Language Models and Recurrent Neural Networks

Natural Language Processing

(based on revision of Chris Manning Lectures)

Announcement

- TA announcements (if any)...

Suggested Readings

1. N-gram Language Models (textbook chapter)
2. The Unreasonable Effectiveness of Recurrent Neural Networks (blog post overview about RNN)
3. Sequence Modeling: Recurrent and Recursive Neural Nets (Sections 10.1 and 10.2)
4. On Chomsky and the Two Cultures of Statistical Learning (some cool stuffs about LM)

Language Modeling

Language Modeling

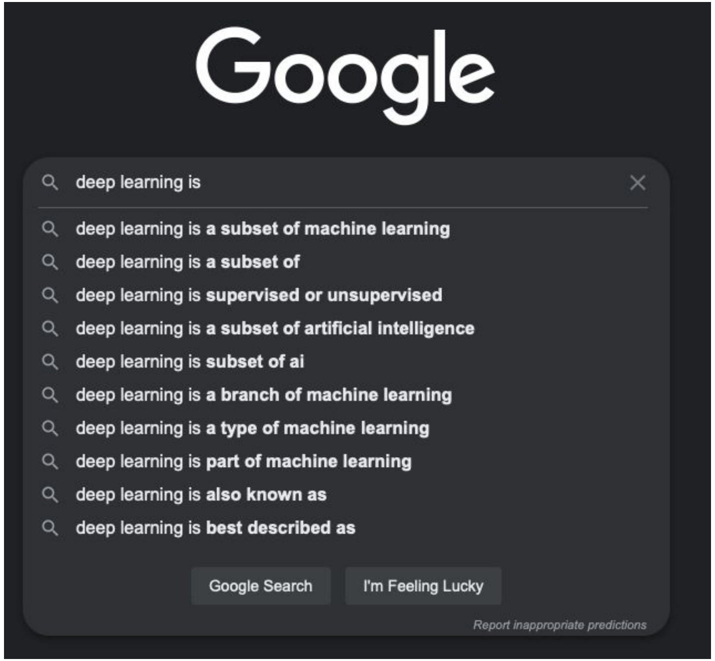
- Language Modeling is the task of predicting what word comes next
  - *The student open their* _______
- More formally: given a sequence of words $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(t)}$, compute the probability distribution of the next word $\mathbf{x}^{(t+1)}$

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \cdots, \mathbf{x}^{(1)})$$

- Such system is called a Language Model

Language Modeling

We use LM everyday!



n-gram Language Models

The student open their _______

- **Question**: How to learn a LM?
- **Answer** (pre-Deep Learning era): learn an n-gram LM!

- **Definition**: A **n-gram** is a chunk of *n* consecutive words
  - unigrams: “the”, “students”, “opened”, “their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - 4-grams: “the students opened their”

- **Idea**: Collect **statistics** about how frequent different n-grams are

n-gram Language Models

- Suppose we are learning a 4-grams LM.

~~as the proctor started the clock, the~~ $\underbrace{\textit{students opened their}}_{\text{condition on this}}$ ______

discard

$$P(\mathbf{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \mathbf{w})}{\text{count}(\text{students opened their })}$$

For example, suppose that in the corpus:

- “Students opened their” (i.e., n-1 grams) occurred **1000** times
- “Students opened their **books**” (i.e., n grams) occurred **400** times
    - P(**books** | students opened their) = 0.5
- “Students opened their **exams**” (i.e., n grams) occurred **100** times
    - P(**exams** | students opened their) = 0.1

n-gram Language Models: Formally

- First, we make a **Markov assumption**: $\mathbf{x}^{(t+1)}$ depends only on the preceding n-1 words

$$P(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \cdots, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \cdots, \mathbf{x}^{(t-n+2)})$$

- **Question**: So what's the probabilities?
- **Answer**: Simply divide prob of n-gram by (n-1) gram

$$= \frac{P(\mathbf{x}^{(t+1)}, \cdots, \mathbf{x}^{(t-n+2)})}{P(\mathbf{x}^{(t)}, \cdots, \mathbf{x}^{(t-n+2)})}$$

- **Question:** But where to get these probabilities in the first place?
- **Answer**: By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\mathbf{x}^{(t+1)}, \mathbf{x}^{t}, \cdots, \mathbf{x}^{(t-n+2)})}{\text{count}(\mathbf{x}^{(t)}, \mathbf{x}^{t}, \cdots, \mathbf{x}^{(t-n+2)})}$$

n-gram Language Models: Problem

- **Problem 1**: What if "*students opened their w*" never occurred in data? Then *w* has a probability of 0
  - Use Laplace smoothing, i.e., add small probability to the count of every *w* in V
- **Problem 2**: What if "*students opened their*" never occurred?
  - Use "opened their" instead. This is called *backoff*.
- **Problem 3:** Need to store count of all n-grams
  - Expensive!

