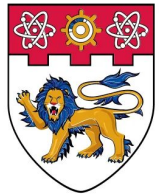


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Natural Language Processing

SC4002 / CE4045 / CZ4045

by Wang Wenya

Email: wangwy@ntu.edu.sg

Contents adapted from Dr. Joty Shafiq's notes

Click here for Lecture 3

Modules we will cover

ML & DL

Introduction to machine/deep learning

Word

Word vectors, language modeling

Transformer

Attention mechanism, encoder/decoder

Sequence

Sequence modeling, seq2seq learning

Pretraining

Masking, natural language generation

Prompting

Prompts, in-context learning

Outline for today

01 Recurrent neural networks

02 Long Short-Term Memory

03 Sequence modeling

04 Sequence-to-sequence learning

Neural Language Model (Revisit)

Source: stanford 224n

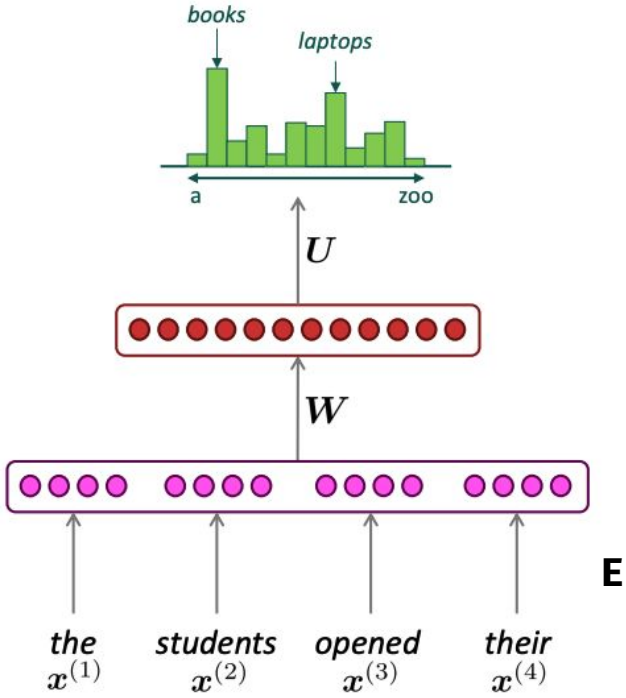
Improvements over *n*-gram LM:

- No sparsity problem
- Don't need to store all observed *n*-grams

Remaining **problems**:

- Fixed window is too small
- Enlarging window enlarges W
- Window can never be large enough!
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W. No symmetry in how the inputs are processed.

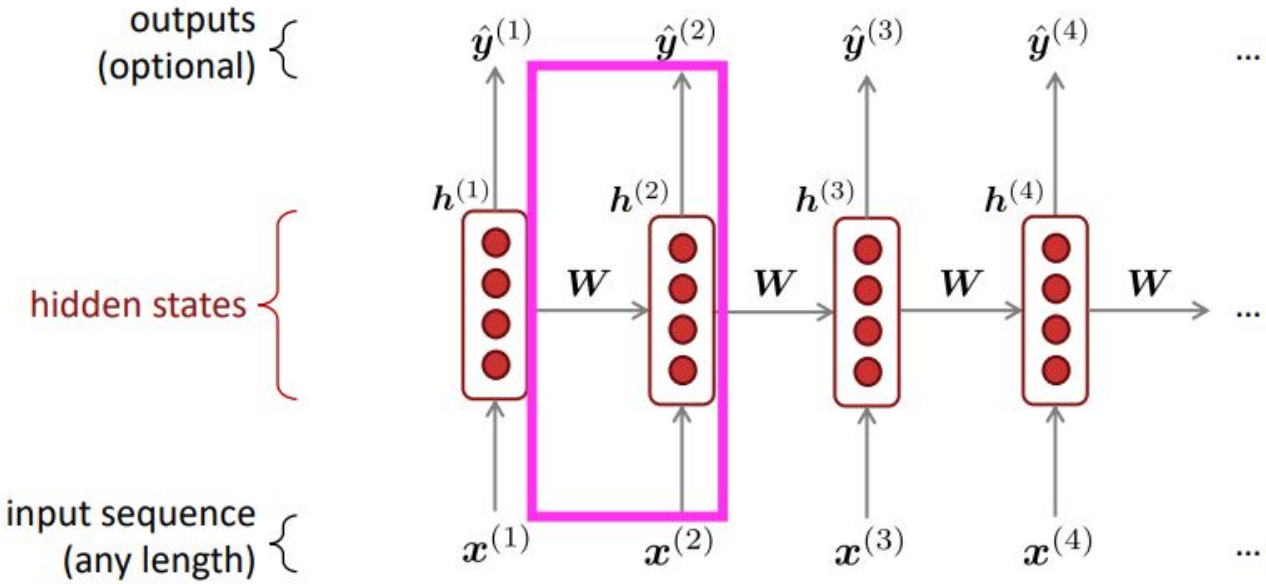
We need a neural architecture that can process *any length input*



Recurrent Neural Networks (RNNs)

Source: stanford 224n

- Main idea: Apply the same weights repeatedly (recurrently)



RNN Language Models

Source: stanford 224n

hidden states

$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$

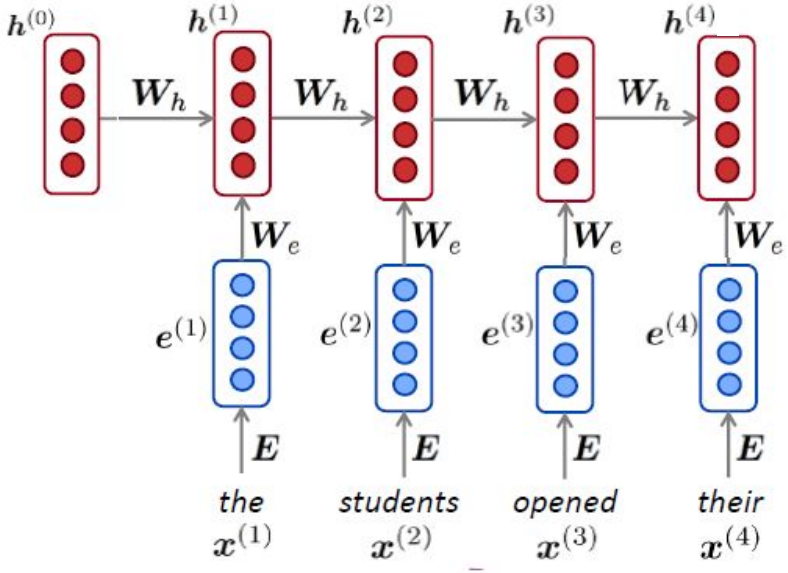
$h^{(0)}$ is the initial hidden state

word embeddings

$e^{(t)} = Ex^{(t)}$

words / one-hot vectors

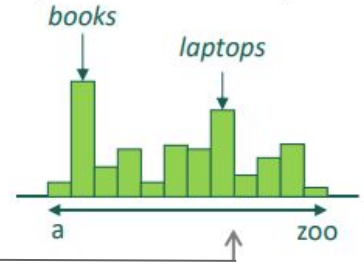
$x^{(t)} \in \mathbb{R}^{|V|}$



The input sequence can be of arbitrary length.

RNN Language Models

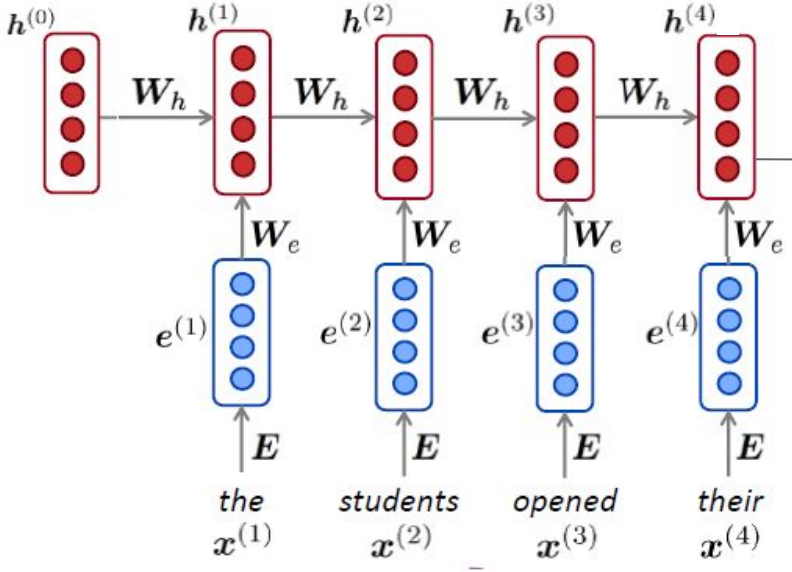
$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$



hidden states

$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$

$\boldsymbol{h}^{(0)}$ is the initial hidden state



word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

The input sequence can be of arbitrary length.

Training a RNN-LM

Source: stanford 224n

- Get a big corpus of text which is a sequence of words $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ for *every step t*.
  - i.e. predict probability dist of *every word*, given words so far

- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

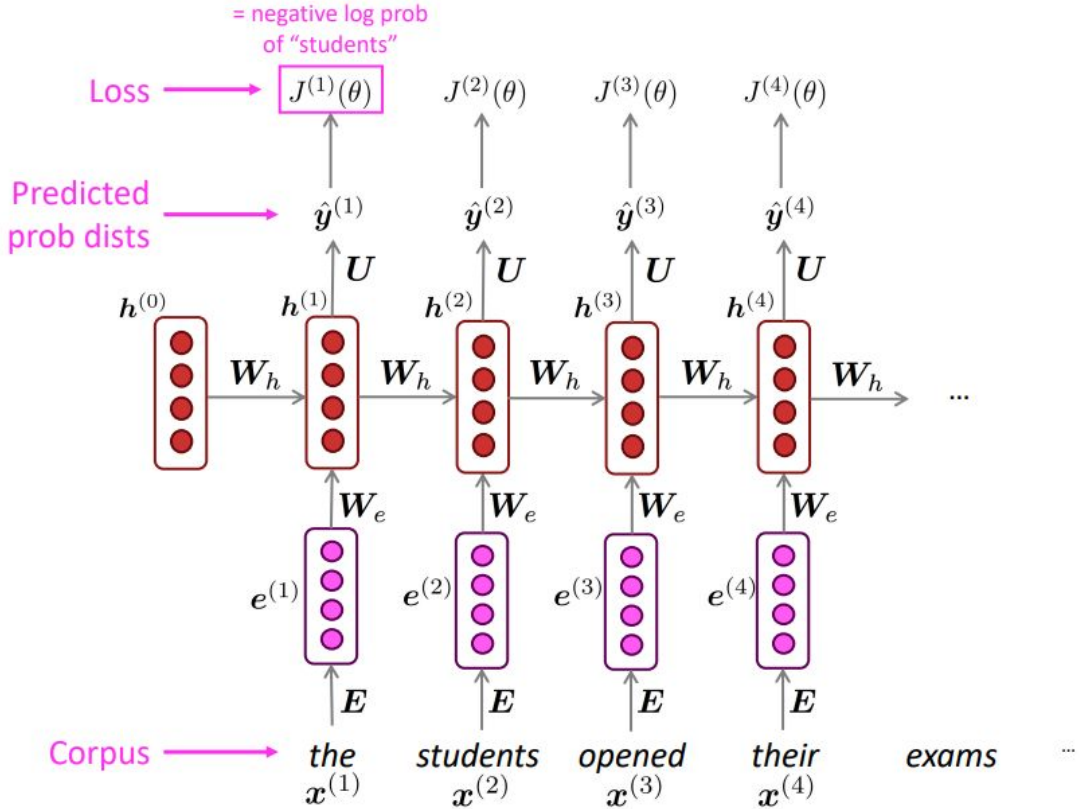
$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

- Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

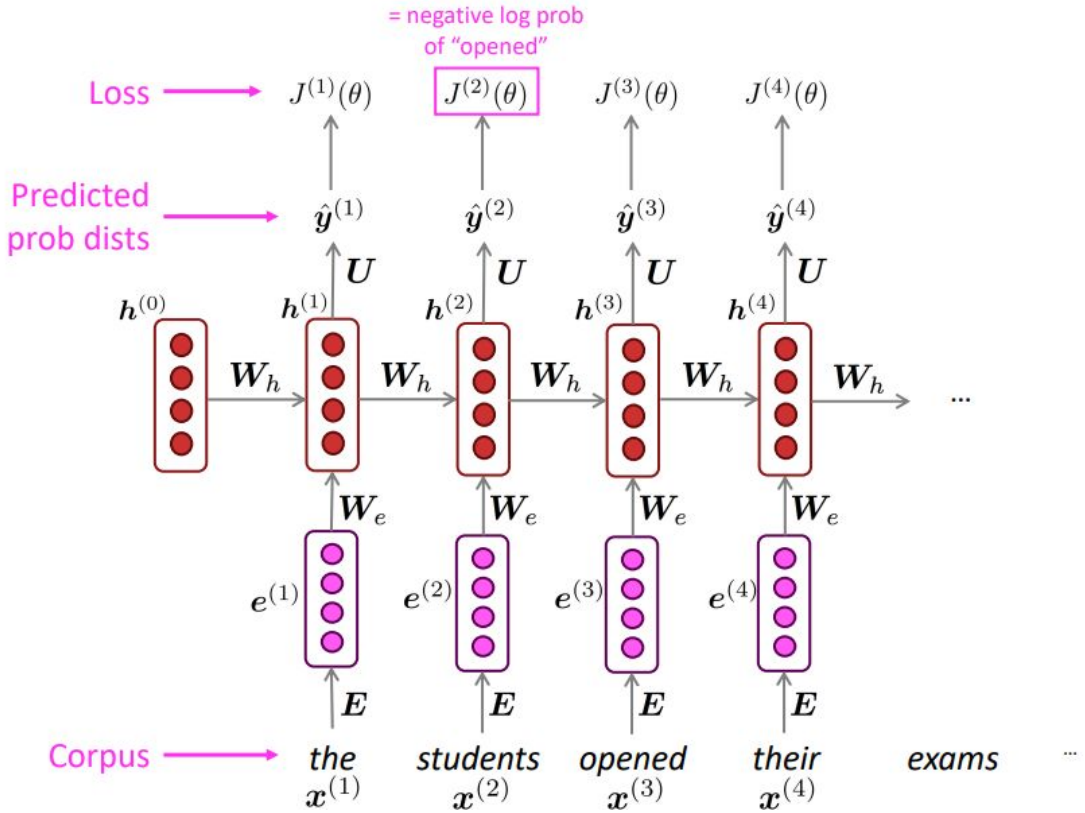
Training a RNN-LM

Source: stanford 224n



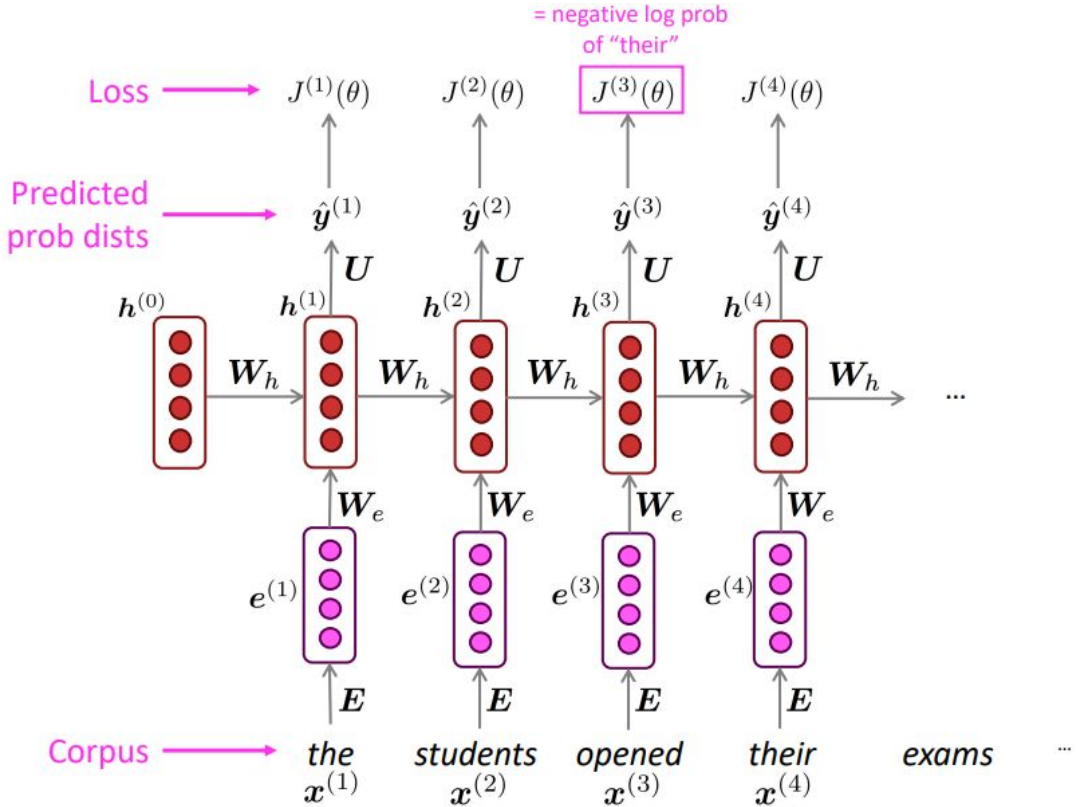
Training a RNN-LM

Source: stanford 224n



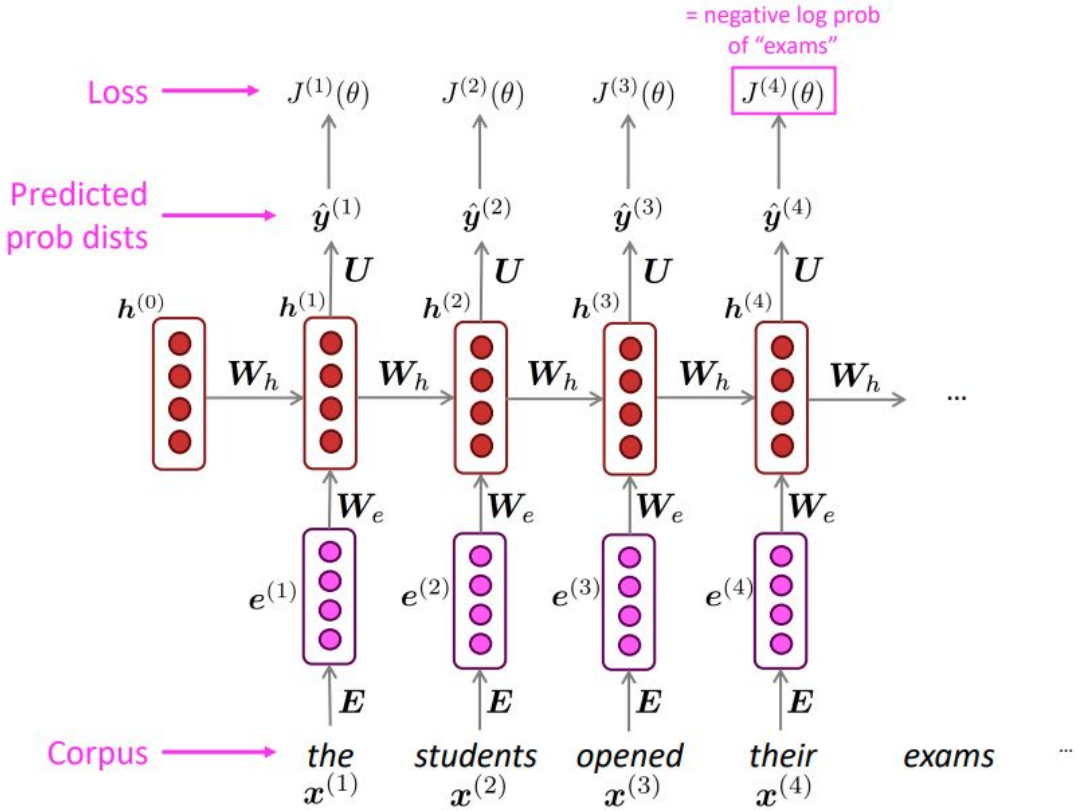
Training a RNN-LM

Source: stanford 224n



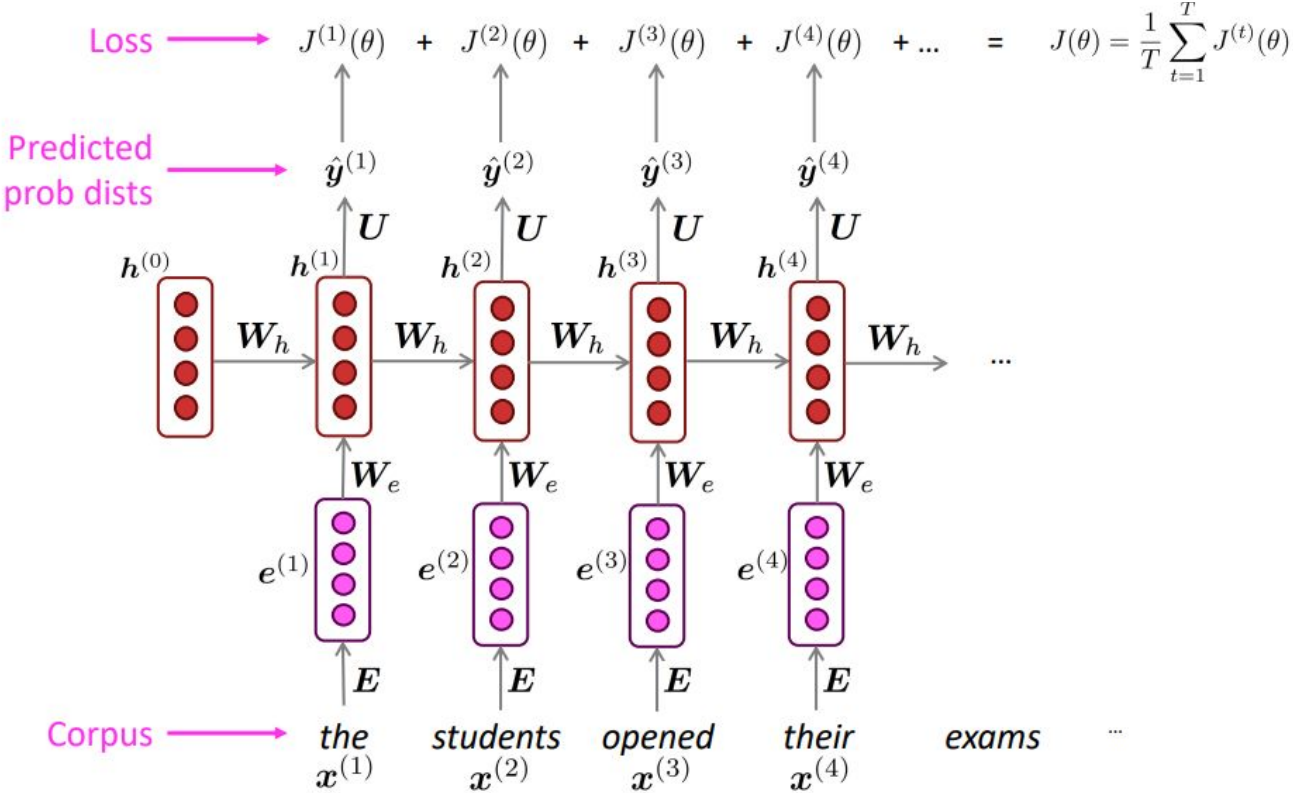
Training a RNN-LM

Source: stanford 224n



Training a RNN-LM

Source: stanford 224n



Training a RNN-LM

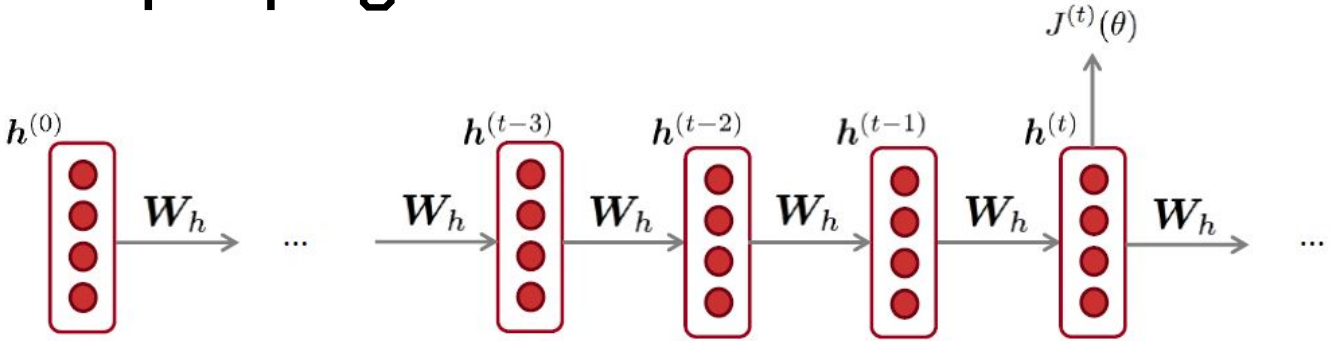
- However, computing loss and gradients across entire corpus $x^{(1)}, \ldots, x^{(T)}$ is too expensive (memory-wise)

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta)$$

- In practice, consider $x^{(1)}, \ldots, x^{(T)}$ as a sentence/document.
- In mini-batch GD, we compute loss and gradient for batches of sentences/documents and update.

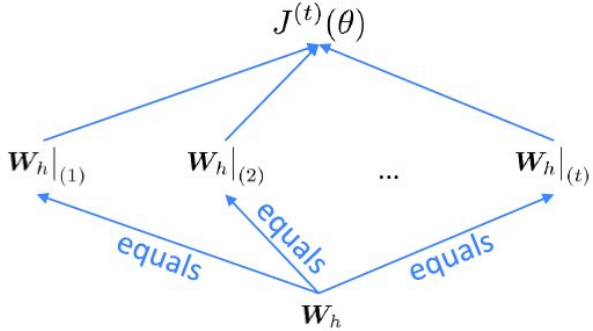
Backpropagation for RNNs

Source: stanford 224n



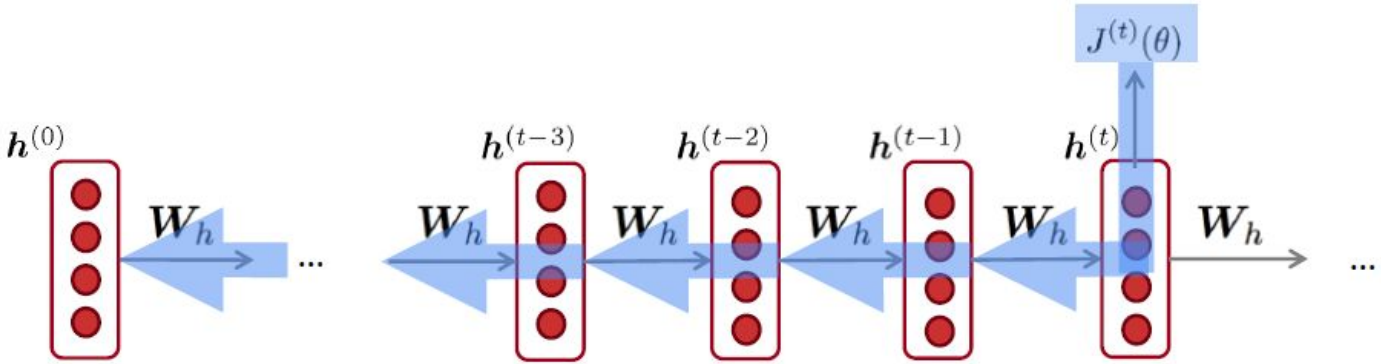
The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)}$$



Backpropagation for RNNs

Source: stanford 224n

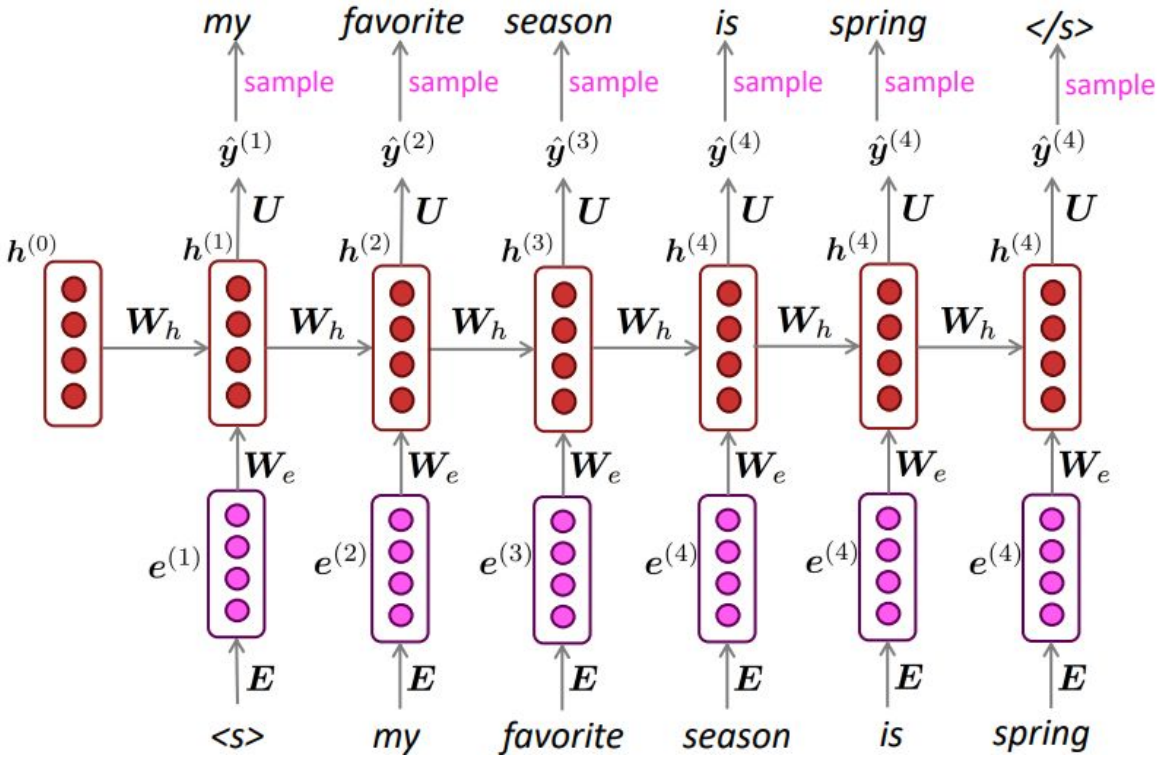


Backpropagate over timesteps i=t,...,0, summing gradients as you go.

