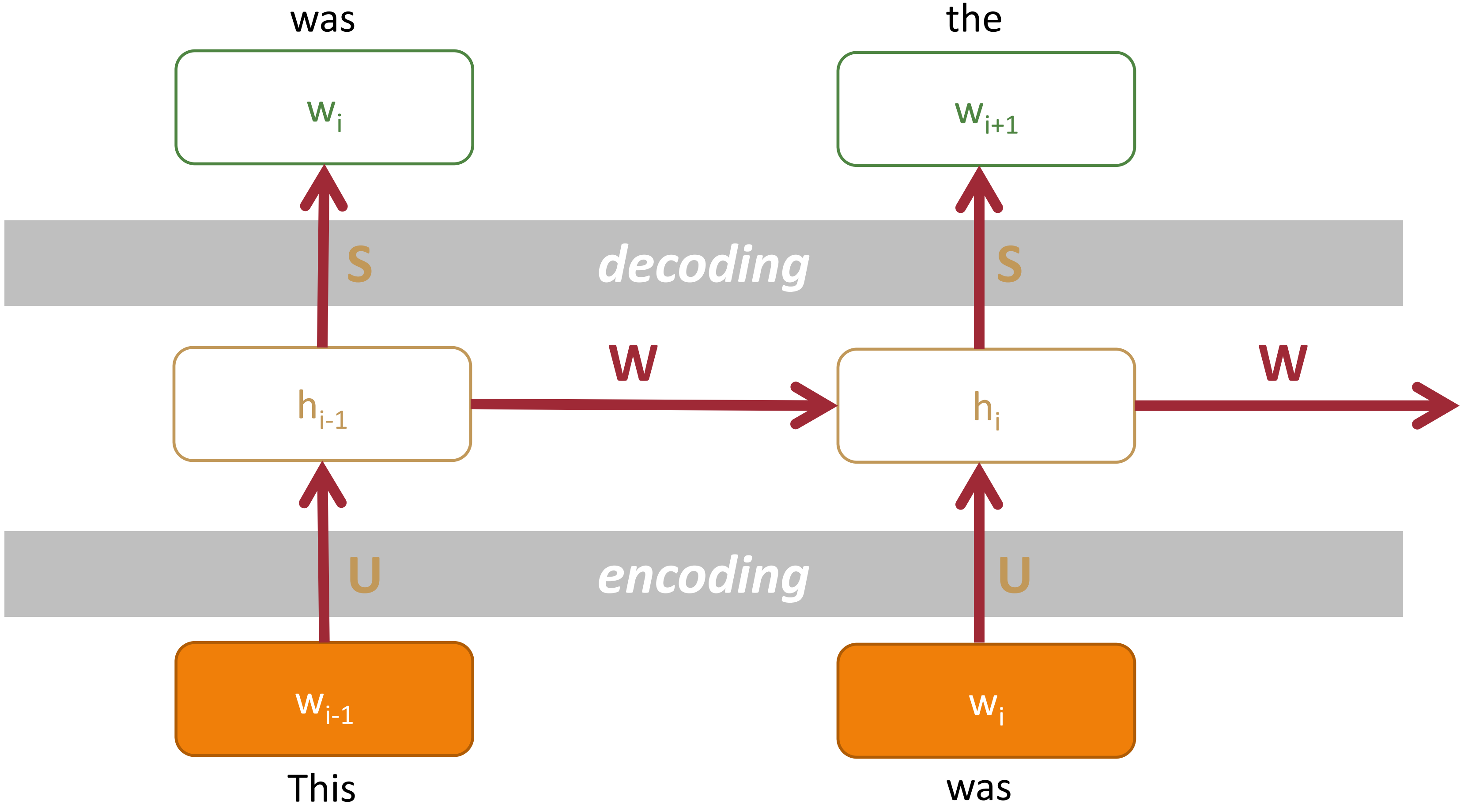
A Recurrent Neural Network Cell



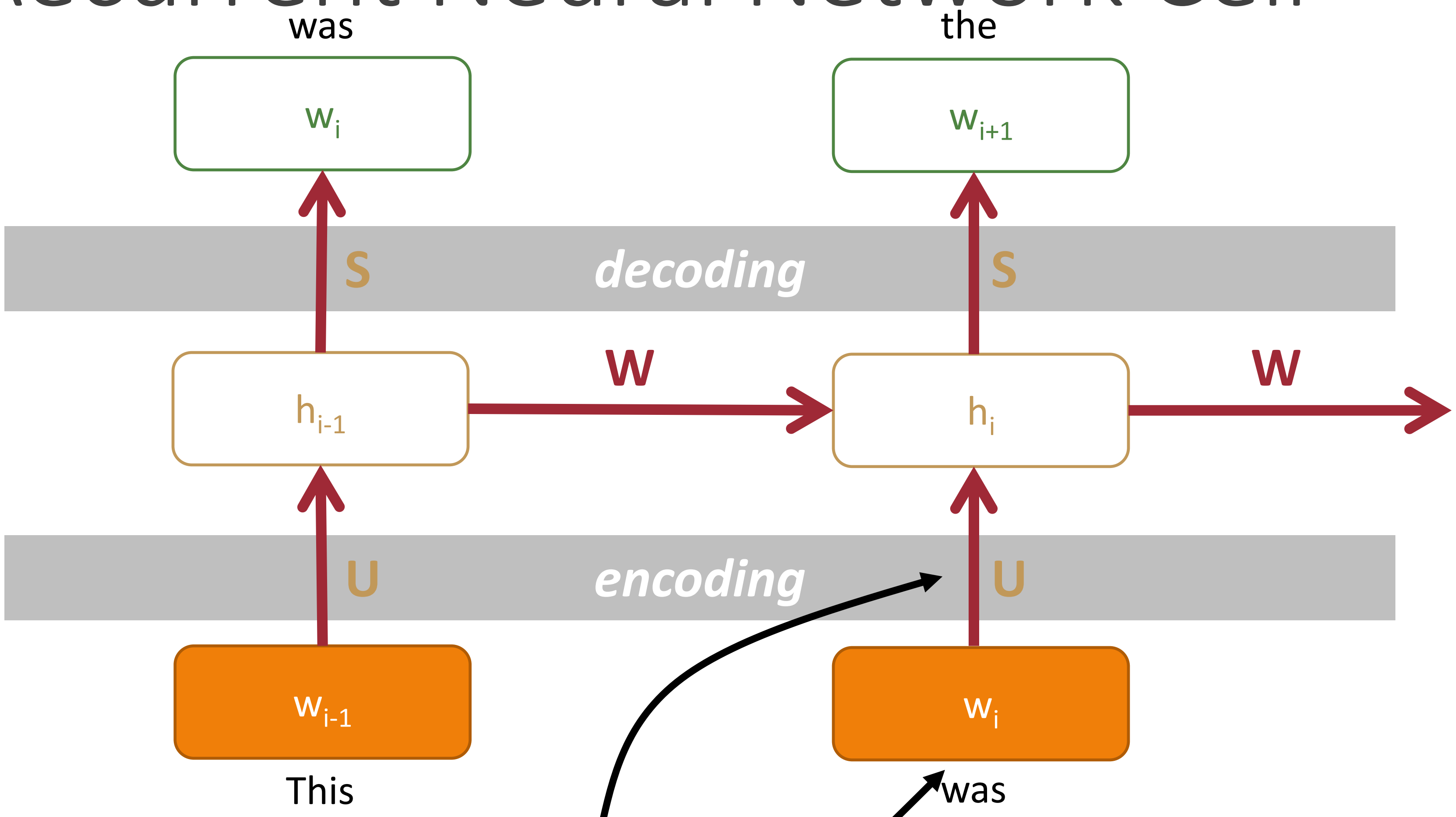
A Recurrent Neural Network Cell



A Recurrent Neural Network Cell



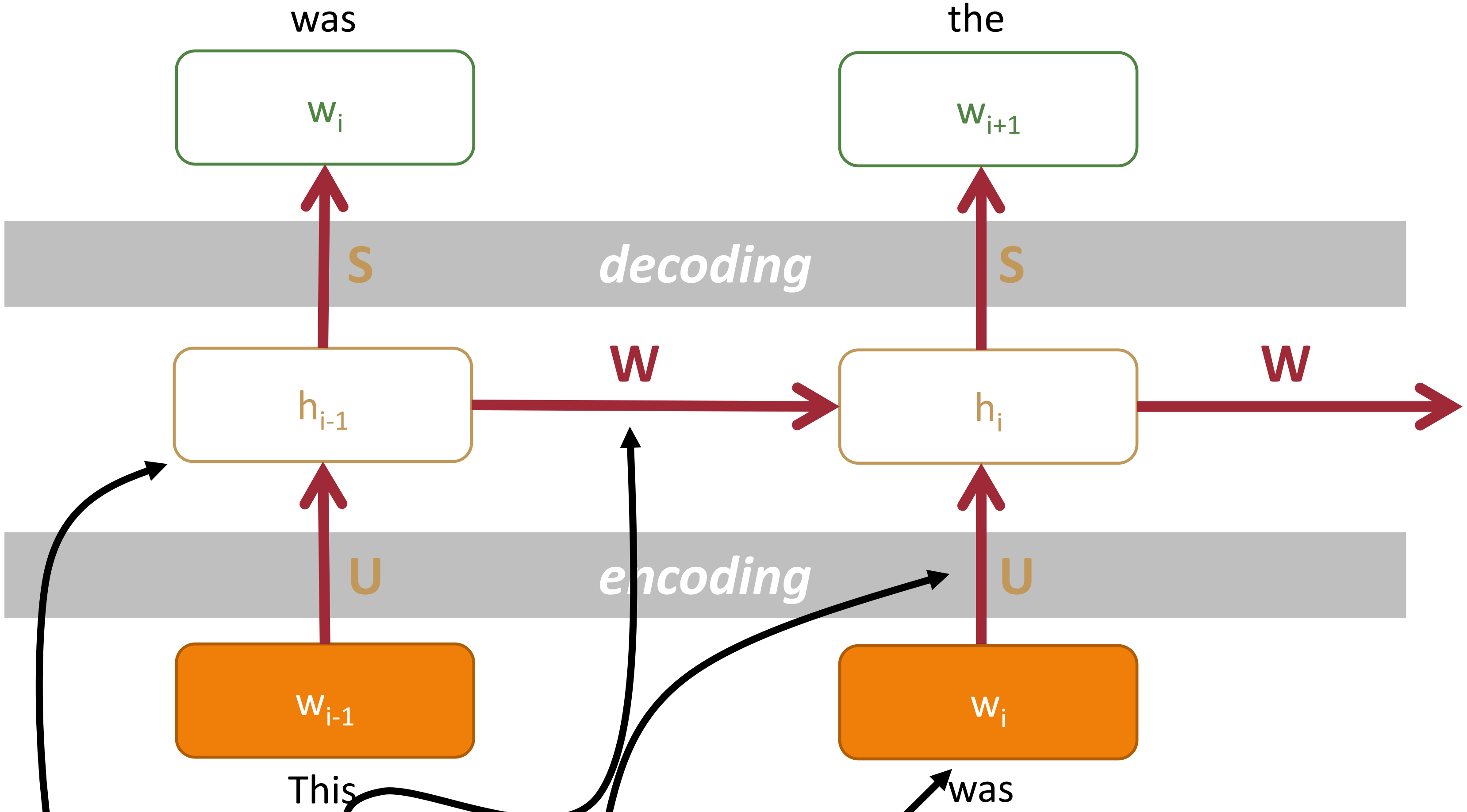
A Recurrent Neural Network Cell



$$h_i = \sigma(W h_{i-1} + U w_i)$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

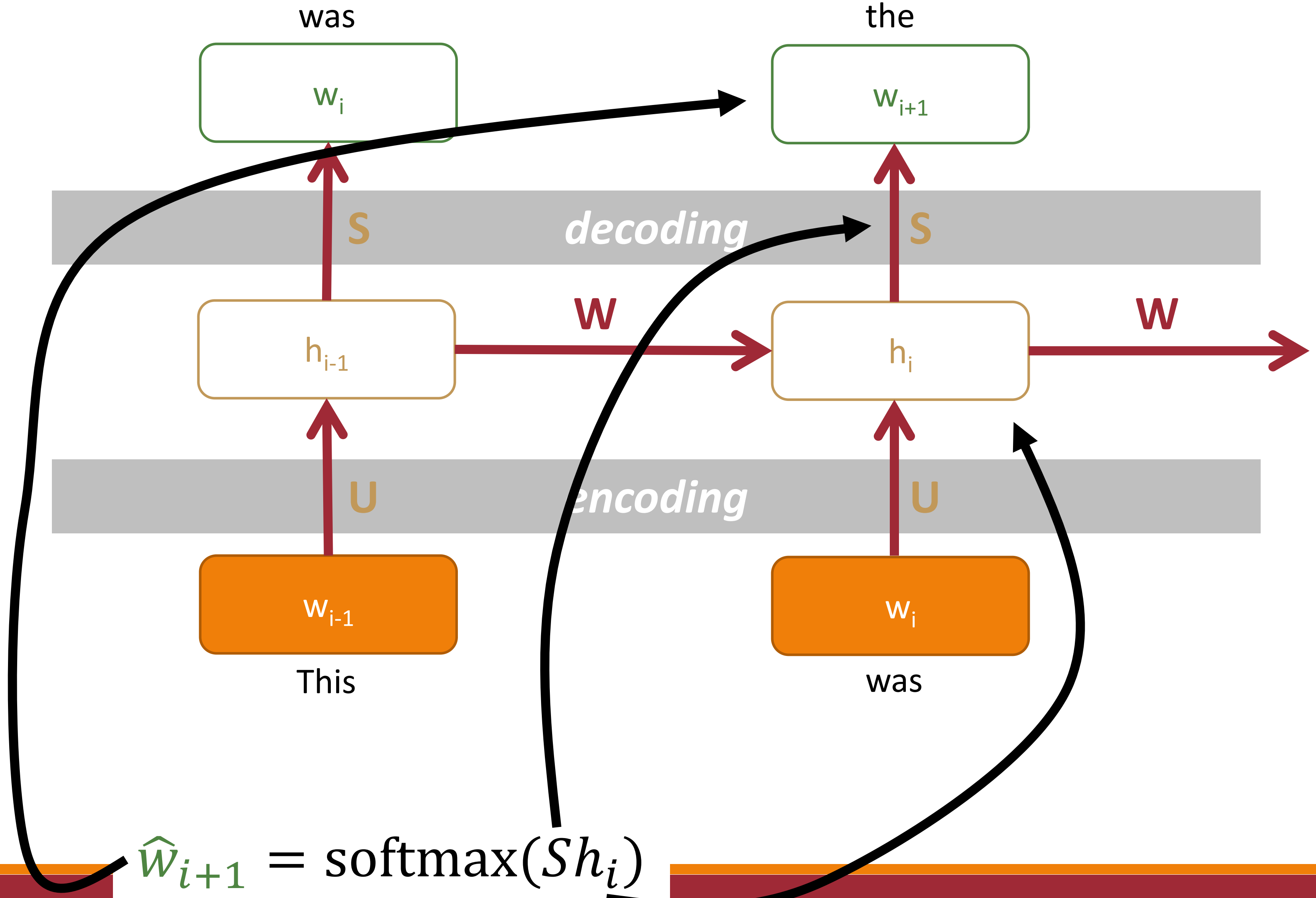