Main caveat: Increase *n* in the n-grams makes these problem worse. Typically we can't have *n* bigger than 5

Generative text with an n-gram LM

You can use a LM to generate text

today the ________

condition on this

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

sample

today the price ________

condition on this

get probability distribution

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
…	

sample

"today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share."

Surprisingly grammatical, but incoherent.

Increasing *n* would works better but worsens sparsity problem and increases model size

Fixed Window Neural LM

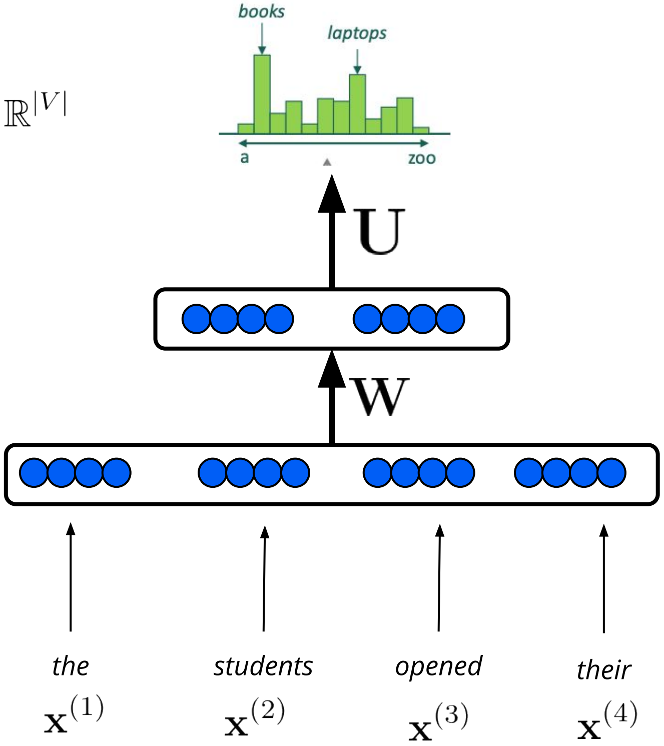
Y. Bengio, et al. (2000/2003): A Neural Probabilistic Language Model

$$\hat{\mathbf{y}} = \mathrm{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$



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

Remaining problems:

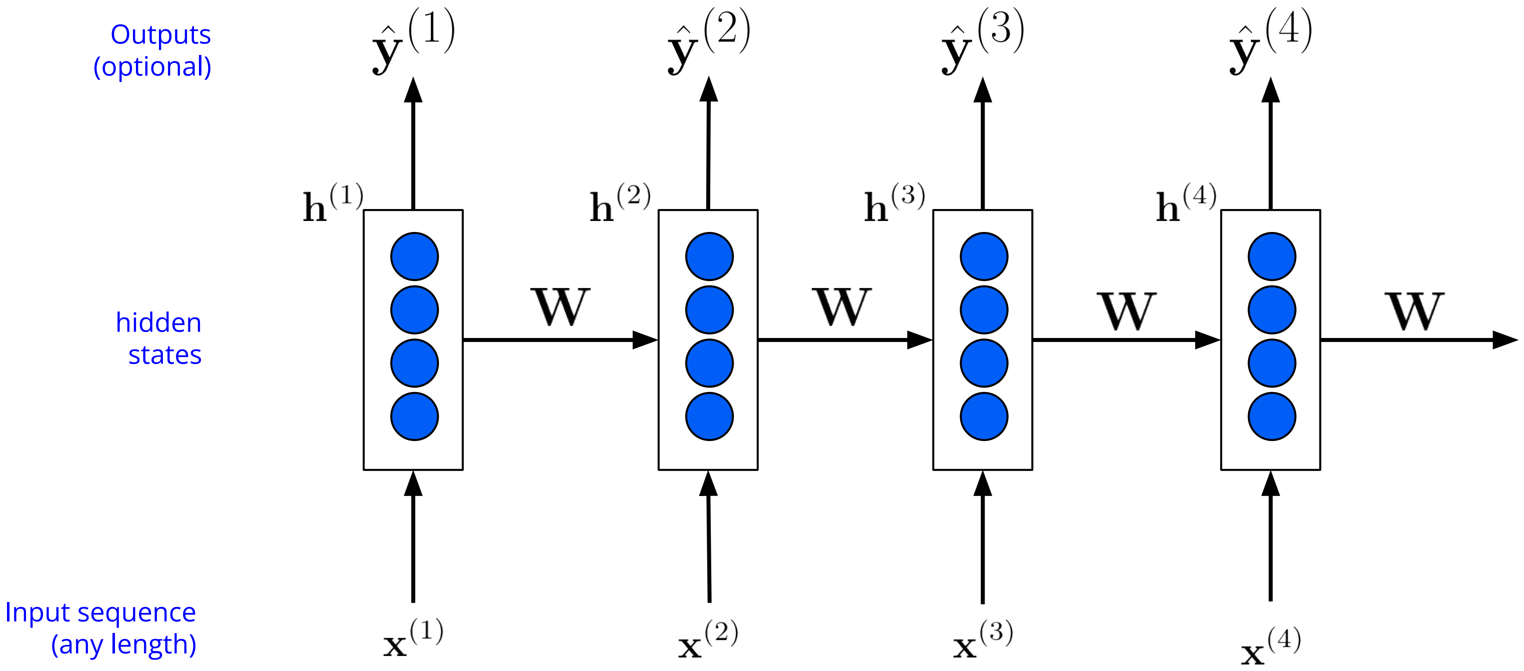
- Fixed window is **too small**
- Enlarging window enlarges W
- In fact, window can never be large enough!
- $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are multiplied by completely different weights in W, allowing no interaction between words.