This algorithm is called **"backpropagation through time"**

Generating Text with an RNN-LM

Source: stanford 224n



Just like an n-gram Language Model, you can use a RNN Language Model to generate text by repeated sampling. Sampled output becomes next step's input.

Fun with RNN-LM Generations

You can train an RNN-LM on any kind of text, then generate text in that style.

Good morning. One of the borders will be able to continue to be here today. We have to say that the partnership was a partnership with the American people and the street continually progress that is a process and distant lasting peace and support that they were supporting the work of concern in the world. They were in the streets and communities that could have to provide steps to the people of the United States and Afghanistan. In the streets — the final decade of the country that will include the people of the United States of America. Now, humanitarian crisis has already rightly achieved the first American future in the same financial crisis that they can find reason to invest in the world.
Thank you very much. God bless you. God bless you. Thank you.

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Fun with RNN-LM Generations

You can train an RNN-LM on any kind of text, then generate text in that style.

Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.
"I'm afraid I've definitely been suspended from power, no chance — indeed?" said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.
Hermione yelled. The party must be thrown by Krum, of course.
Harry collected fingers once more, with Malfoy. "Why, didn't she never tell me. ..." She vanished.
And then, Ron, Harry noticed, was nearly right.
"Now, be off," said Sirius, "I can't trace a new voice."

https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

Fun with RNN-LM Generations

You can train an RNN-LM on any kind of text, then generate text in that style.

Title: CARAMEL CORN GARLIC BEEF
Categories: Soups, Desserts
Yield: 10 Servings

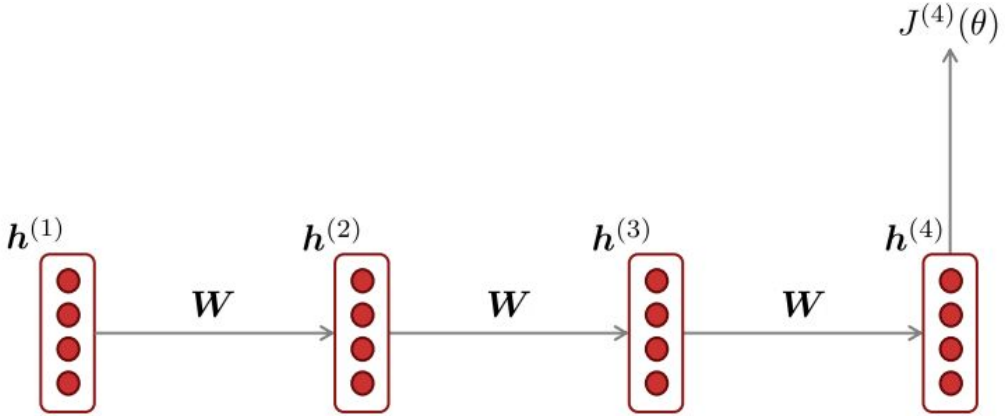
2 tb Parmesan cheese, ground
1/4 ts Ground cloves
-- diced
1 ts Cayenne pepper

Cook it with the batter. Set aside to cool. Remove the peanut oil in a small saucepan and pour into the margarine until they are soft. Stir in a a mixer (dough). Add the chestnuts, beaten egg whites, oil, and salt and brown sugar and sugar; stir onto the boqtly brown it.

The recipe from an oiled by fried and can. Beans, by Judil Cookbook, Source: Pintore, October, by Chocolates, Breammons of Jozen, Empt.com

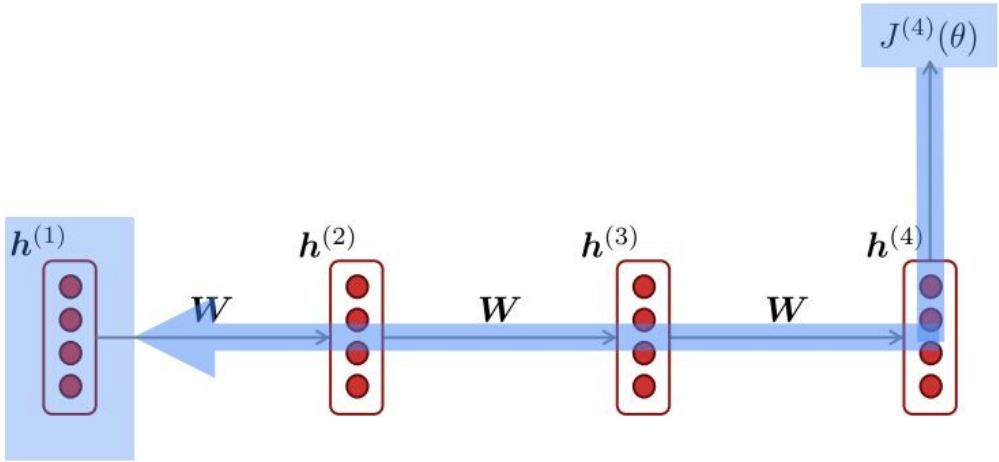
https://gist.github.com/nylki/1efbaa36635956d35bcc

Problems with RNNs – Vanishing Gradients



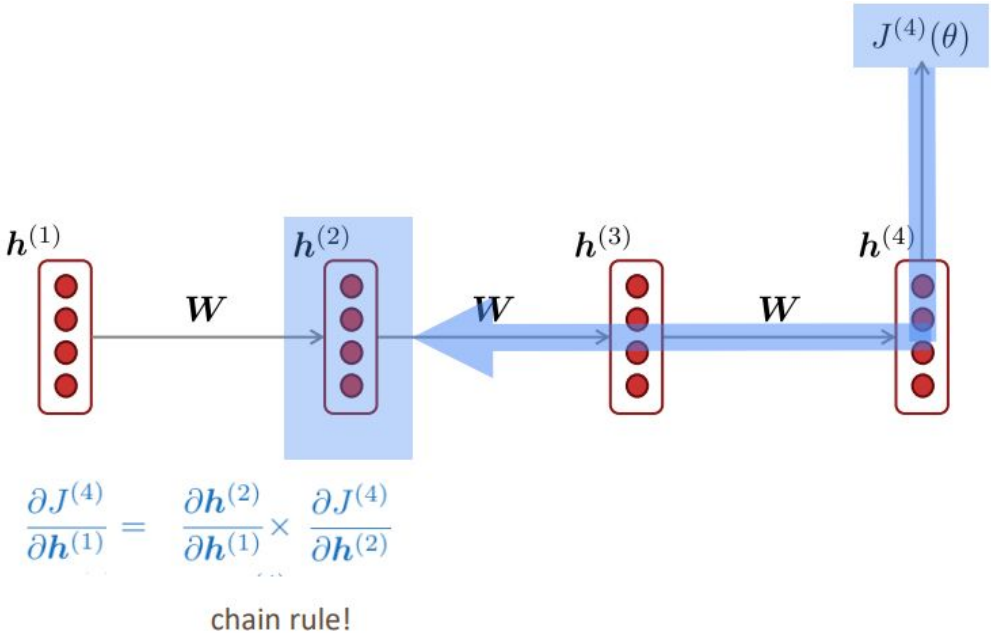
Source: stanford 224n

Problems with RNNs – Vanishing Gradients



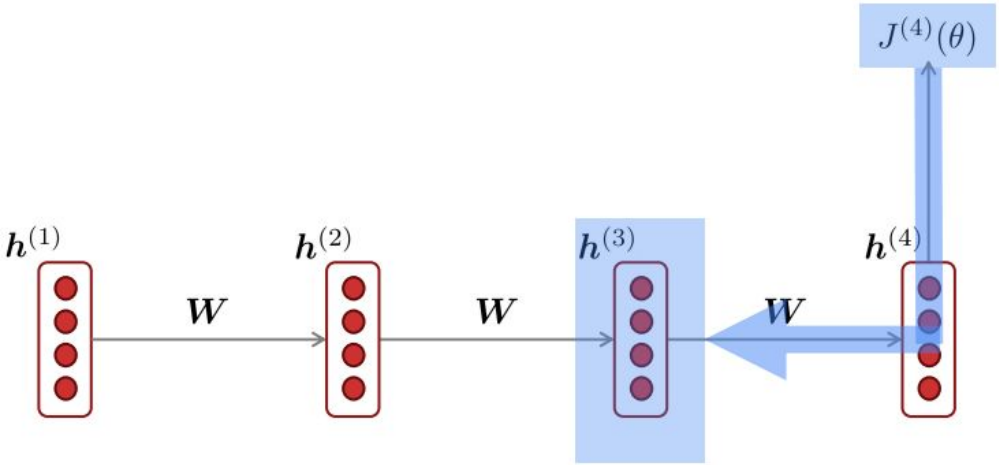
Source: stanford 224n

Problems with RNNs – Vanishing Gradients



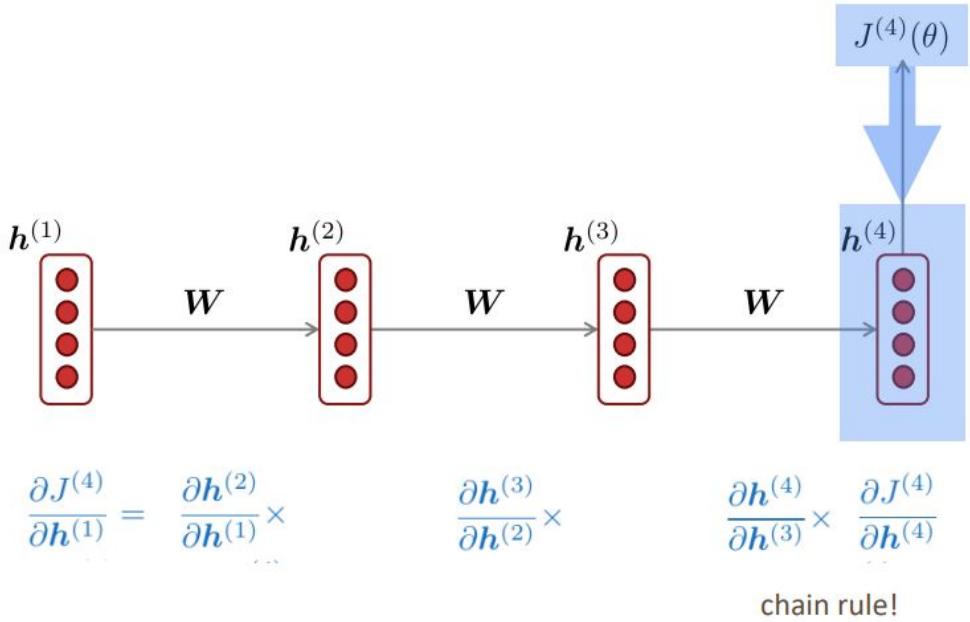
Source: stanford 224n

Problems with RNNs – Vanishing Gradients



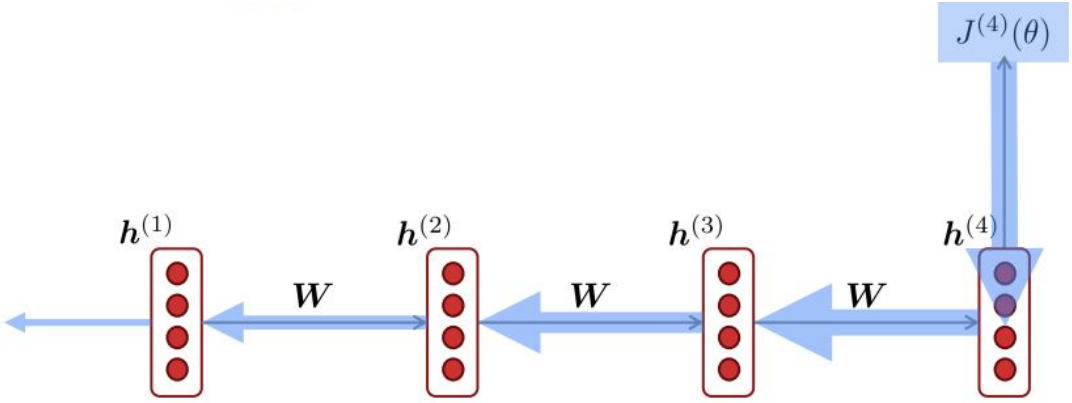
Source: stanford 224n

Problems with RNNs – Vanishing Gradients



Source: stanford 224n

Problems with RNNs – Vanishing Gradients

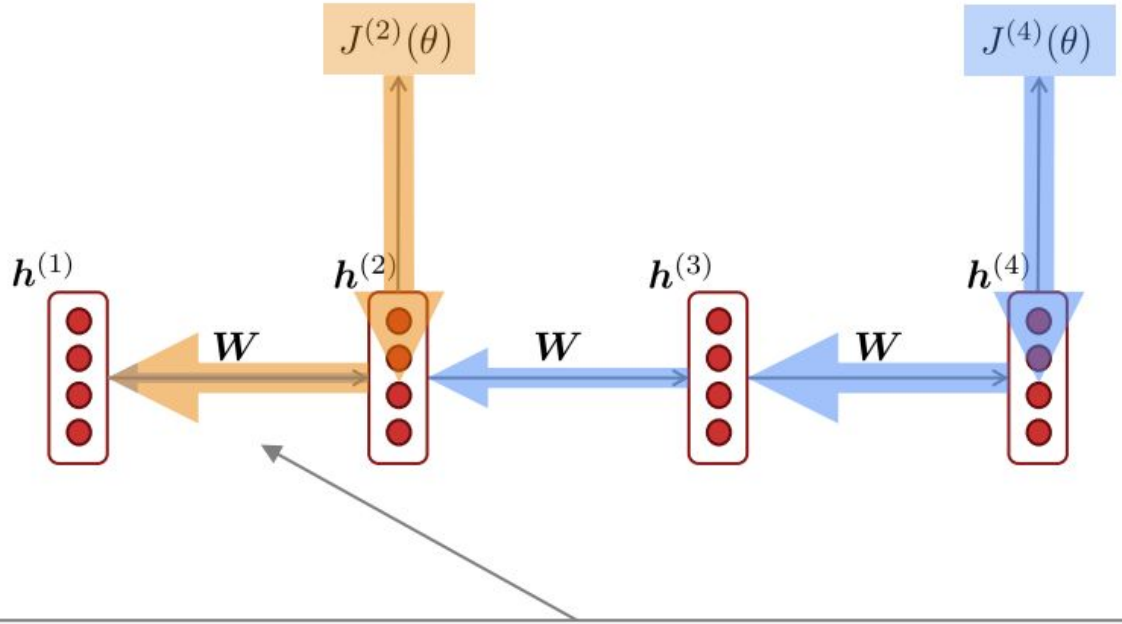


$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Problems with RNNs – Vanishing Gradients



Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

Problems with RNNs – Vanishing Gradients

- **LM task:** When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____
- To learn from this training example, the RNN-LM needs to model the dependency between "tickets" on the 7th step and the target word "tickets" at the end.
- But if gradient is small, the model can't learn this dependency

Problems with RNNs – Vanishing Gradients

- **LM task:** *The writer of the books ___* → *is* / *are*
- **Correct answer**: *The writer of the books is planning a sequel*
- **Syntactic recency:** *The writer of the books is* (correct)
- **Sequential recency:** *The writer of the books are* (incorrect)

Due to vanishing gradient, RNN-LMs are better at learning from sequential recency than syntactic recency, so they make this type of error more often [Linzen et al 2016]

Linzen et al, 2016. "Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies"

RNN Language Models

Advantages of RNNs

- Can process any length input. Computation for step t can (in theory) use information from many steps back.
- Model size doesn't increase for longer input context.
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

Disadvantages of RNNs

- Recurrent computation is slow.
- In practice, difficult to access information from many steps back.

Outline for today

01 Recurrent neural networks

02 Long Short-Term Memory

03 Sequence modeling

04 Sequence-to-sequence learning

Tackling Vanishing Gradients

- The main problem is that it's too difficult for the vanilla RNNs to learn to preserve information over many timesteps. The "same" hidden state is constantly updated

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}\right)$$

- How about a RNN with separate memory?

Long Short-Term Memory (LSTM)

- At step t, there is a hidden state $\mathbf{h}^{(t)}$ and a cell state $\mathbf{c}^{(t)}$
  - Both are vectors of length n
  - The cell stores long-term information
  - The LSTM can read, erase, and write information from the cell
- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors of length n
  - At each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between
  - The gates are dynamic: their value is computed based on the current context

Long Short-Term Memory (LSTM)

Source: stanford 224n

- Given current input and previous hidden states, compute gates

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

Sigmoid function: all gate values are between 0 and 1

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

Long Short-Term Memory (LSTM)

Source: stanford 224n

- Now compute how much to forget, update and output

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

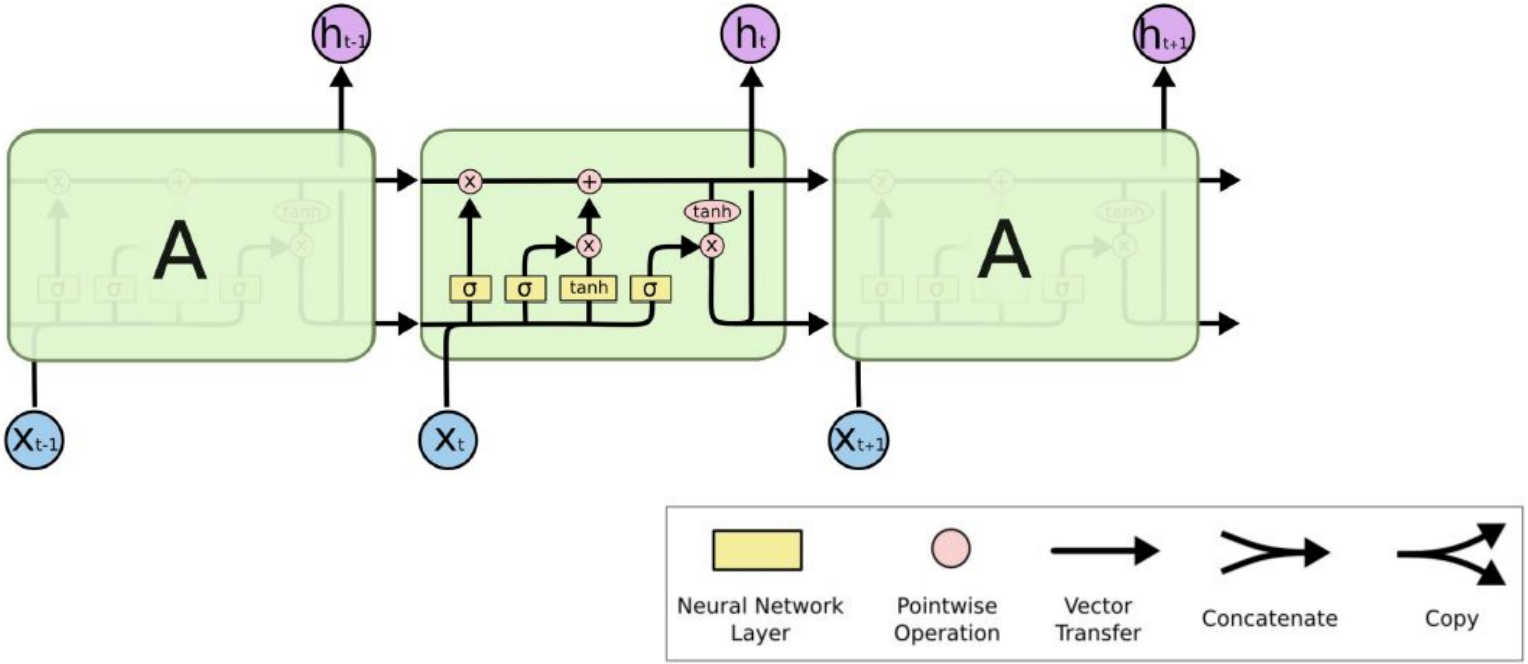
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

Gates are applied using element-wise product

All of them are the same length!

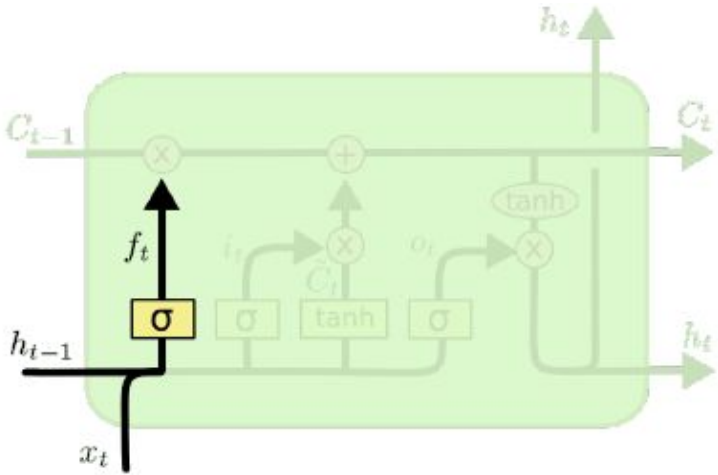
Long Short-Term Memory (Optional)



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (Optional)

Compute forget gate

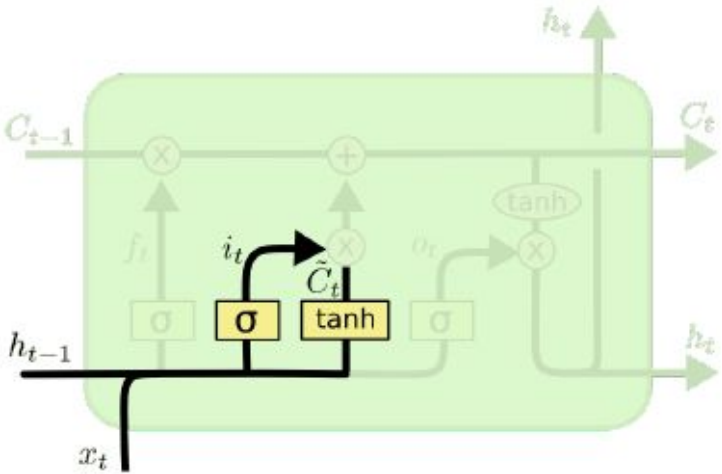


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (Optional)

Compute current cell state and input gate



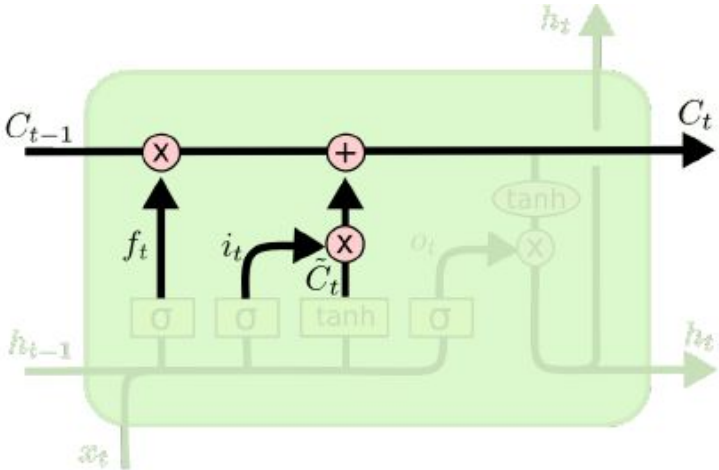
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (Optional)

Forget (erase) something and write (update) something new

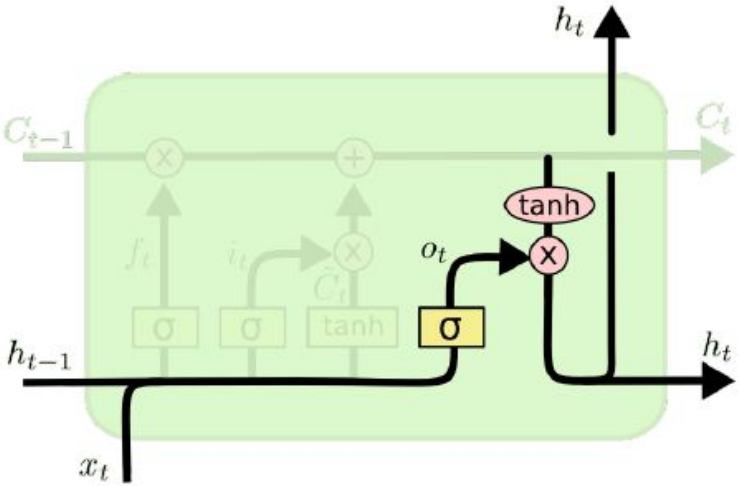


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (Optional)

Compute output gate and output state

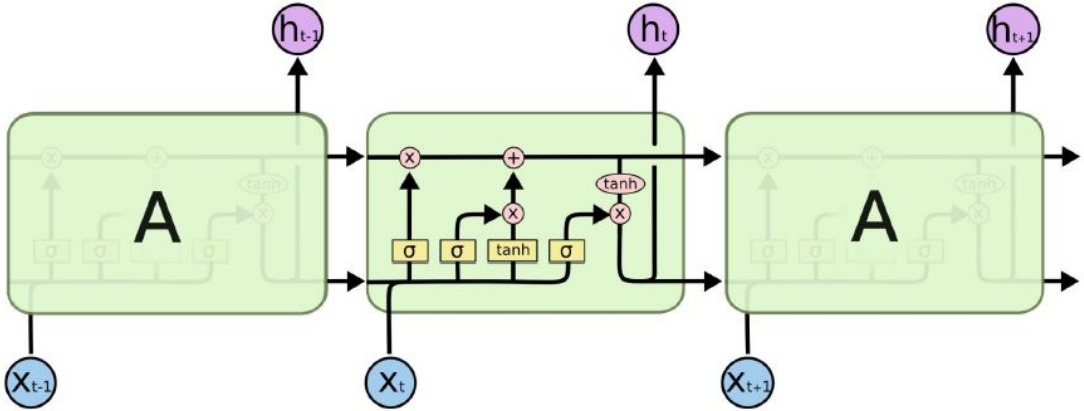


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

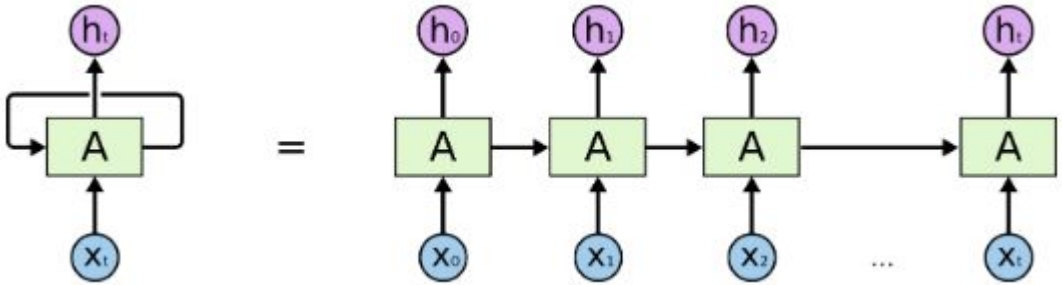
$$h_t = o_t * \tanh\left(C_t\right)$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



RNN



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

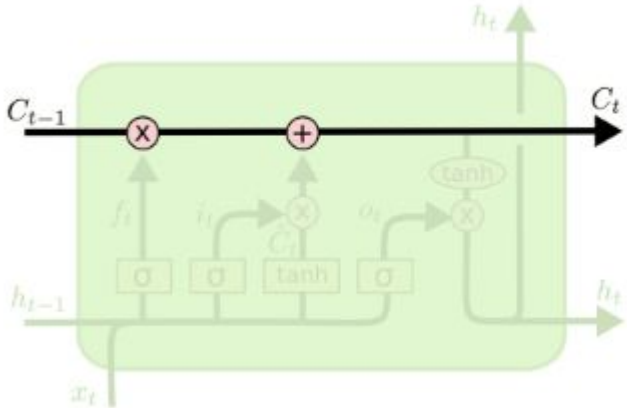
Why LSTM Works

Source: stanford 224n

- The LSTM architecture makes it easier for the RNN to preserve information over many time steps.
  - If the forget gate is set to remember everything on every time step, then the info in the cell is preserved indefinitely.
  - In practice, LSTM uses adaptive gates, which are also learned (i.e., using their respective parameters)
- LSTM however doesn't guarantee that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies.