A Recurrent Neural Network Cell



$$h_i = \sigma(W h_{i-1} + U w_i)$$

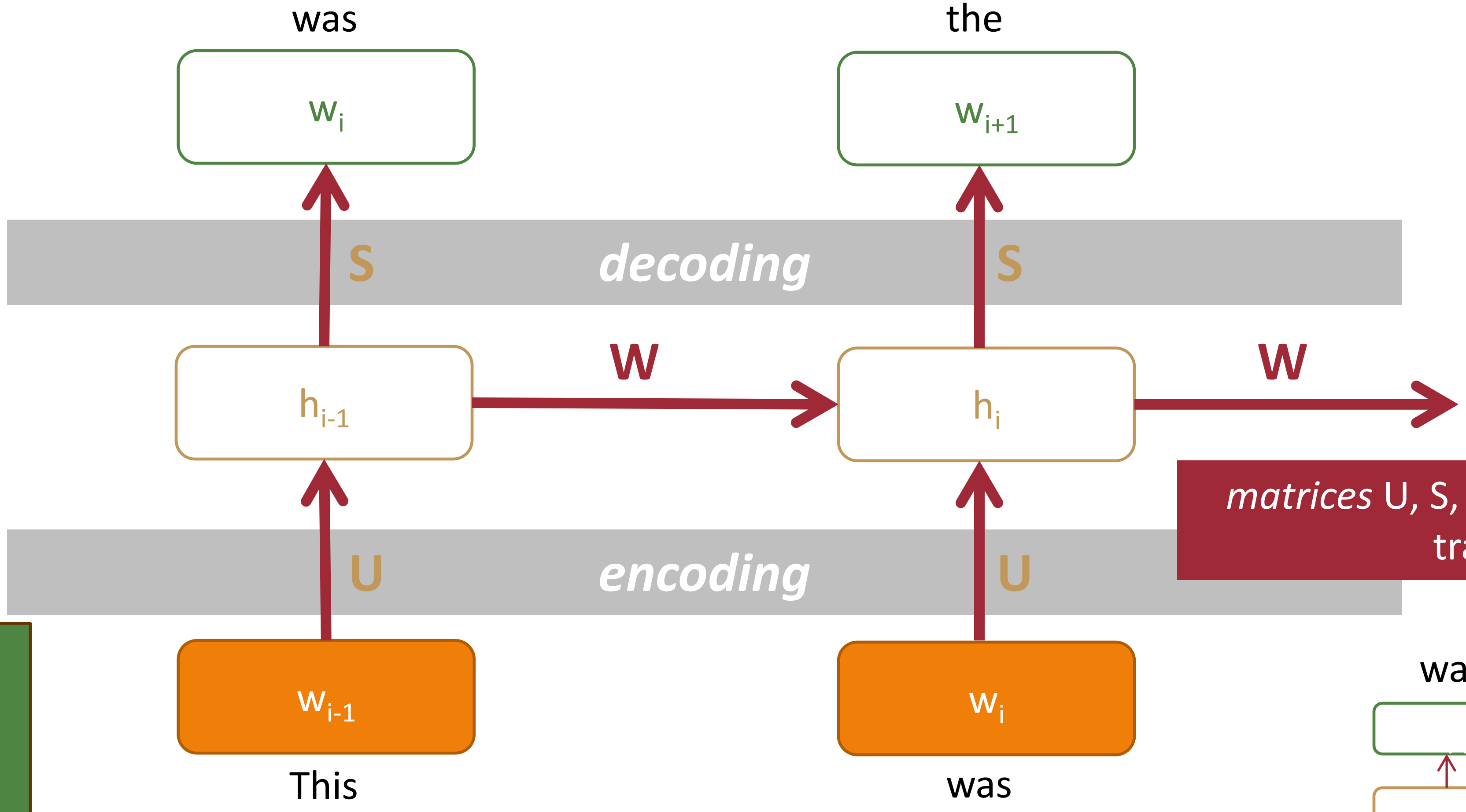
$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

A Recurrent Neural Network Cell



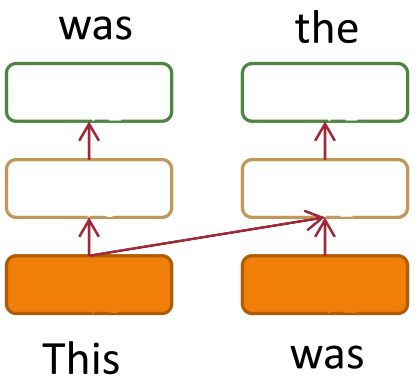
$$\hat{w}_{i+1} = \mathrm{softmax}(Sh_i)$$

A Recurrent Neural Network Cell



matrices U, S, W are learned via training

Why is an RNN considered "recurrent"?