We need a neural architecture that can process any length input

Recurrent Neural Networks

Recurrent Neural Networks (RNN)

Core idea: Apply the same weights W repeatedly



A simple RNN LM

output distribution

$\hat{\mathbf{y}}^{(t)} = \mathrm{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b_2}) \in \mathbb{R}^{|V|}$

hidden states

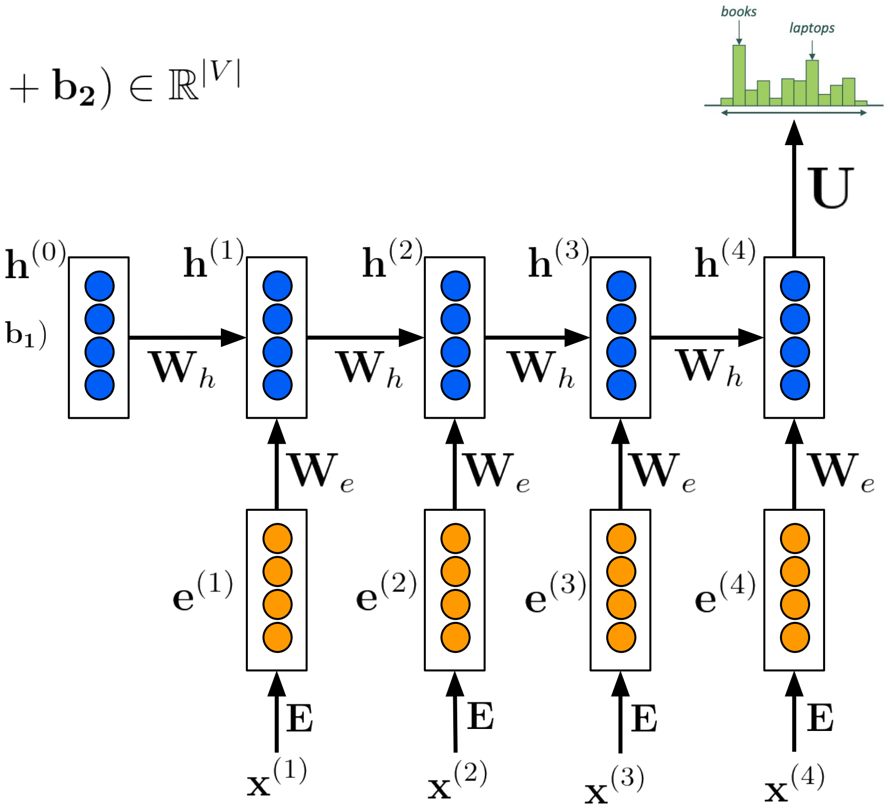
$\mathbf{h}^{(t)} = f(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$

words embeddings

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

words/one-hot vectors

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



- Can process any length input
- Can use information from many steps back
- Model size does not increase because W is shared