Long Short-Term Memory (LSTM)

The top line shows a simple linear interactions among cells



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Success of LSTM

Source: stanford 224n

- In 2013-2015, LSTMs started achieving state-of-the-art results
  - Successful tasks: handwriting recognition, speech recognition, machine translation, parsing, image captioning.
  - Became the dominant approach
- In recent years (2017-2020), other approaches (e.g. Transformers) have become more dominant for certain tasks. For example in **WMT** (a MT conference + competition):
  - WMT 2016, the summary report contains "RNN" 44 times
  - In WMT 2018, the report contains "RNN" 9 times and "Transformer" 63 times

Comparison on Perplexity

https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/

n-gram model →

Increasingly complex RNNs ↓

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better) ↓

Vanishing/Exploding Gradient

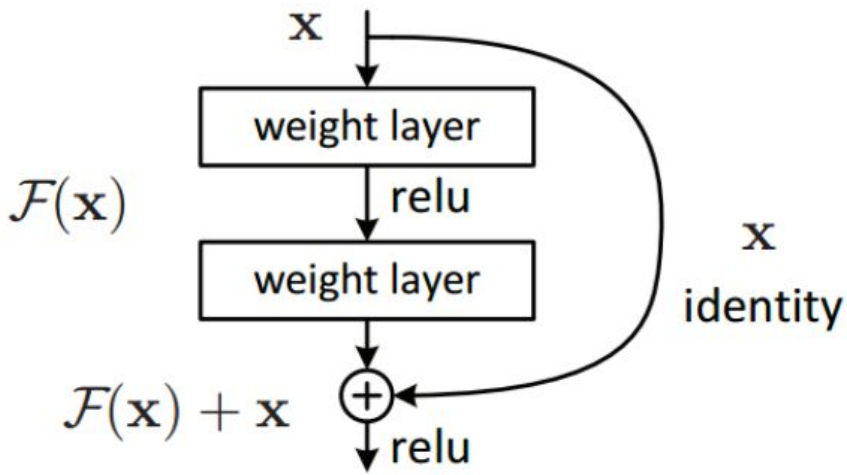
Source: stanford 224n

- Vanishing/Exploding gradient problems are not specific to only RNNs
- Can happen in all neural deep architectures (including feed-forward and convolutional; RNNs are deep in time).
  - Due to chain rule and/or choice of nonlinearity function, gradient can become vanishingly small as it backpropagates.
  - Thus lower layers are learnt very slowly (hard to train)
- Solution: lots of new deep feedforward/convolutional architectures add more direct connections (thus allowing the gradient to flow)

Solution: Residual Connections

Source: stanford 224n

- Solution: lots of new deep feedforward/convolutional architectures add more direct connections (thus allowing the gradient to flow)
  - Residual connections aka "ResNet"
  - Also known as skip-connections
  - The identity connection preserves information by default
  - This makes deep networks much easier to train



https://arxiv.org/pdf/1512.03385.pdf

Outline for today

01 Recurrent neural networks

02 Long Short-Term Memory

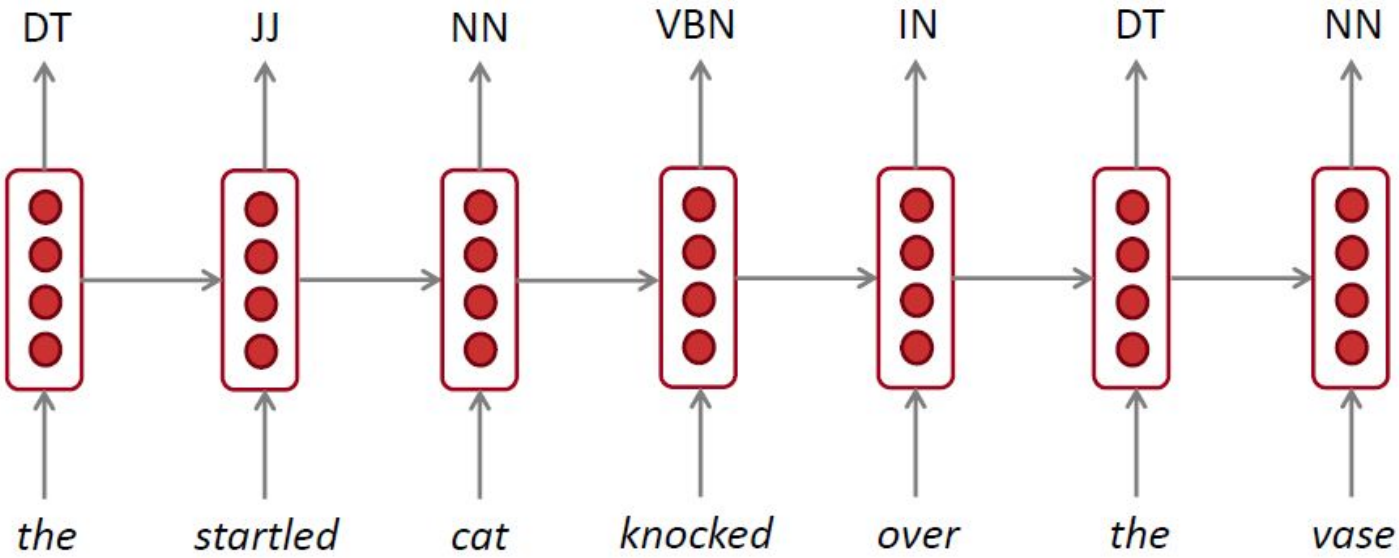
03 Sequence modeling

04 Sequence-to-sequence learning

Sequence Tagging with RNNs

Source: stanford 224n

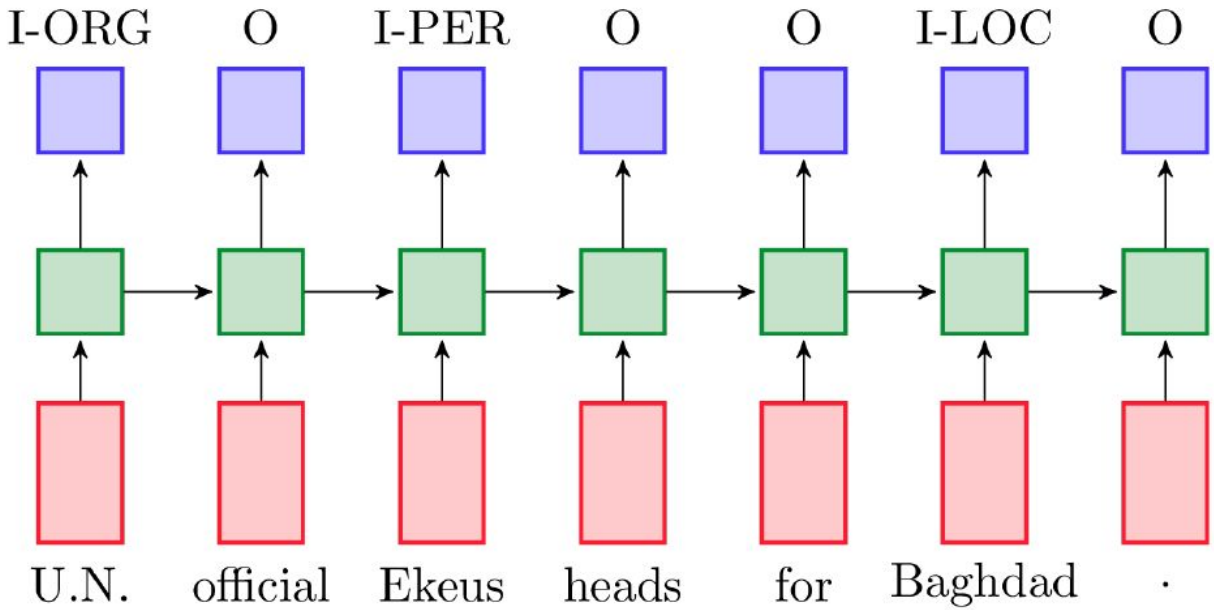
- POS Tagging



Sequence Tagging with RNNs

Source: stanford 224n

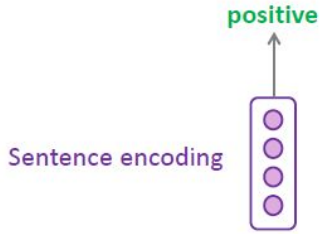
- NER Tagging



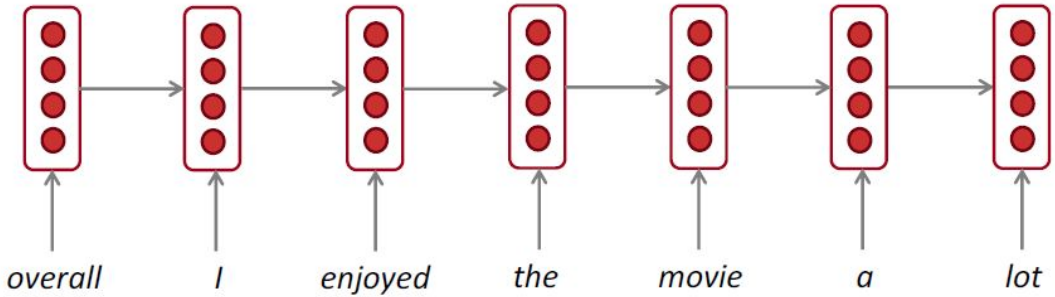
Sequence Classification with RNNs

Source: stanford 224n

- Sentiment Classification:



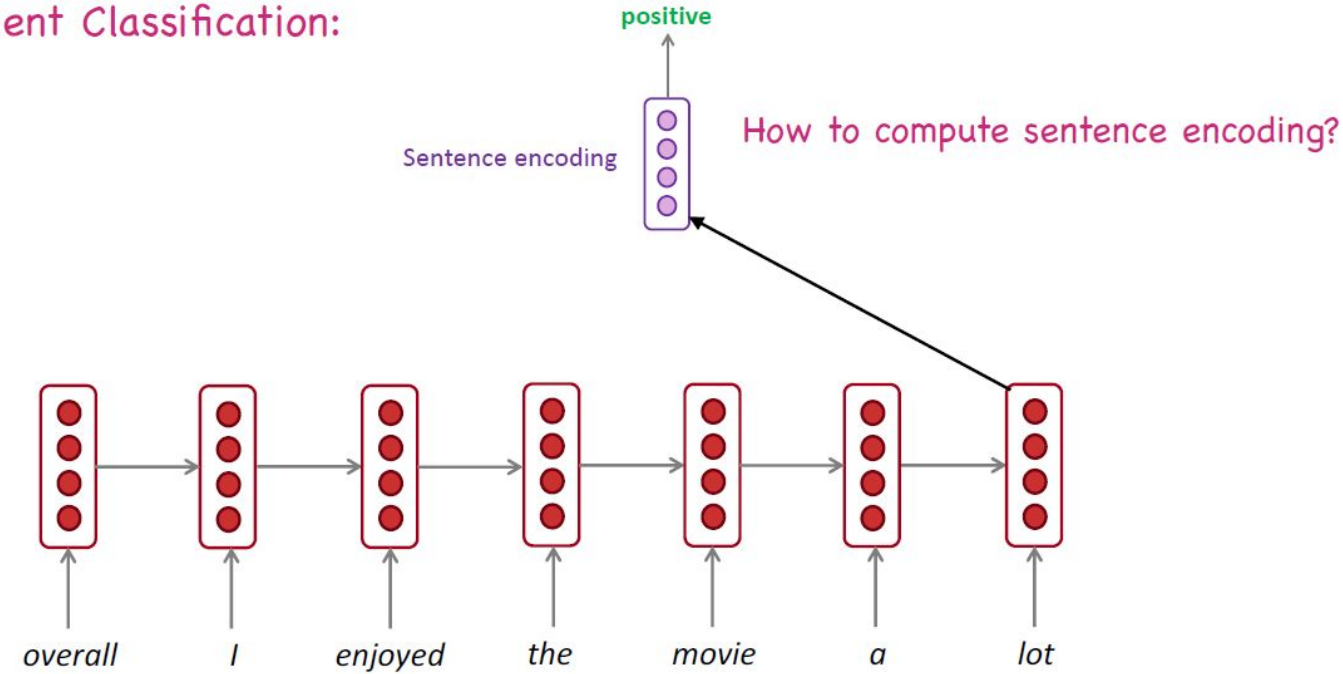
How to compute sentence encoding?