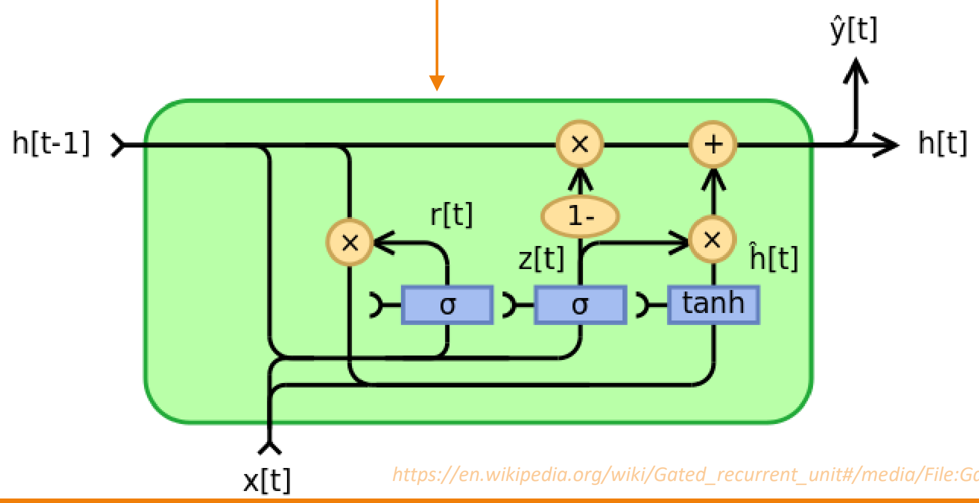
Feed forward

Types of RNN cells: LSTMs/GRUs

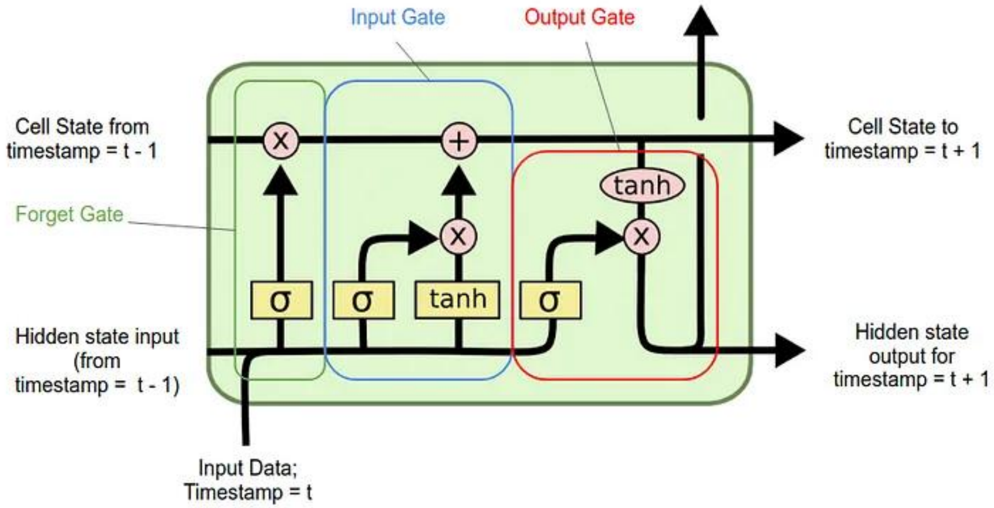
LSTM: Long Short-Term Memory (Hochreiter & Schmidhuber, 1997)

LSTMs were originally designed to keep around information for longer in the hidden state as it gets repeatedly updated.

GRU: Gated Recurrent Unit (Cho et al., 2014)

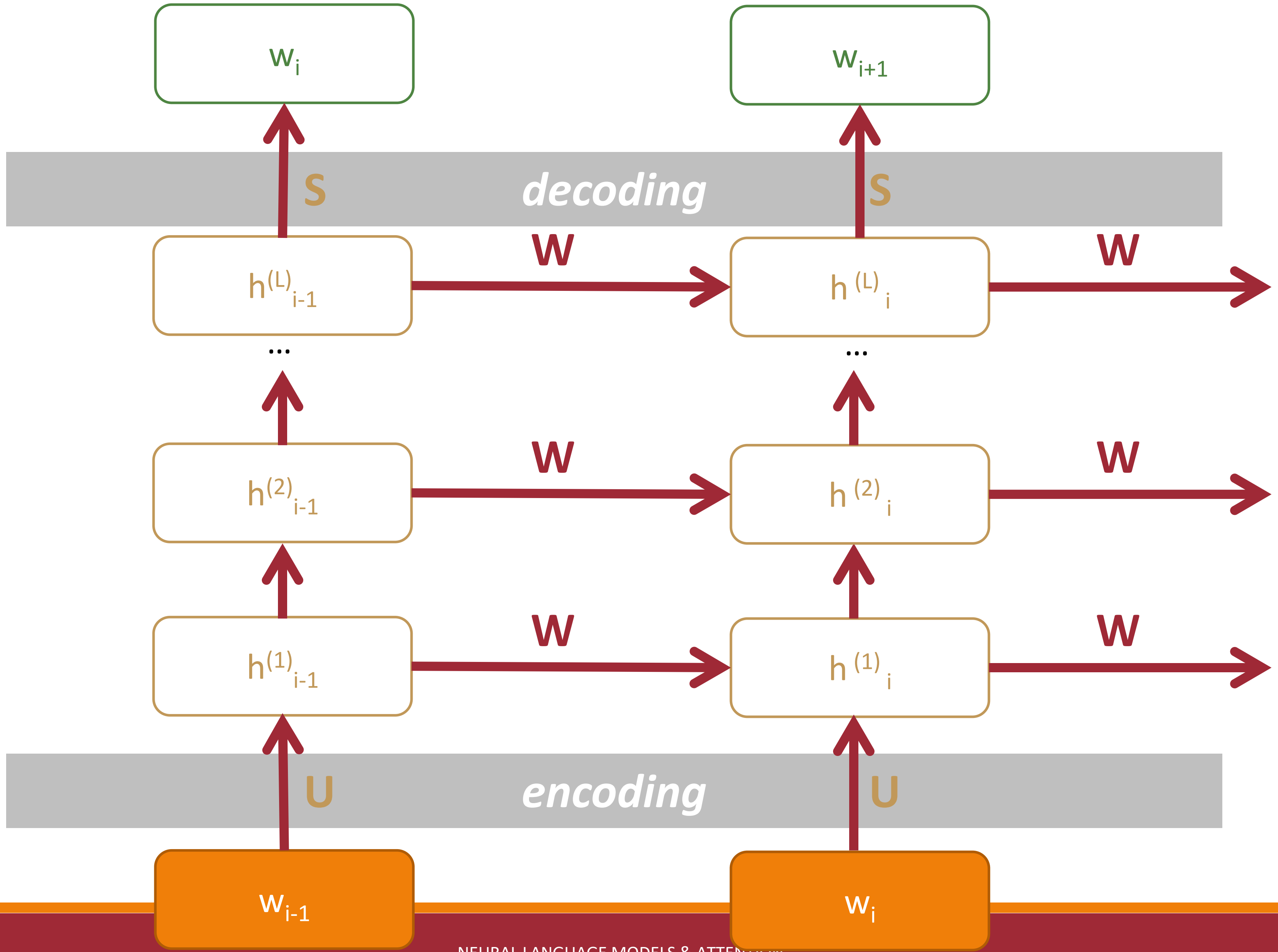


https://en.wikipedia.org/wiki/Gated_recurrent_unit#/media/File:Gated_Recurrent_Unit,_base_type.svg