Remaining problems:

- Recurrent computation is **slow** because it's sequential
- In reality, **difficult to access information from many steps back** (more on this later in the course)

Training an RNN LM

- Get a **big corpus of text** which is a sequence of words $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\mathbf{y}}^{(t)}$ 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 $\mathbf{y}^{(t)}$, and the true next word $\mathbf{x}^{(t+1)}$ (one hot for $\hat{\mathbf{y}}^{(t)}$):

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

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

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

Training an RNN LM

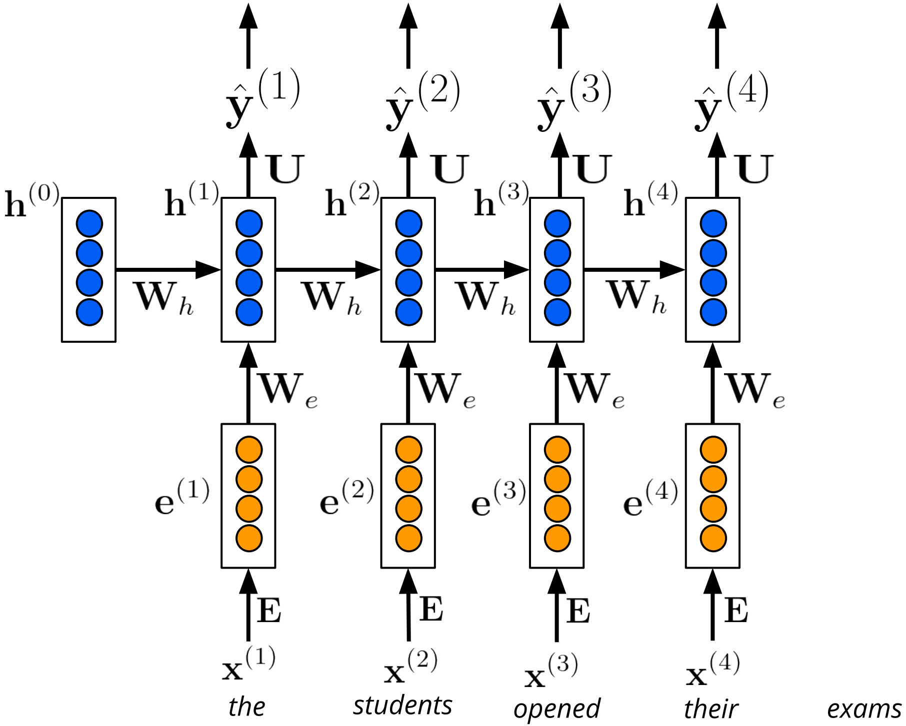
Cross entropy between predicted word $\hat{\mathbf{y}}^{(3)}$ and "*their*"

Loss

$$J^{(1)}(\theta) + J^{(2)}(\theta) + \boxed{J^{(3)}(\theta)} + J^{(4)}(\theta) = J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\mathbf{y}}^{(t)}_{\mathbf{x}_{t+1}}$$

Predicted prob dists.

Corpus



This way of training is called "**Teacher forcing**"