Sequence Classification with RNNs

Source: stanford 224n

- Sentiment Classification:

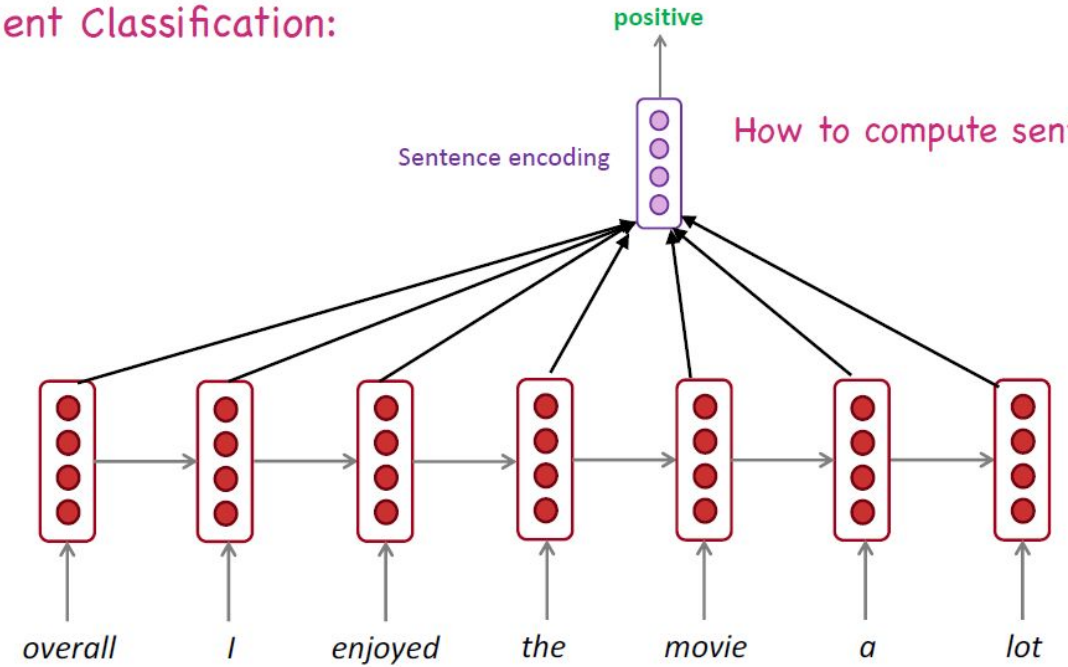


Use the final hidden state as the summary of the sentence

Sequence Classification with RNNs

Source: stanford 224n

- Sentiment Classification:

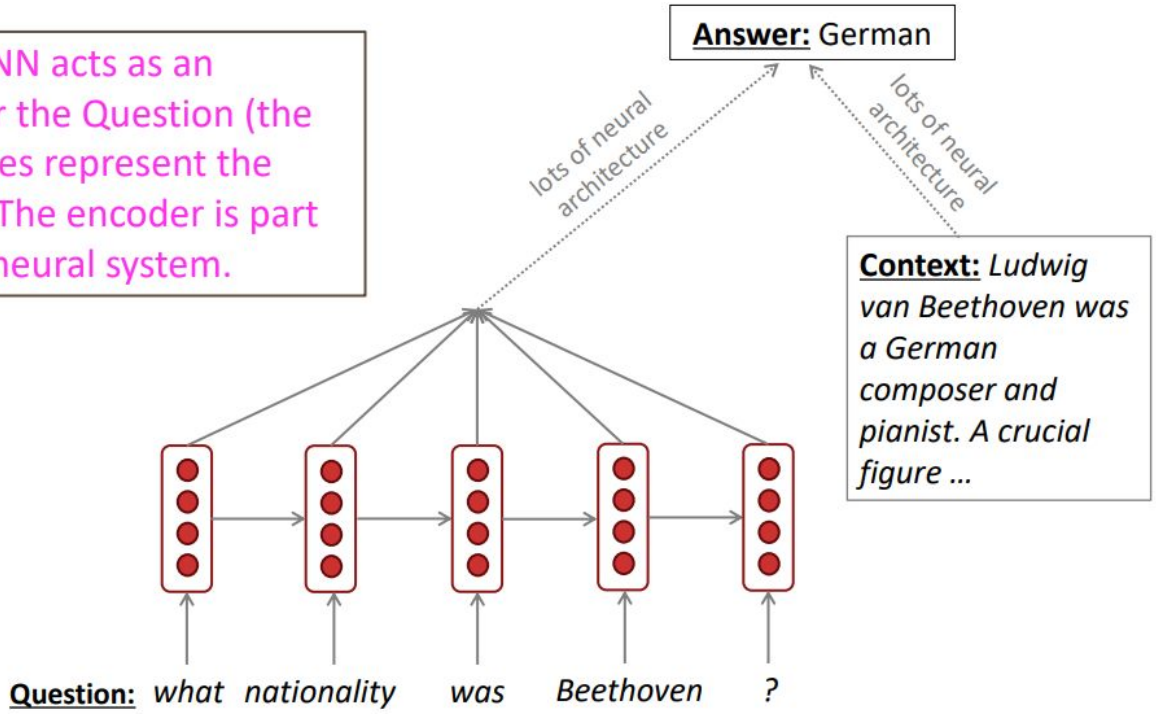


Use max/average pooling as the summary of the sentence

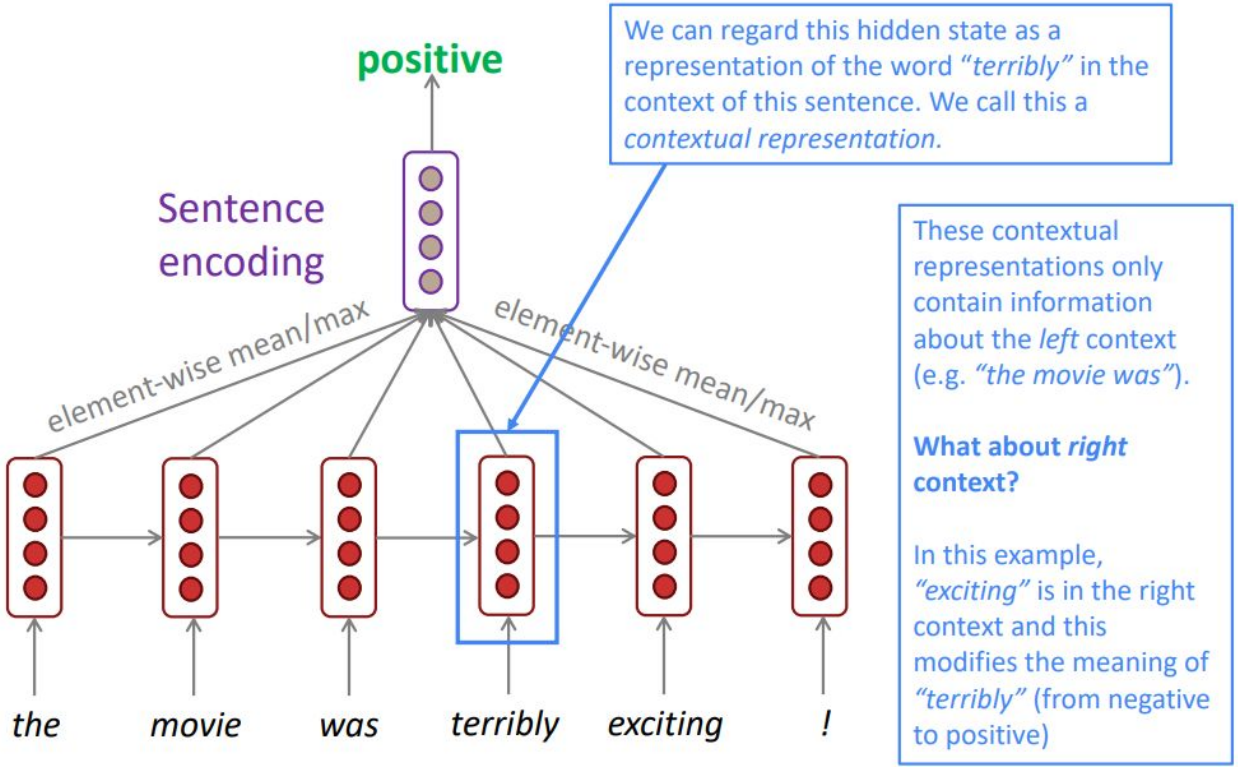
Sequence Classification with RNNs

Source: stanford 224n

Here the RNN acts as an **encoder** for the Question (the hidden states represent the Question). The encoder is part of a larger neural system.



Other Variants of RNNs – Bidirectional RNNs



We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation*.

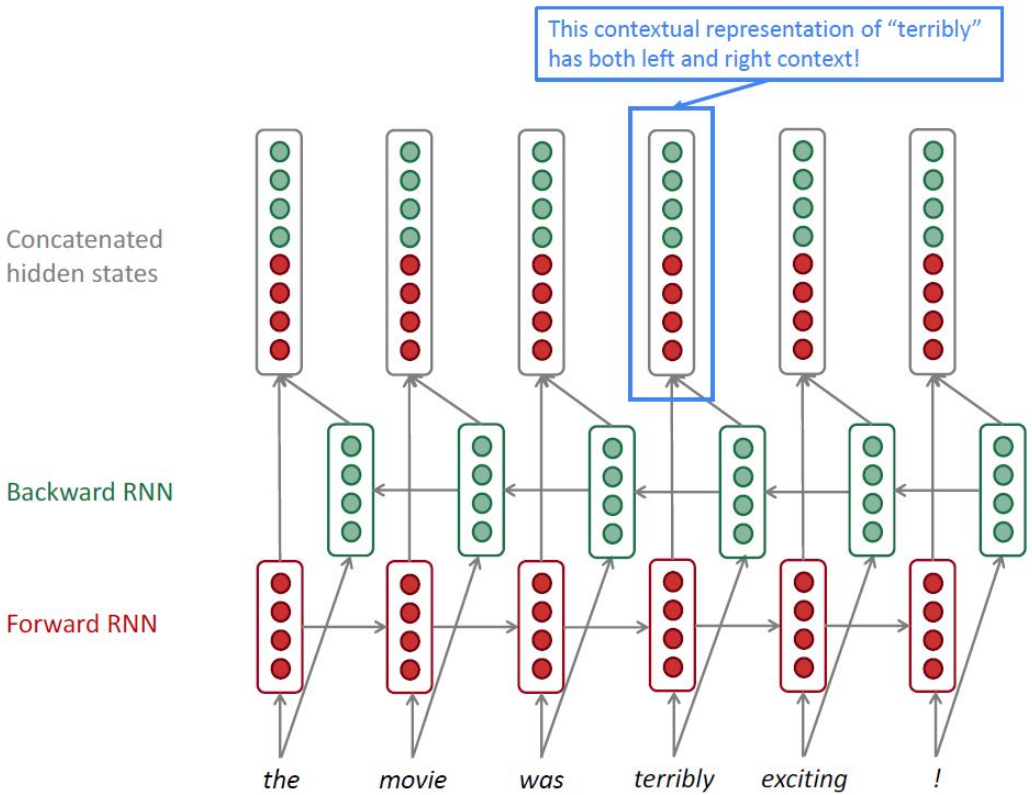
These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)

Bidirectional RNNs

Source: stanford 224n



$$\overleftarrow{h}^{(t)} = \sigma(W_{bx}x^{(t)} + W_{bh}h^{(t+1)} + b_b)$$

$$\overrightarrow{h}^{(t)} = \sigma(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + b_f)$$

Bidirectional RNNs

Source: stanford 224n

This is a general notation to mean "compute one forward step of the RNN" – it could be a simple RNN or LSTM computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

Generally, these two RNNs have separate weights

Concatenated hidden states $\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

Bidirectional RNNs

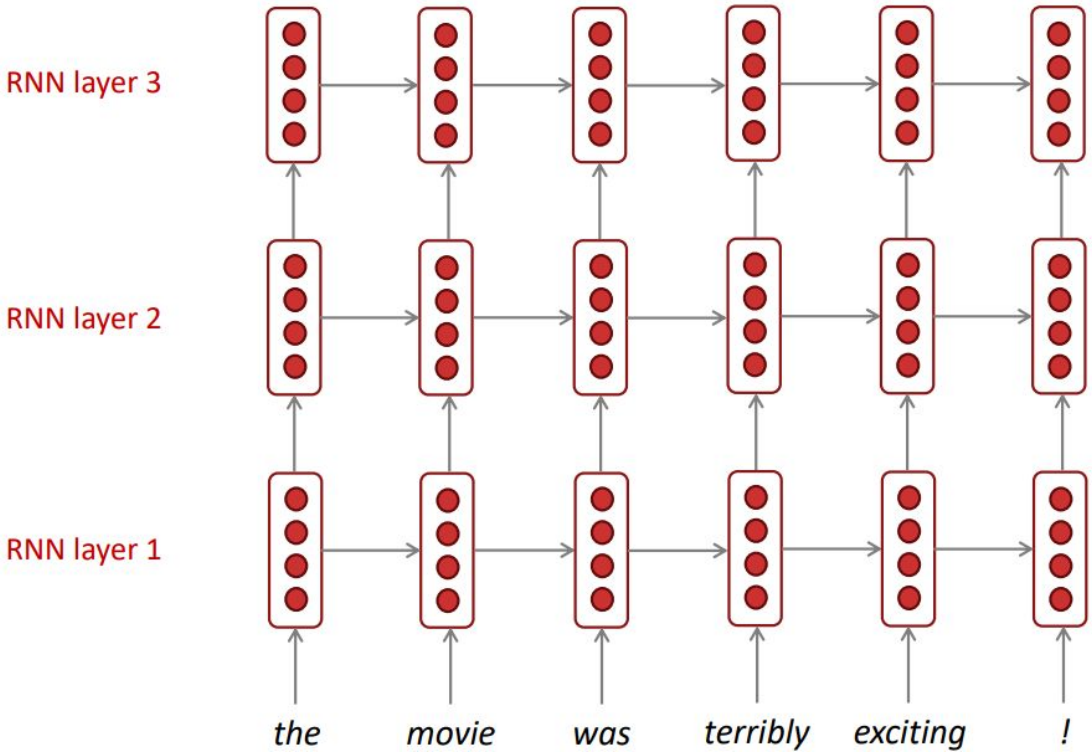
- Bidirectional RNNs are only applicable when you have access to the entire input sequence. They are **not** applicable to Language Modeling as the future tokens are not accessible.
- If you do have entire input sequence, bidirectionality is powerful for encoding (you should use it by default).
- For example, BERT (**Bidirectional** Encoder Representations from Transformers) is a powerful pretrained contextual representation system built on bidirectionality (we'll learn more about BERT later)

Stacked RNNs

- RNNs are already "deep" in one dimension (time)
- We can also make them "deep" in another dimension by putting an RNN on top of another – this is a multi-layer RNN.
- This allows the network to compute more complex representations
  - The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features.
- Multi-layer RNNs are also called stacked RNNs.

Stacked RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1



Stacked RNNs

- High-performing RNNs are often multi-layer (but aren't as deep as convolutional or feed-forward networks)
- For example: In a 2017 paper, Britz et al find that for Neural Machine Translation, 2 to 4 layers is best for the encoder RNN, and 4 layers is best for the decoder RNN
  - However, skip-connections/dense-connections are needed to train deeper RNNs (e.g. 8 layers)
  - Transformer-based networks (e.g. BERT) can be up to 24 layers

"Massive Exploration of Neural Machine Translation Architecutres", Britz et al, 2017

Outline for today

01 Recurrent neural networks

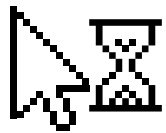
02 Long Short-Term Memory

03 Sequence modeling

04 Sequence-to-sequence learning

Seq2Seq Applications

- **Machine Translation**
- Summarization
- Dialogue Generation



Machine Translation (MT)

- Machine that can translate a text in one language to another.
- Classic test of language understanding
  - Language analysis and generation

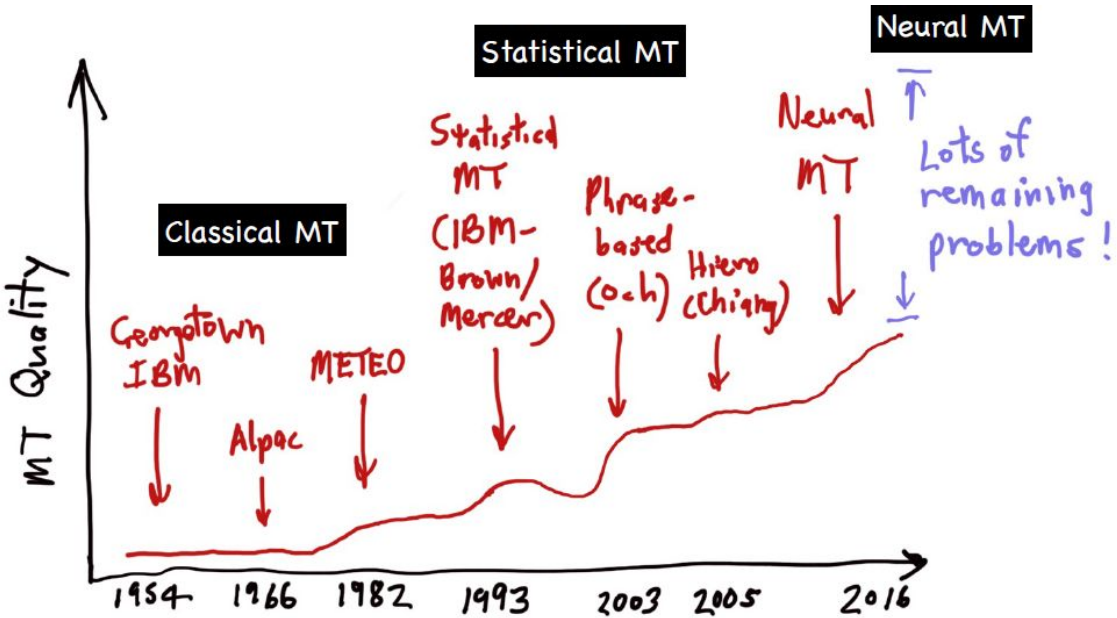
- German <=> English

Automatische Textverarbeitung gefällt mir.