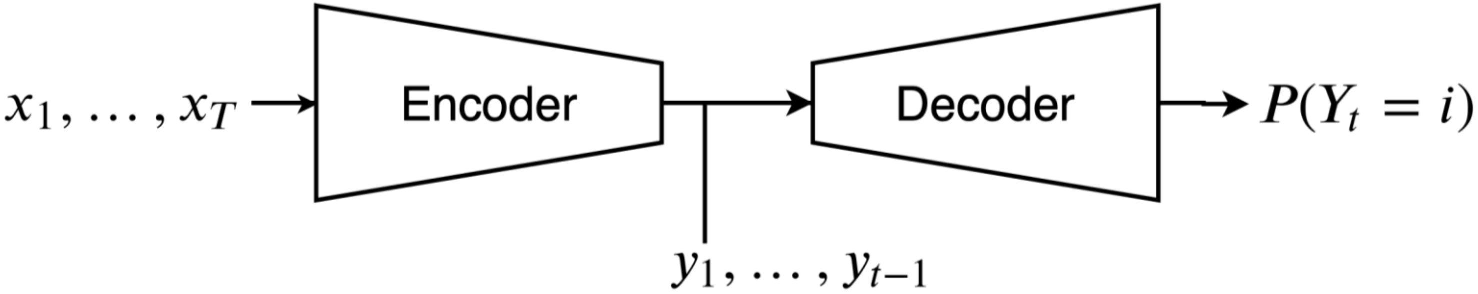


https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2

A Multi-Layer Recurrent Neural Network Cell

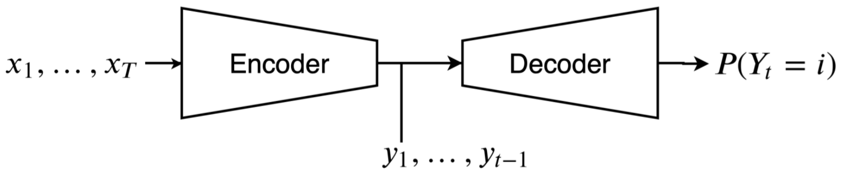


Sequence-to-Sequence / Encoder-Decoder Models

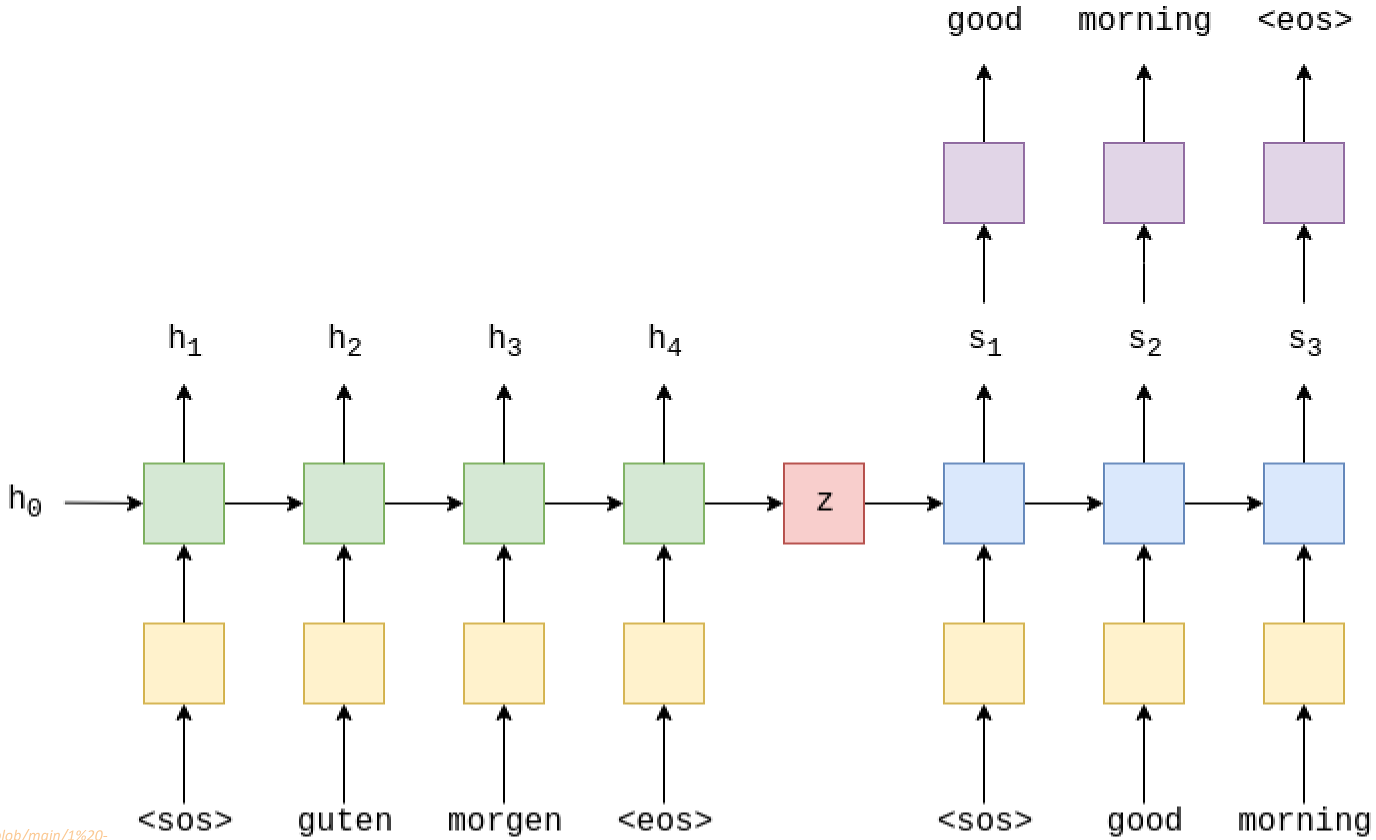


Think-pair-share:
Sequence-to-sequence models divided up the encoder and decoder components of RNNs.
Why do you think this was helpful?

I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2014, pp. 3104–3112. https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

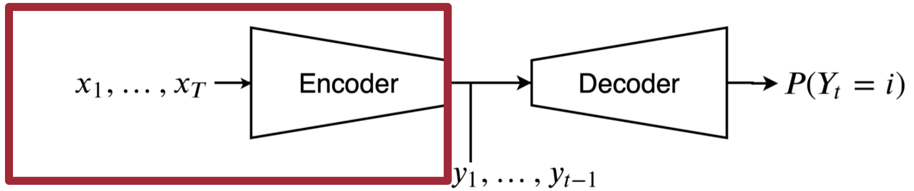


Sequence-to-Sequence / Encoder-Decoder Models



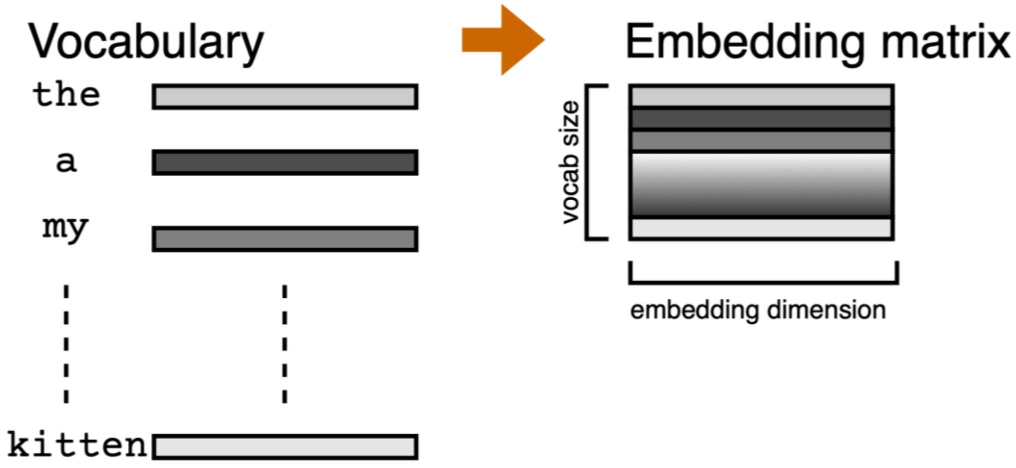
https://colab.research.google.com/github/bentrevett/pytorch-seq2seq/blob/main/1%20-%20Sequence%20to%20Sequence%20Learning%20with%20Neural%20Networks.ipynb#scrollTo=k6sRrL4wKsmi

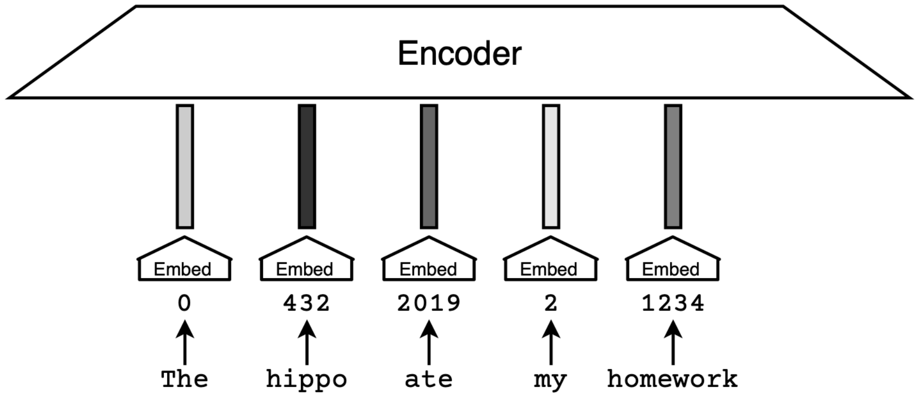
I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Conference on Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, 2014, pp. 3104–3112. https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

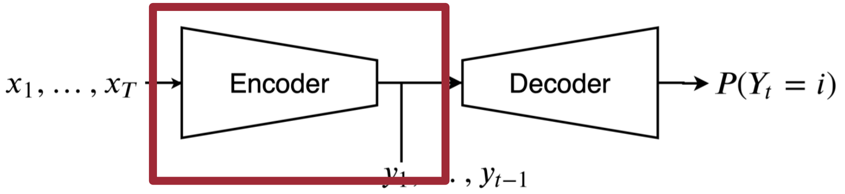


Inputs to the Encoder

The encoder takes as input the embeddings corresponding to each token in the sequence.