Training an RNN LM - more details

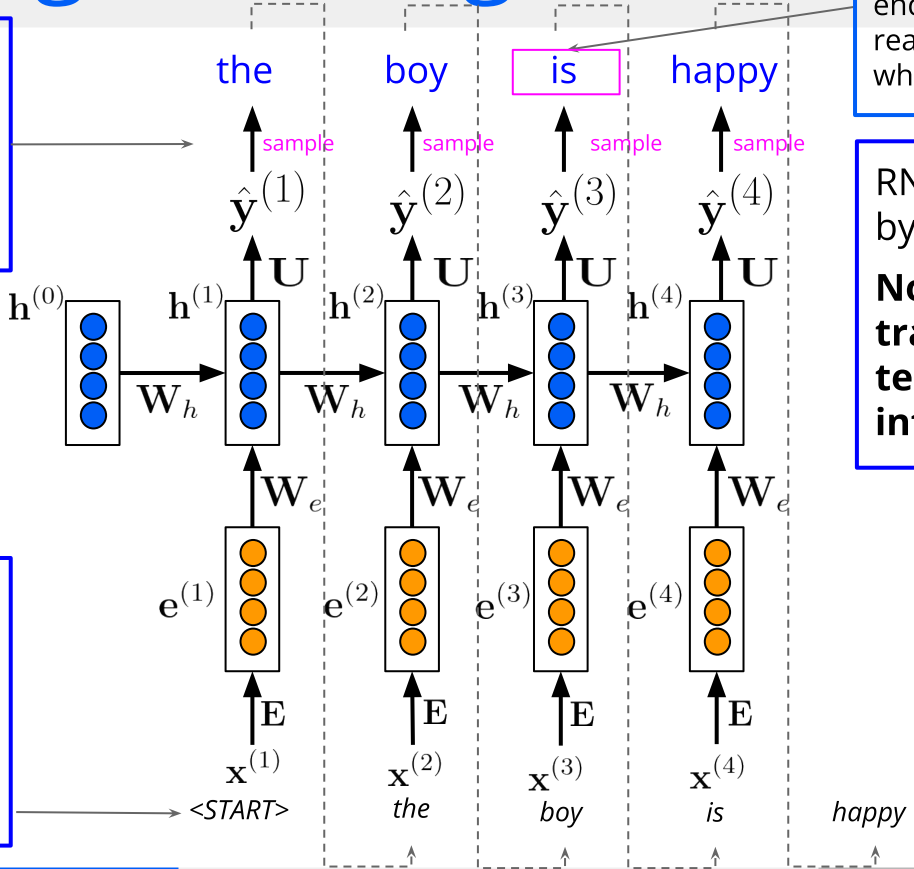
- Better to perform **stochastic gradient descent** instead to save computational time
  - Use batch of sentences, instead of the whole corpus
- The derivative w.r.t the repeated weight matrix $\mathbf{W}$ is simply the sum of all gradients of each time step

$$\frac{\partial J^{(t)}}{\partial \mathbf{W}_h} = \sum_{i=1}^{t} \left. \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \right|_{(i)}$$

This is also known as "**backpropagation through time**"

RNN LM - generating text

This sampling process is called "**decoding**", which has many approaches, e.g., random, top-k, top-1, beam-search, etc. We shall cover this later in our course.



The word "is" will be fed as $\mathbf{x}^{(4)}$. The whole generation process can end when the desired length is reached, or we can also set to end when reaching <END> token.

RNN-LM can generate text by **repeated samplings.**

Note that this is NOT in training mode (i.e., teacher forcing) but inference/testing mode.

If we want to generate purely from scratch, during training, we need to append <START> and <END> token so that the model know what words are usually in the beginning or ending of a sentence

RNN LM - generating text

Obama speeches

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

The speeches

Here is a selection of some of my favorite speeches the Obama-RNN generated so far. Keep in mind this is a just a quick hack project. With more time & effort the results can be improved.

SEED: Jobs

Good afternoon. God bless you.

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done. The promise of the men and women who were still going to take out the fact that the American people have fought to make sure that they have to be able to protect our part. It was a chance to stand together to completely look for the commitment to borrow from the American people. And the fact is the men and women in uniform and the millions of our country with the law system that we should be a strong stretcks of the forces that we can afford to increase our spirit of the American people and the leadership of our country who are on the Internet of American lives.

Thank you very much. God bless you, and God bless the United States of America.

RNN LM - generating text

Let’s try some harry potter theme

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

“The Malfoys!” said Hermione.
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.

RNN LM - generating text

Let’s try some recipes

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

```
    Title: BARS REST LAYER SAUCE
Categories: Poultry, Beef
    Yield: 4 Servings

    3 tb Baking powder
  1/3 c  Brown sugar
    2 c  Flour; sifted
    2 tb Butter or margarine
    1 ts Sugar
    3 tb Lemon juice
    1    Garlic cloves; etchick

 1.  Grill and divide into serving platter. Each balls (airlecking)
  overning ripe beef with a spoonfuls on platter. Sprinkle each rounds
 over top. Slice the muffin cups and pourient of a 2-quart pan. Place the
 oil in a large skillet over medium heat until chicken is
 boiled, to the center of the egg mixture. Pour over and roll it for an
 about 3-4 pounds and can be stored is changes have for sized onion from
 page in with pan with canned extratty fish on a glass or rounding
 pan. Source: Nutryand Cooking prokess; 1971.
```

Evaluation of LM

- The standard evaluation metric for LM is perplexity (lower better!)