↕

I like natural language processing.

Progress of MT



Source: Luong, Cho, Manning (ACL tutorial 2016)

- Microsoft, Google, Yandex claimed **human parity** in MT in 2018 with NMT.

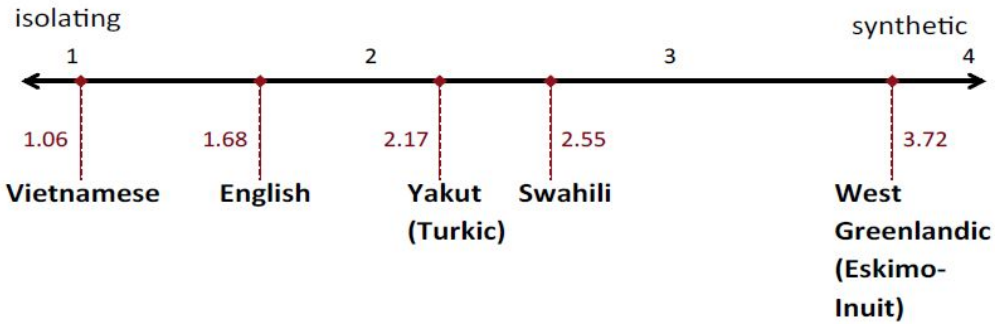
Why Is MT Difficult

- Think about the domains of language. How do they differ from one language to another?
  - Morphology
  - Syntax
  - Semantics
  - Pragmatics and discourse

Morphological Differences

Source: Dan Jurafsky (MT lecture)

- Morpheme: "Minimal meaningful unit of a language"
  Word = Morpheme + Morpheme + Morpheme +...
  reading = read + ing;
- Number of morphemes per word



Morphological Differences

Source: Dan Jurafsky (MT lecture)

- Cantonese:

"He said this was the biggest building in the whole country"

"Hkeui wa chyuhn gwok jeui daaih gaan nguk haih li gaan!"
he say entire country most big bldg house is this bldg

  - Each word in this sentence has one morpheme (and one syllable)

- Turkish:

"uygarlaştıramadıklarımızdanmışsınızcasına"

"uygar+laş+tır+ama+dık+lar+ımız+dan+mış+sınız+casına"
Behaving as if you are among those whom we could not cause to become civilized

Lexical Differences

Source: Dan Jurafsky (MT lecture)

- Word boundaries:

  Boundaries between words not marked in some languages

  - Chinese, Japanese, Thai

- Lexical Gaps:

  One-to-one mapping between lexical items is rare

  -Chinese have no term for "brother", "sister", "aunt", "uncle"...

- Mapping of words to phrases:

  Computer science => informatique (French)

Syntactic Differences

Source: Dan Jurafsky (MT lecture)

- Word order differences

English/German/French/Mandarin: **subject-verb-object (SVO)**

Hindi/Japanese: subject-object-verb (SOV)

Irish, Classical Arabic: **verb-subject-object (VSO)**

- English (SVO): He adores listening to music
- Japanese (SOV): He music to listening adores

Semantic Differences

Source: Dan Jurafsky (MT lecture)

- How languages distinguish spatial relations

Satellite-framed languages:

direction of motion is marked on the satellite (particle, prepositional/adverbial phrases)

E.g., English, Chinese

English: The bottle **floated** out

Spanish: The bottle exited **floating**

Verb-framed languages:

direction of motion is marked on the verb

E.g., Spanish, Japanese, French

Discourse and Pragmatic Differences

Use of pronouns, discourse markers, grammatical politeness

- **Cold** language: References to actors are not explicit, e.g., Chinese, Japanese
- **Hot** language: References to actors are explicit, e.g., English

飓风丽塔已经减弱为第三级飓风，
Rita weakened and was downgraded to a Category 3 storm;

ø 迫近美国德课萨斯州和路易斯安那州，
[Rita/it/the storm] is moving close to Texas and Louisiana;

1950s Early Machine Translation

Source: stanford 224n

- MT is how NLP got started!
- Russian → English
(motivated by the Cold War!)
- Systems were mostly rule-based, using a bilingual dictionary to map Russian words to their English counterparts

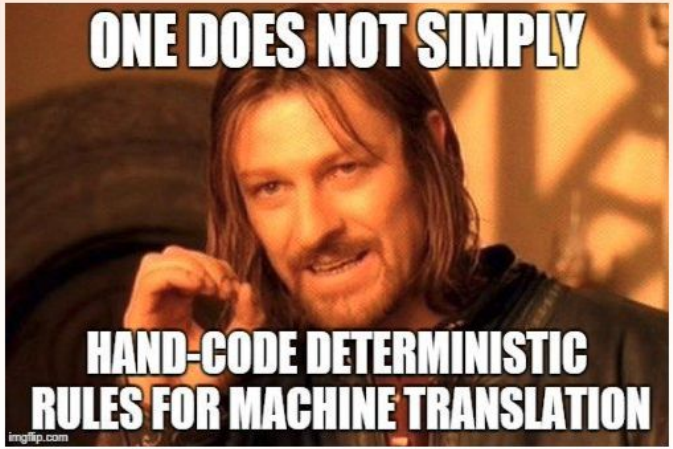
https://youtu.be/K-HfpsHPmvw

Problems with Rule-based Systems

- Writing rules requires a lot of human effort and knowledge

"Every time I fire a linguist, the performance of the speech recognizer goes up"

- Frederick Jelinek



- We should not undermine the importance of rules.
- Statistical/Neural MT relies on huge bilingual corpora, which may not be available for restricted domains (e.g., dialects, instructions).

Statistical Machine Translation (1985-2014)

- Parallel corpora (aka bilingual, bitext) are available in several language pairs.
  - Contains the "same" text in two or more languages
- Basic idea: use a parallel corpus as a training set of translation examples and learn a translation model.

- **E.g., Canadian Hansard** – parliamentary debates in English and French

E: *Canada should therefore drop any reference to any system other than the metric system in ads, on signs, and on packaging. The petitioners are also calling for containers to be standardized to the metric system in units of 100 grams or 100 millilitres.*

F: *Le Canada devrait donc abandonner toute référence à un système autre que le système métrique dans la publicité, l'affichage et sur les contenants. Les pétitionnaires demandent aussi l'uniformisation des contenants au système métrique par tranche de 100 grammes ou de 100 millilitres.*

Ms. Ève Péclet, 41st Parliament, #232

Statistical Machine Translation (1985-2014)

- Given a foreign (French) sentence **F**, find an English sentence **E**

$$\hat{E} = \underset{E \in English}{\operatorname{argmax}} P(E \mid F)$$

$$= \underset{E \in English}{\operatorname{argmax}} \frac{P(F \mid E)P(E)}{P(F)}$$

$$= \underset{E \in English}{\operatorname{argmax}} P(F \mid E)P(E)$$

Decoder (Beam search)

Translation model (faithful)

Language model (fluency)

Models how words and phrases should be translated; learn from parallel data

Models how to write good English; learn from monolingual data

Fluency Modeling P(E)

- We need a metric that ranks this sentence

  "That car almost crash to me!"

  as less fluent than this one:

  "That car almost hit me.!"

- Answer: language models

  P(me|almost, hit) > P(to|almost, crash)

- Advantage: this is monolingual knowledge.

Faithfulness Modeling P(F|E)

Spanish:

Maria no dió una bofetada a la bruja verde

English candidate translations:

Mary didn't slap the green witch
Mary not give a slap to the witch green
The green witch didn't slap Mary
Mary slapped the green witch

- All are fluent

- More faithful translations will be composed of phrases that are high probability translations
  - How often was "slapped" translated as "dió una bofetada"

in a large **bitext** (parallel English-Spanish corpus)

Faithfulness Modeling P(F|E)

- Goal is to compute P(F|E) from a bitext (E,F) corpus

Three options:

– Consider **sentence-pairs** (E,F) to compute P(F|E)?

=> Sparse

– Consider **words-pairs** (e_i, f_j) of the sentences and then take conditional independence assumption to compute P(F|E)?

=> Word alignment; Word-based MT

– Consider **phrase-pairs** ($\mathbf{e}_i$, $\mathbf{f}_j$) to compute P(F|E)?

=> Phrasal alignment; Phrase-based MT

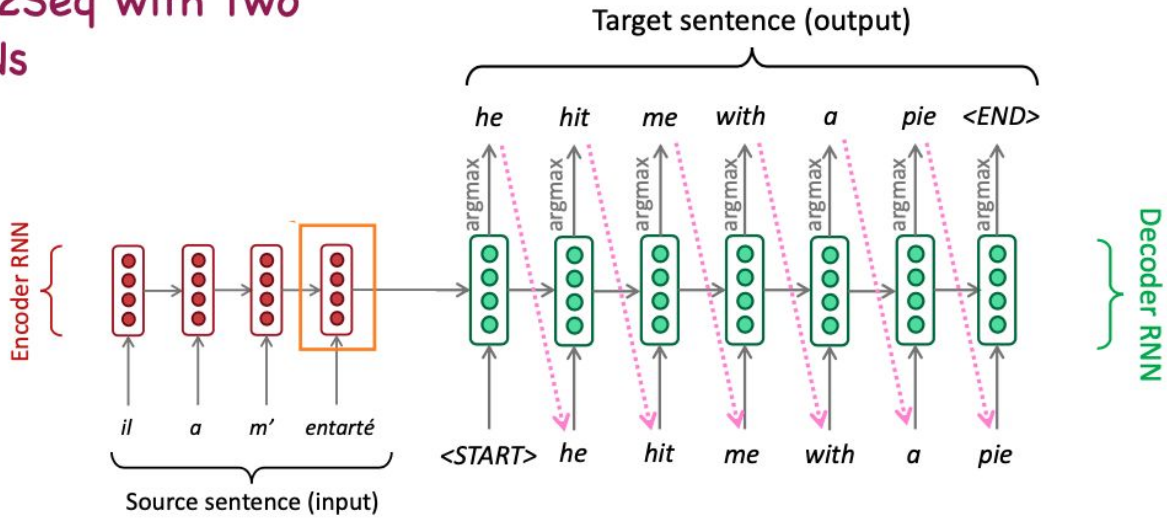
Neural Machine Translation (2014-)

- Machine Translation with a single neural network
- Two models (e.g., RNNs) are put together
  - One acts as an Encoder
  - The other acts as a Decoder
- The neural architecture is called sequence-to-sequence (seq2seq)
- **Distributed representation** of words and phrases
- **End-to-end** training
- Allows **larger** context (even extra sentential)

Sequence-to-Sequence Model

Source: stanford 224n

- Seq2Seq with Two RNNs

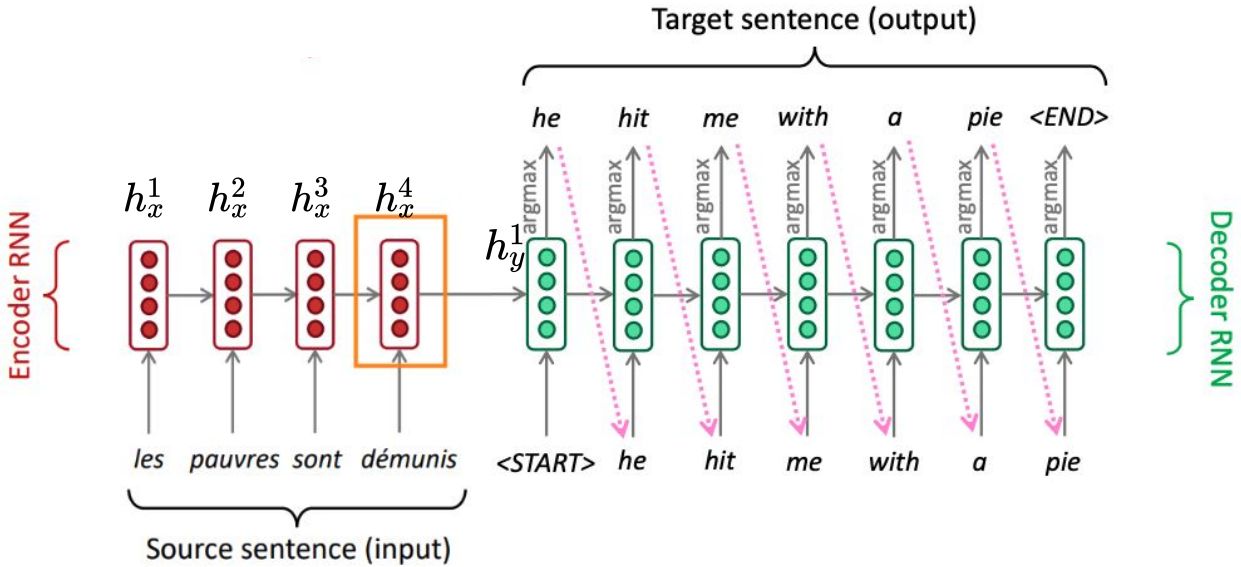


Encoder produces an encoding of the source sentence.