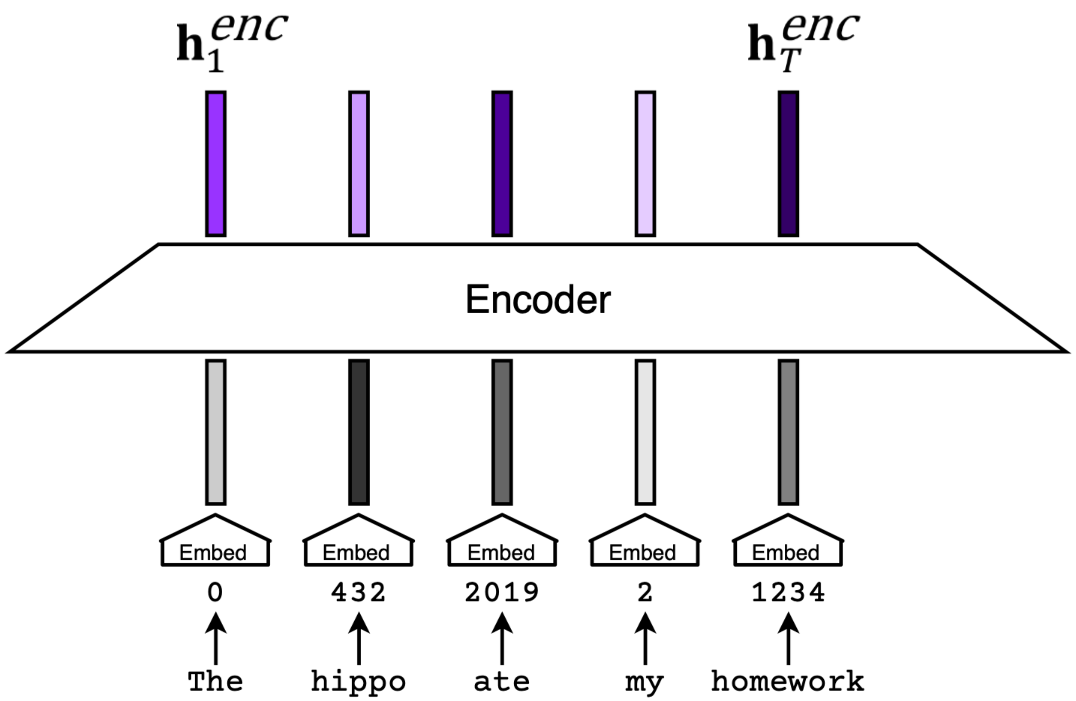


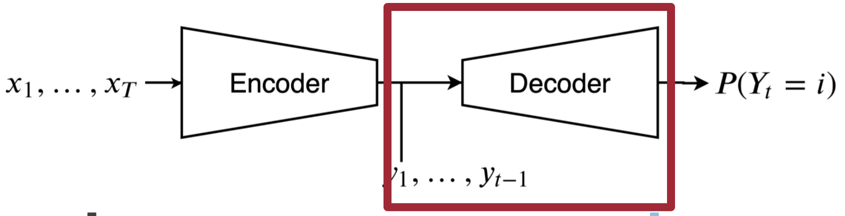




Outputs from the Encoder

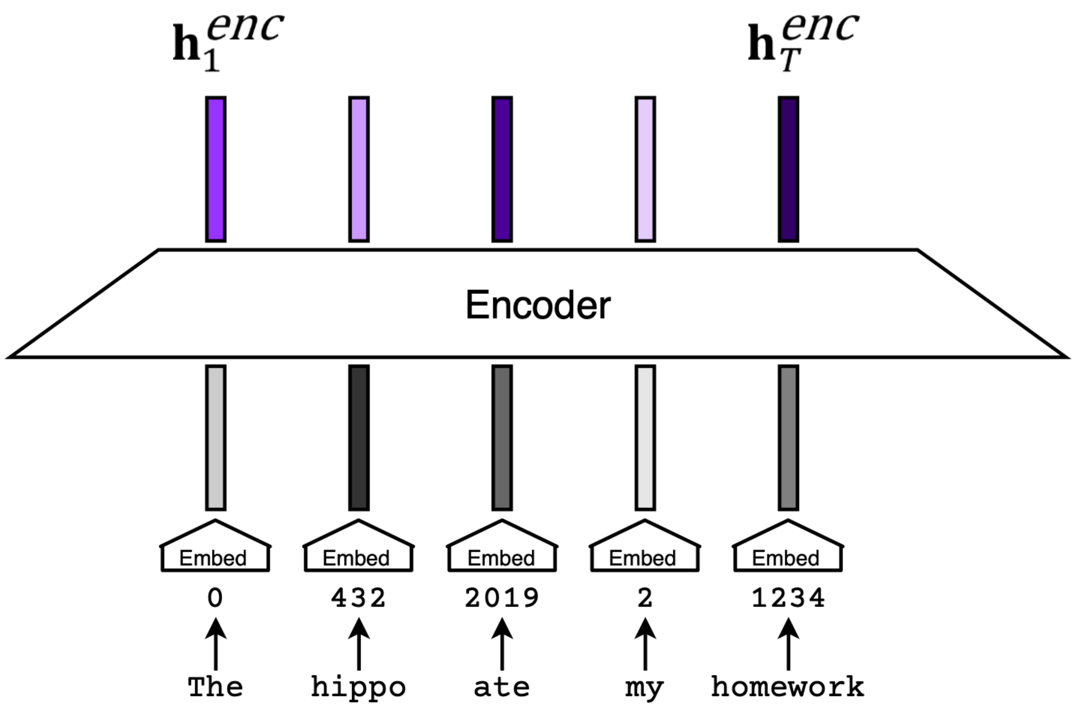
The encoder outputs a sequence of vectors. These are called the hidden state of the encoder.

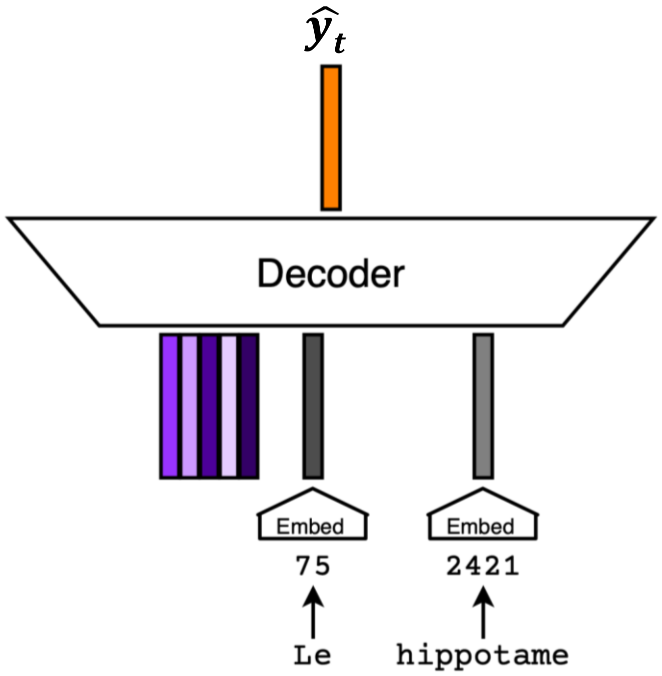


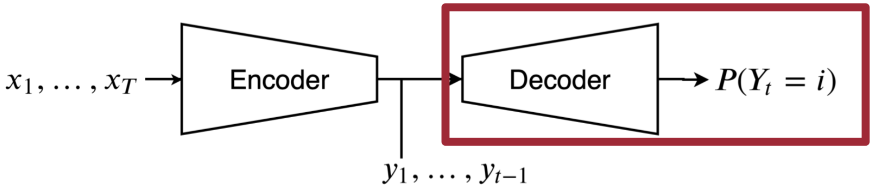


Inputs to the Decoder

The decoder takes as input the hidden states from the encoder as well as the embeddings for the tokens seen so far in the target sequence.

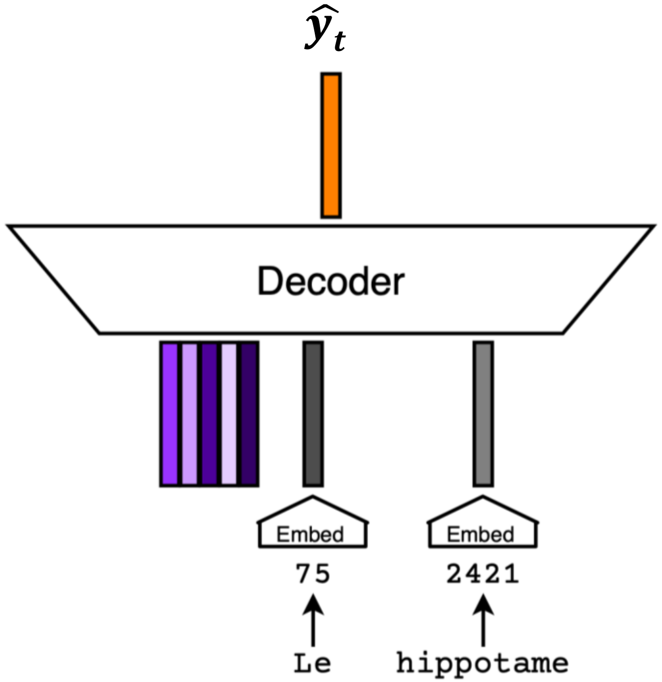






Outputs from the Decoder

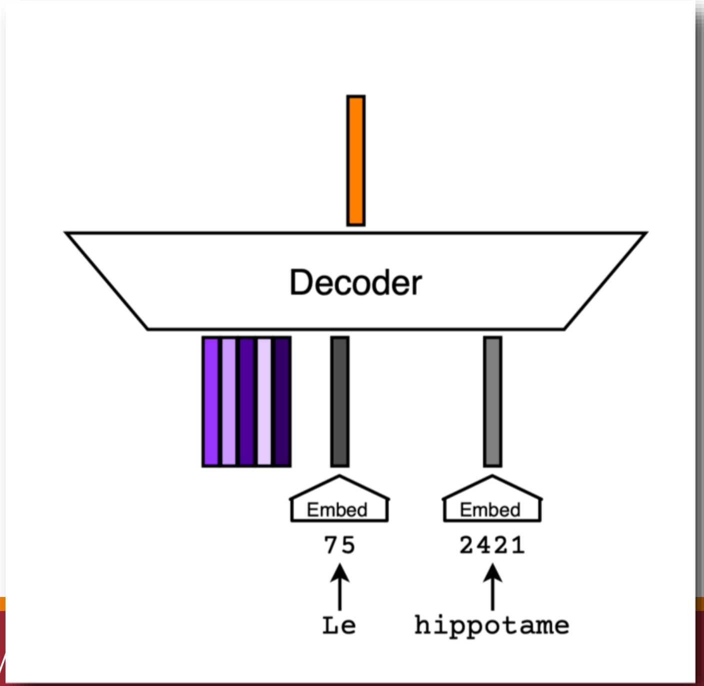
The decoder outputs an embedding $\widehat{\boldsymbol{yt}}$. The goal is for this embedding to be as close as possible to the embedding of the true next token.

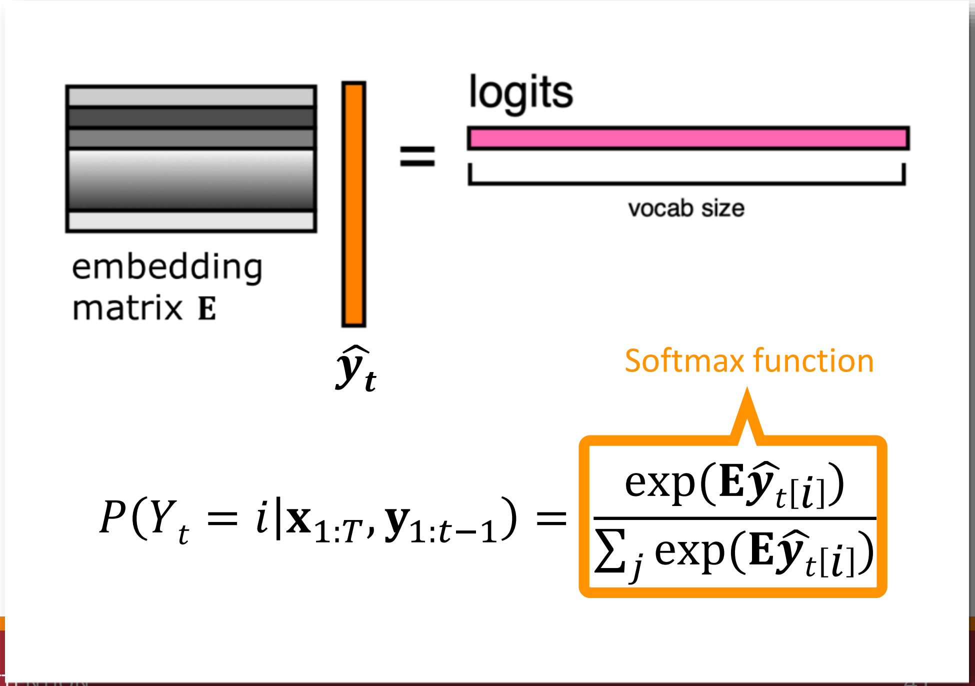


Turning $\widehat{yt}$ into a Probability Distribution

We can multiply the predicted embedding $\widehat{yt}$ by our vocabulary embedding matric to get a score for each vocabulary word. These scores are referred to as logits.

The **softmax function** then lets us turn the logits into probabilities.

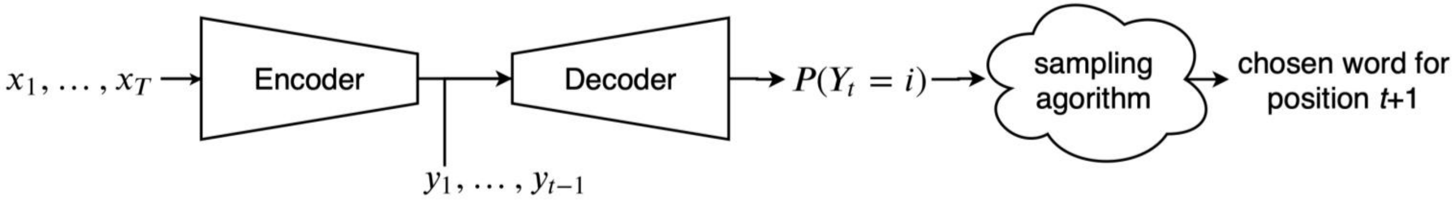




Generating Text

Also sometimes called decoding 🙄

To generate text, we need an algorithm that selects tokens given the predicted probability distributions.



More on this in the next lecture!

Loss Function

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(Y_t = \boxed{i^*} | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

The index of the true tth word in the target sequence.

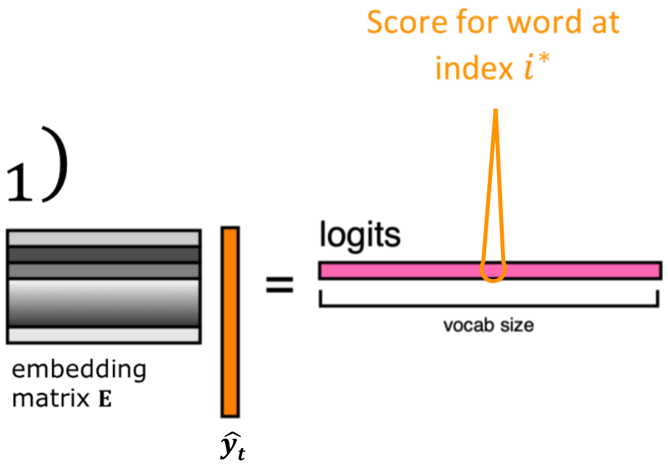
Loss Function

$$\mathcal{L} = -\sum_{t=1}^{T} \log \boxed{P(Y_t = i^* | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})}$$

The probability the language model assigns to the true tth word in the target sequence.

Loss Function

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$



$$= -\sum_{t=1}^{T} \log \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i^*])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}$$

$$P(Y_t = i | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1}) = \frac{\exp(\mathbf{E}\hat{\boldsymbol{y}}_{t[i]})}{\sum_j \exp(\mathbf{E}\hat{\boldsymbol{y}}_{t[i]})}$$

Loss Function

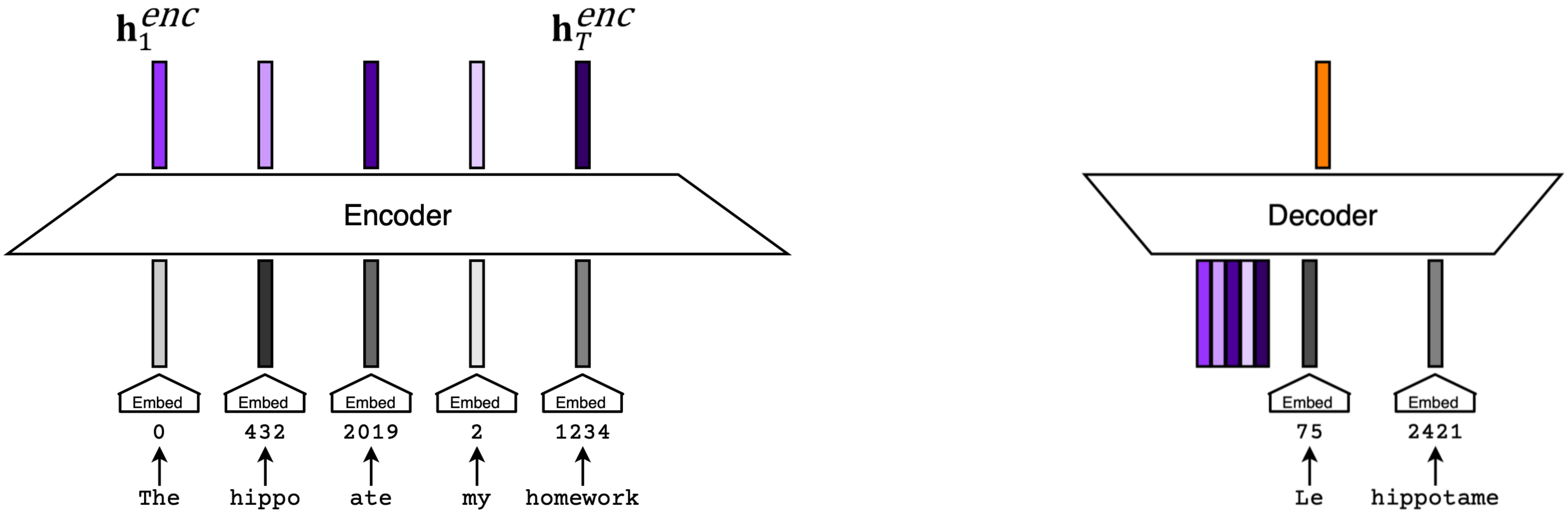
$$\mathcal{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* | \mathbf{x}_{1:T}, \mathbf{y}_{1:t-1})$$