Provides Initial hidden state for Decoder

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

Sequence-to-Sequence Model



Encoding of the source sentence. This needs to capture all information about the source sentence.

$$h_y^1 = f(W_{ye} \cdot e_{<START>} + W_{yh} \cdot h_x^4 + b_y)$$

$$h_x^1 = f(W_{xe} \cdot e_{les} + W_{xh} \cdot h_x^0 + b_x)$$

Sequence-to-Sequence Model

- Directly models the conditional probability p(y|x) of translating a source sequence $x_1, \ldots, x_n$ to a target sequence $y_1, \ldots, y_m$.

$$x = les\ pauvres\ sont\ de'munis$$
$$y = the\ poor\ dont\ have\ any\ money$$

$$p(y|x) = \sum_{j=1}^{m} \log p(y_j|y_{<j}, \mathbf{s})$$

s is the encoding of the source sentence.

$$p(y_j|y_{<j}, s) = softmax(Uh_j + b)$$

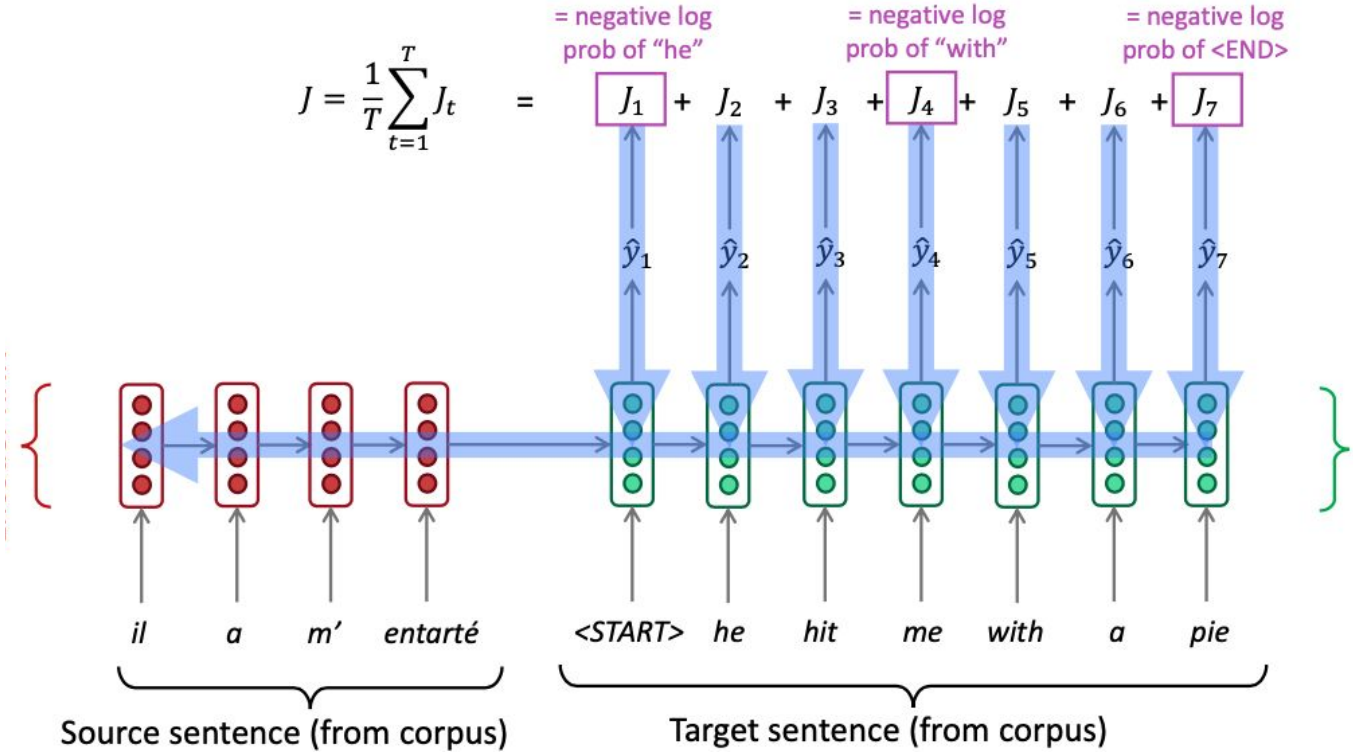
$$h_j = RNN(h_{j-1}, y_{j-1})$$

- Training objective

$$J_t = \sum_{(x,y)\in\mathcal{D}} -\log p(y|x)$$

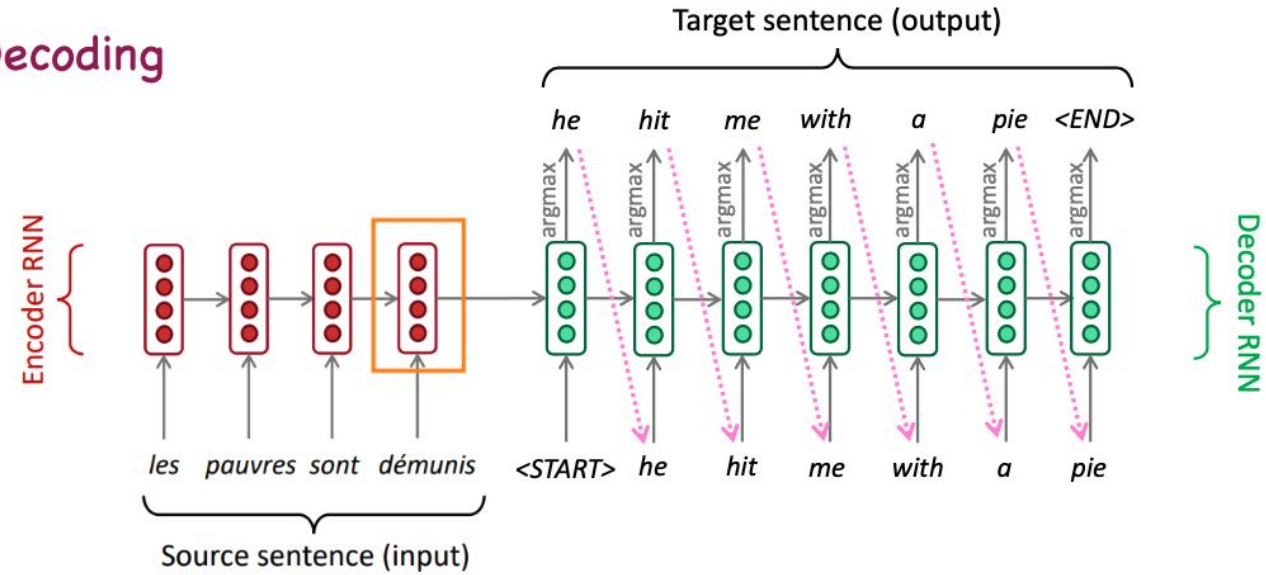
Training a Seq2Seq Model

Source: stanford 224n



Generation

- Decoding



- Generate (or "decode") the target sentence by taking argmax on each step of the decoder
- This is greedy decoding (take most probable word on each step)

Evaluate NMT

Source: stanford 224n

BLEU (**B**ilingual **E**valuation **U**nderstudy)

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - *n*-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations
- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation ☹

BLEU (Optional)

- N-gram precision

Precision 1-gram = Number of correct predicted 1-grams / Number of total predicted 1-grams

$$GAP = \prod_n p_n^{w_n} = p_1^{1/2} p_2^{1/2}$$

p_1

Target Sentence: The guard arrived late because ~~it was raining~~

Predicted Sentence: The guard arrived late because of the rain

p_2

Target Sentence: The guard arrived late because it was raining

Predicted Sentence: The guard arrived late because of the rain

BLEU (Optional)

- Brevity penalty: penalizes sentences that are too short.

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

- c is *predicted length* = *number of words in the predicted sentence* and
- r is *target length* = *number of words in the target sentence*

$$bleu = \text{Brevity Penalty} \times GAP$$

Advantages of NMT

Source: stanford 224n

- 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

Disadvantages of NMT

Source: stanford 224n

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!

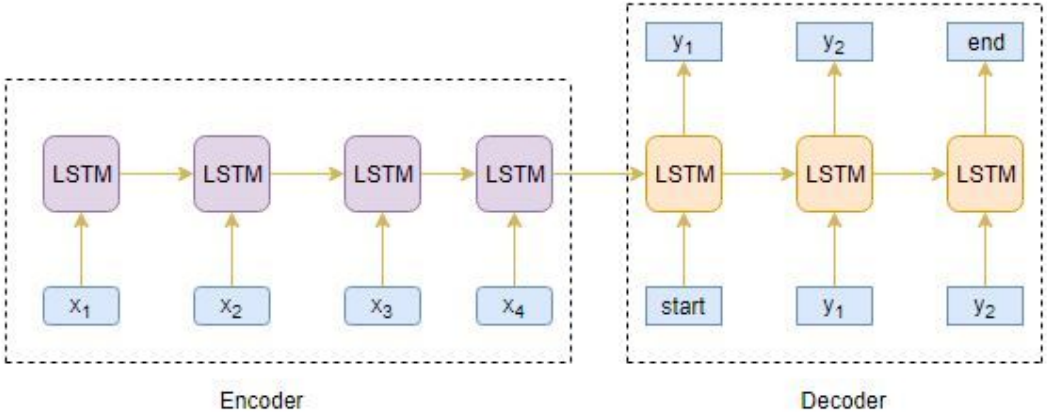
Seq2Seq Usage

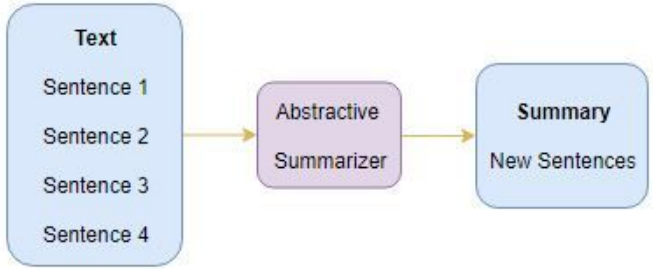
Source: stanford 224n

- One model for almost all NLP tasks

  - Not just MT!
  - Many NLP tasks can be phrased as sequence-to-sequence:
    - Summarization (long text → short text)
    - Dialogue (previous utterances → next utterance)
    - Parsing (input text → output parse tree)
    - Code generation (natural language → Python code)
    - Segmentation/tagging (input text → output tag sequence)

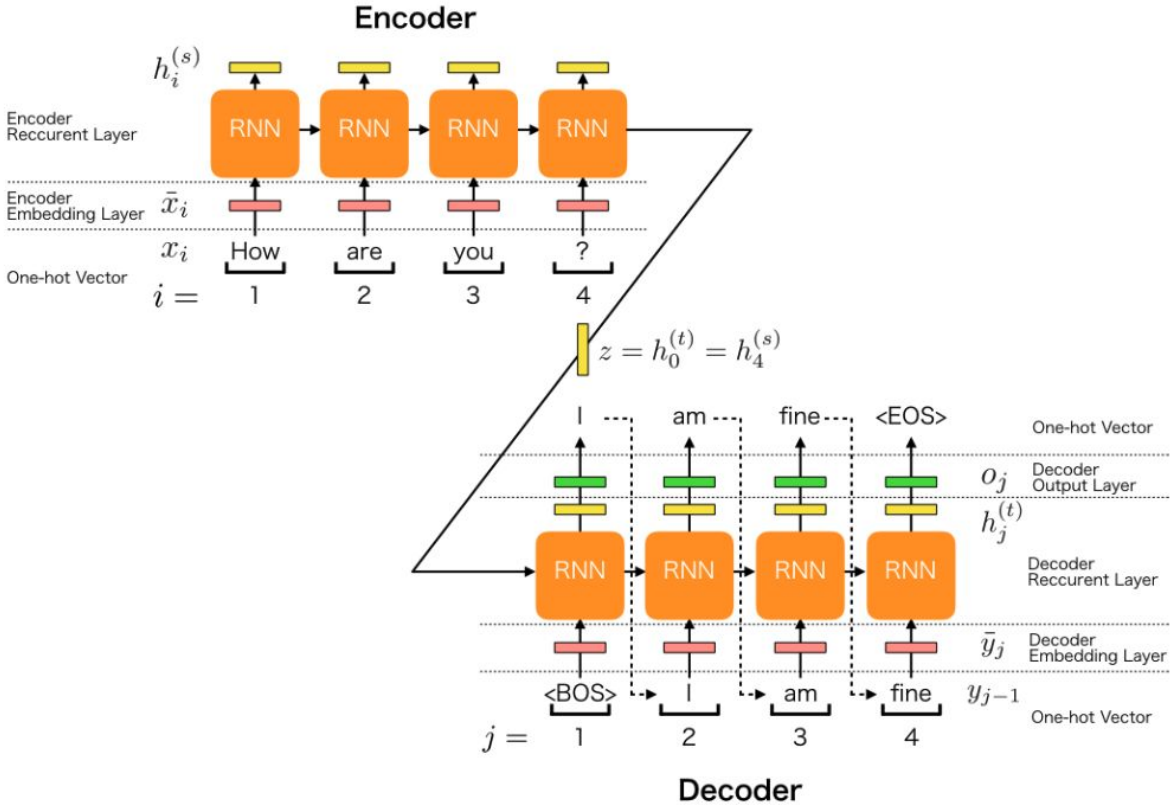
Seq2Seq for Summarization



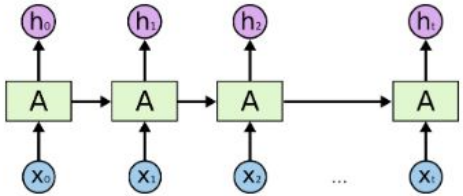


> "Automatic text summarization is the task of producing a concise and fluent summary while preserving key information content and overall meaning"
>
> -Text Summarization Techniques: A Brief Survey, 2017

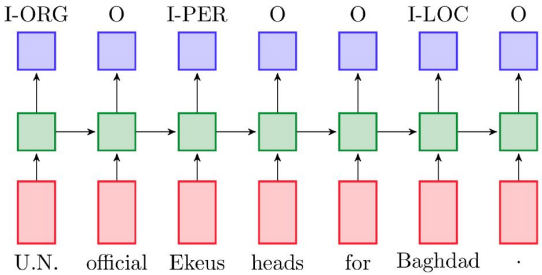
Seq2Seq for Dialogue



Summary

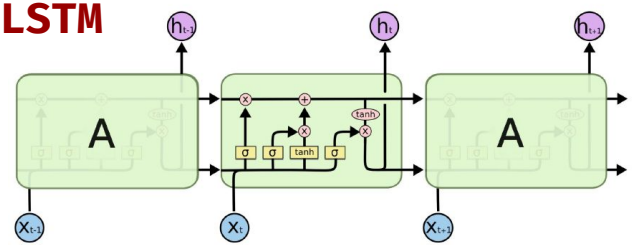


RNN



Sequence Tagging

LSTM



Seq2Seq