$$= -\sum_{t=1}^{T} \cancel{\log} \frac{\cancel{\exp}(\mathbf{E}\mathbf{y}_t[i^*])}{\cancel{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])}}$$

$$= -\sum_{t=1}^{T} \mathbf{E}\hat{\mathbf{y}}_t[i^*]$$

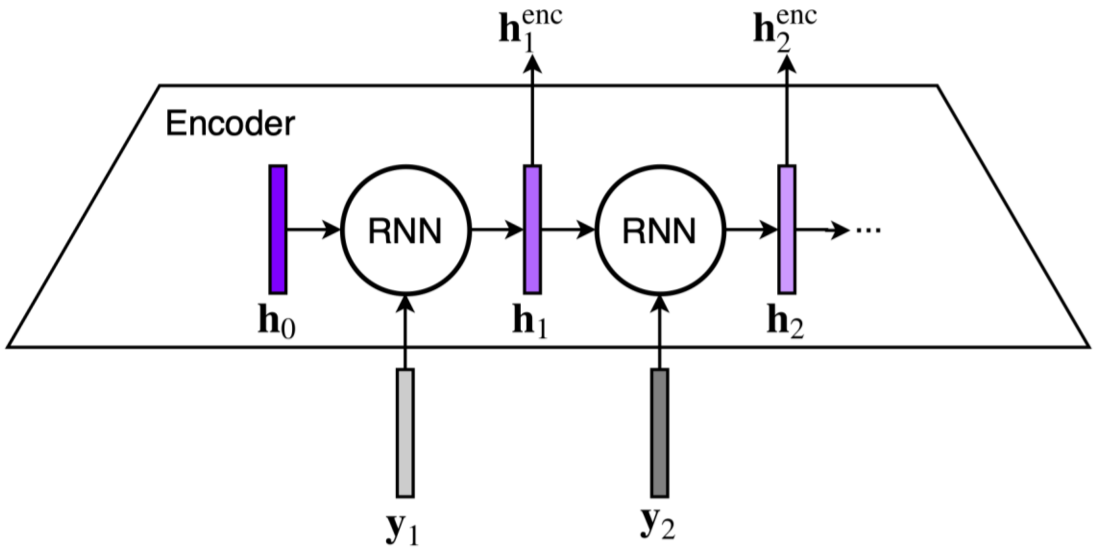
RNN Encoder-Decoder Architectures

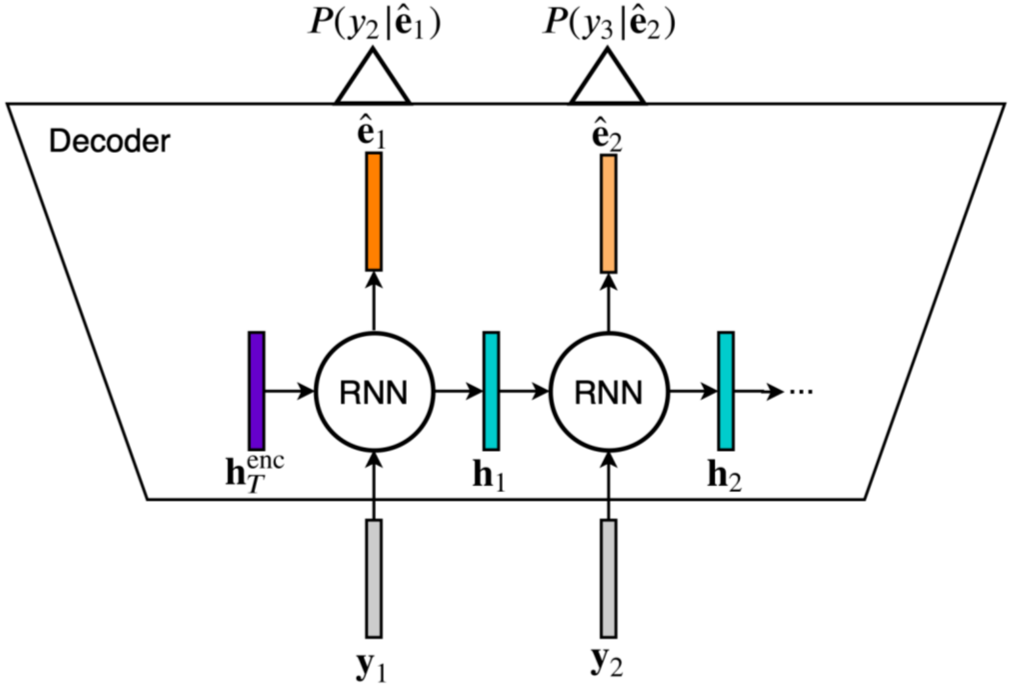
How might we combine the two parts to make an encoder-decoder model?



RNN Encoder-Decoder Architectures

Simplest approach: Use the final hidden state from the encoder to initialize the first hidden state of the decoder.





RNN Encoder-Decoder Architectures

[The, hippopotamus, …

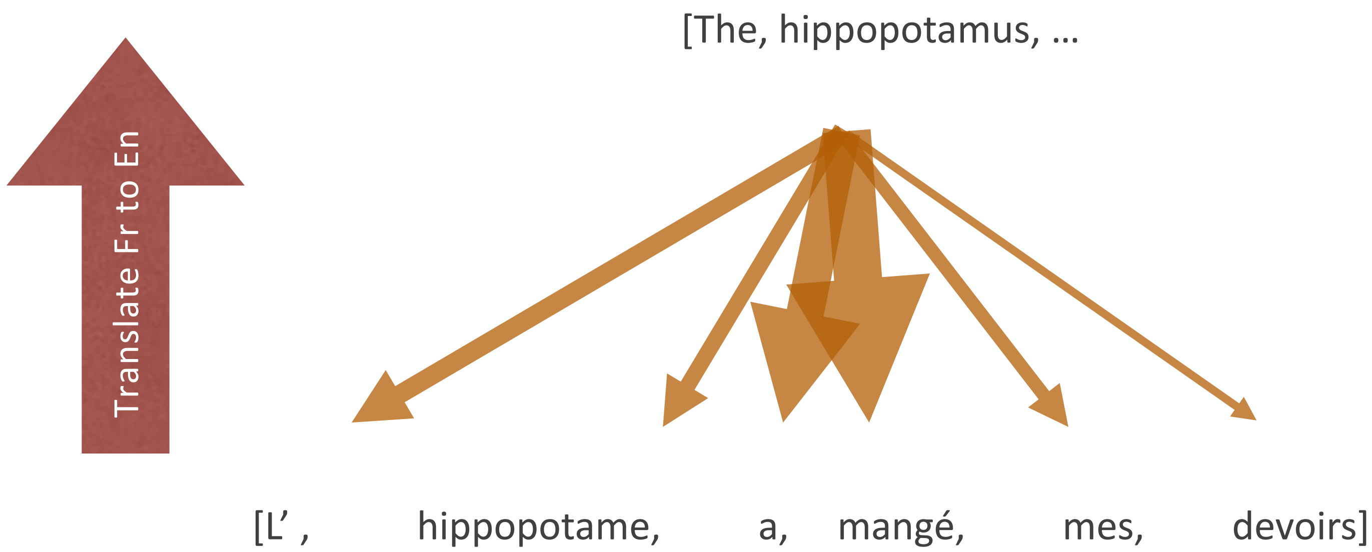
Translate Fr to En

When predicting the next English word, how much weight should the model put on each French word in the source sequence?

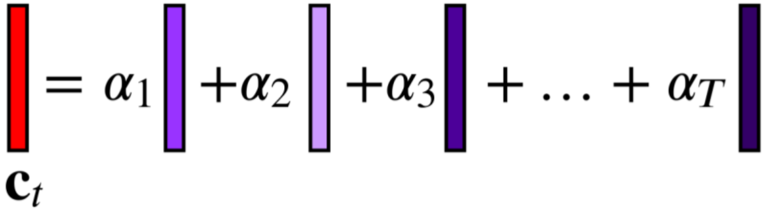
[L’ , hippopotame, a, mangé, mes, devoirs]

Attention

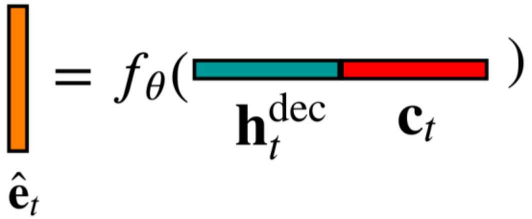
Better approach: an attention mechanism



Compute a linear combination of the encoder hidden states.

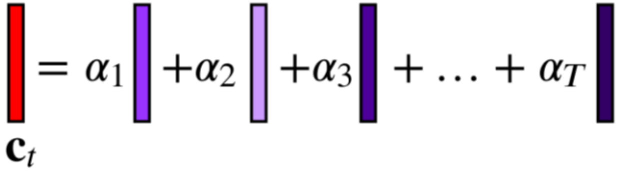


Decoder's prediction at position *t* is based on both the context vector and the hidden state outputted by the RNN at that position.



RNN Encoder-Decoder Architectures

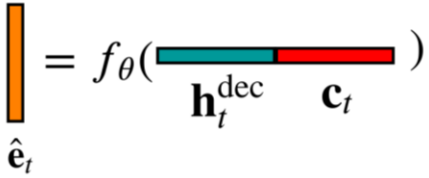
Compute a linear combination of the encoder hidden states.



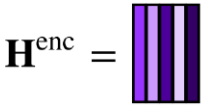
The t[th] context vector is computed as $\mathbf{c}t = \mathbf{H}^{\text{enc}} at$

at[i] = softmax(att_score($\mathbf{h}_t^{\text{dec}}$, $\mathbf{h}i^{\text{enc}}$))

Decoder's prediction at position t is based on both the context vector and the hidden state outputted by the RNN at that position.

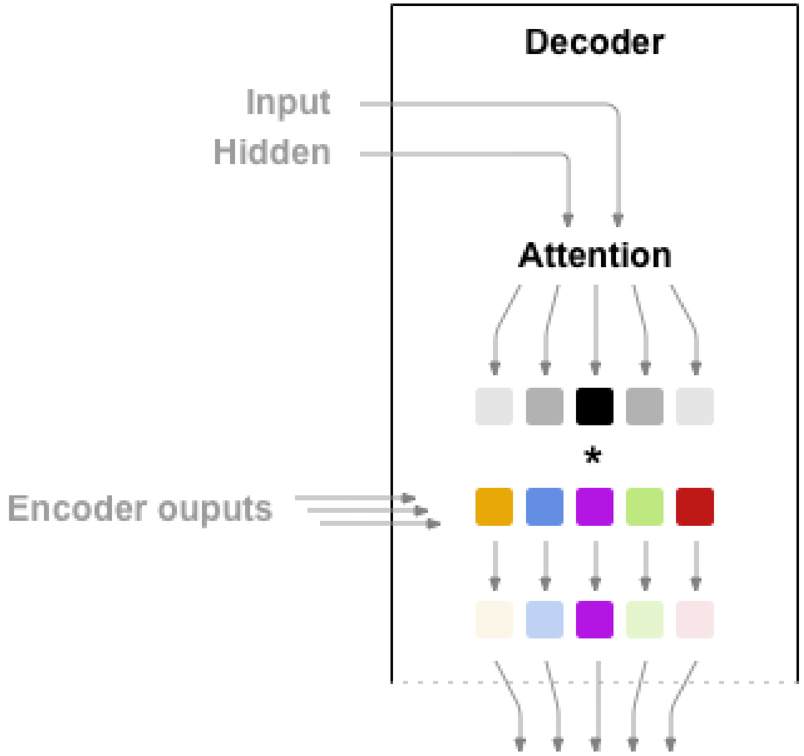


There are a few different options for the attention score:



$$\text{att\_score}(\mathbf{h}_t^{\text{dec}}, \mathbf{h}_i^{\text{enc}}) = \begin{cases} \mathbf{h}_t^{\text{dec}} \cdot \mathbf{h}_{\boldsymbol{i}}^{\text{enc}} & \text{dot product} \\ \mathbf{h}_t^{\text{dec}}\, \mathbf{W}a\, \mathbf{h}_{\boldsymbol{i}}^{\text{enc}} & \text{bilinear function} \\ w_{a1}^{\top} \tanh(\mathbf{W}a\mathbf{2}[\mathbf{h}_t^{\text{dec}}, \mathbf{h}_{\boldsymbol{i}}^{\text{enc}}]) & \text{MLP} \end{cases}$$

Attention Decoder



https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Limitations of Recurrent architecture

Slow to train.

- Can’t be easily parallelized because of the recurrence
- The computation at position t is dependent on first doing the computation at position t-1.

Difficult to access information from many steps back.

- If two tokens are K positions apart, there are K opportunities for knowledge of the first token to be erased from the hidden state before a prediction is made at the position of the second token.

Seq2Seq Output (2017)

R2-D2 carrying some drinks on a tray strapped to his back passes Yoda who uses his force powers to hog the drinks

Expected:

Obi Wan and Anakin are drinking happily when Chewbacca takes a Polaroid picture of Anakin and Obi Wan

Predicted:

Can this block gives him the advantage to personally run around with a large stick of cheese

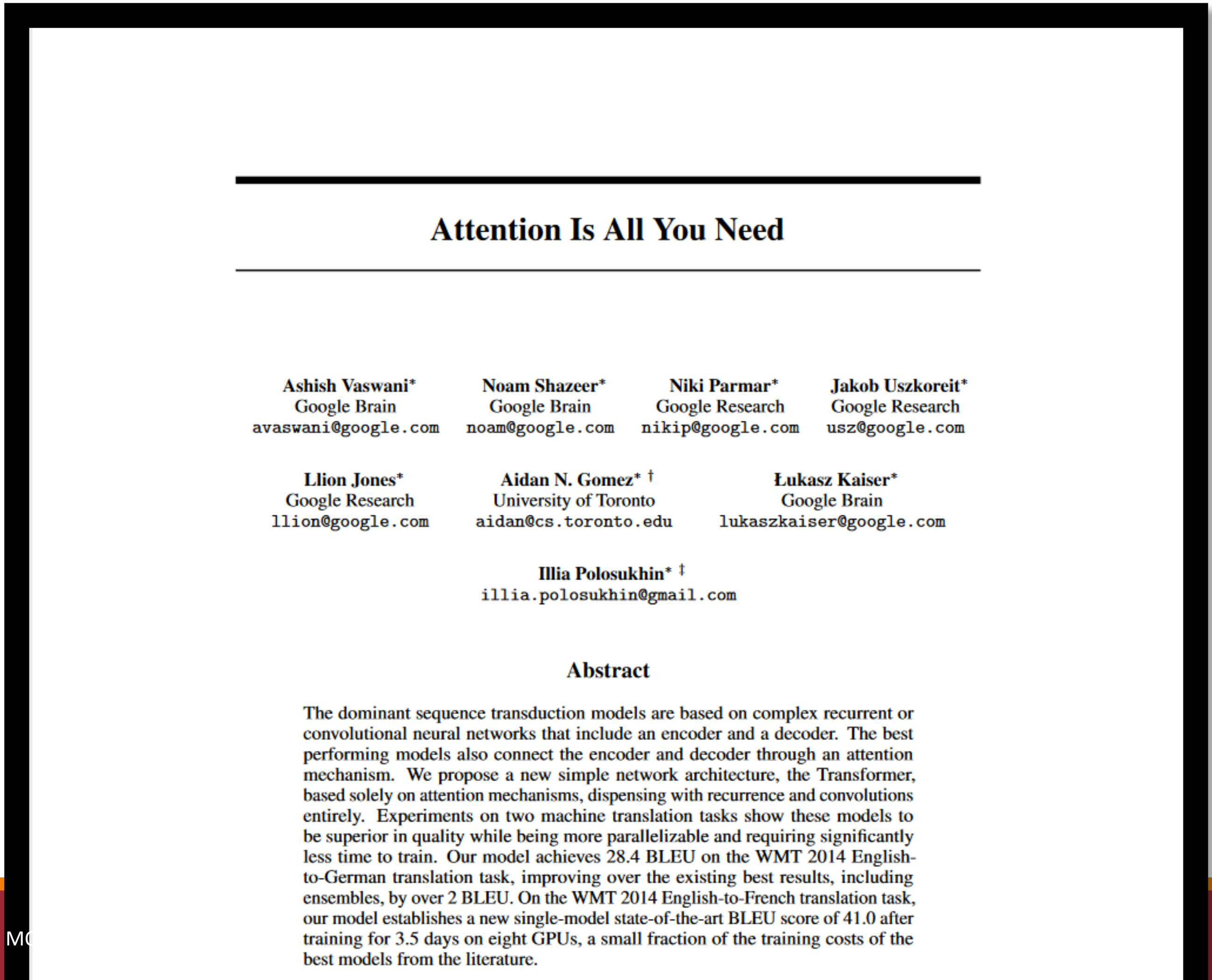
Summary for *Lego Star Wars: Revenge of the Brick*. From the original presentation for **Martin, Lara J.**, Prithviraj Ammanabrolu, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. "Event Representations for Automated Story Generation with Deep Neural Nets." *KDD 2017 Workshop on Machine Learning for Creativity*, 2017.

Neural Language Model Timeline

A Neural Probabilistic Language Model
Yoshua Bengio, Réjean Ducharme, Pascal Vincent, & Christian Janvin

Sequence to Sequence Learning with Neural Networks
Ilya Sutskever, Oriol Vinyals, & Quoc V. Le

Attention Is All You Need
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, & Illia Polosukhin



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Transformers

The Transformer is a **non-recurrent** non-convolutional (feed-forward) neural network designed for language understanding

- introduces self-attention in addition to encoder-decoder attention