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \cdots, \mathbf{x}^{(1)})} \right)^{\boxed{1/T}}$$

normalization factor

- Equal to the exponential of the cross-entropy loss

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)} \right) = \exp(J(\theta))$$

RNN has improved perplexity

n-gram model →

Increasingly complex RNNs

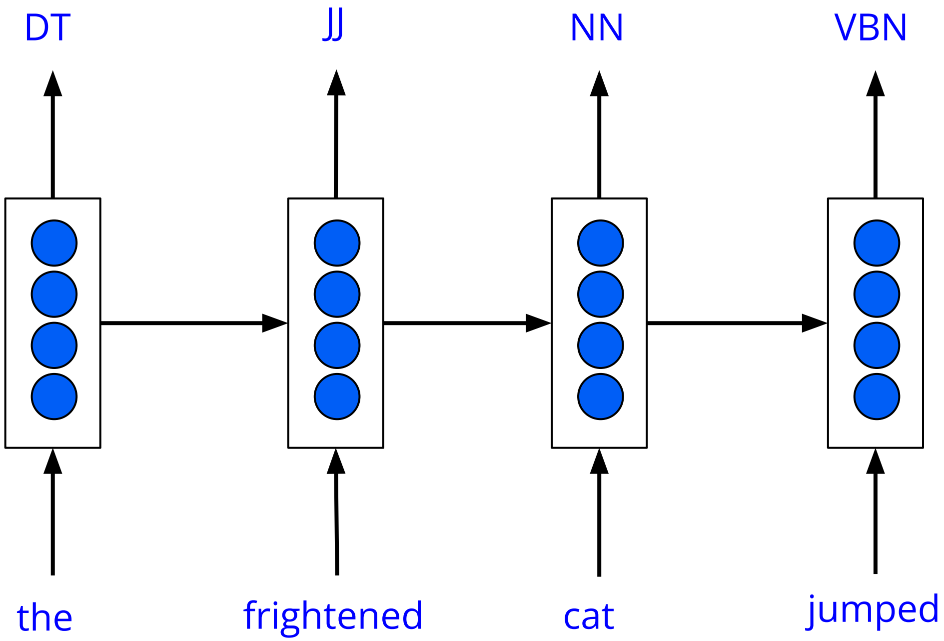
Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192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

Perplexity improves (lower is better)

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

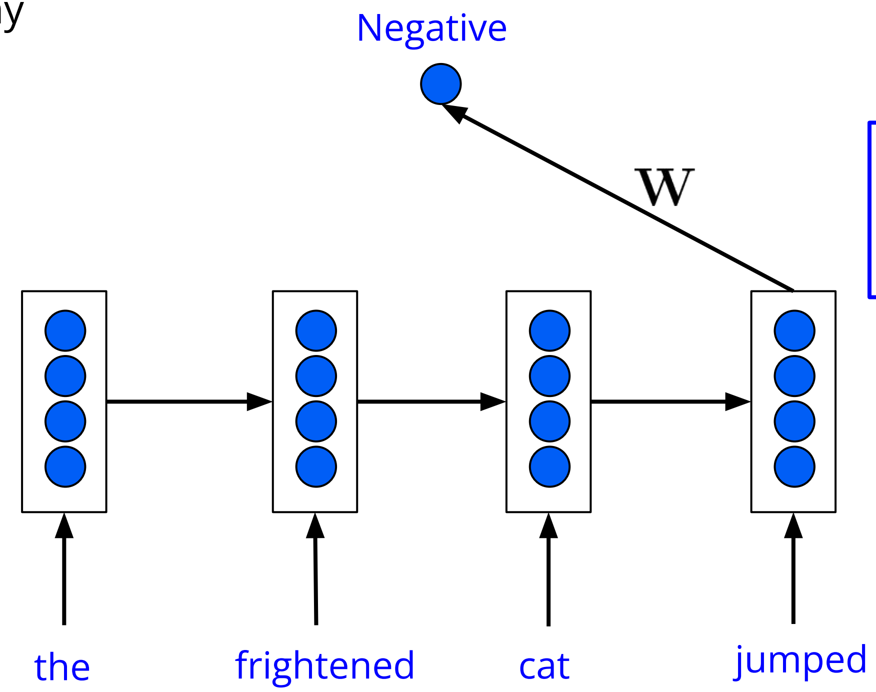
RNN is NOT LM - POS Tagging, NER

RNN can be useful for many purposes, e.g., **POS tagging, NER**



RNN is NOT LM - Sentiment Analysis

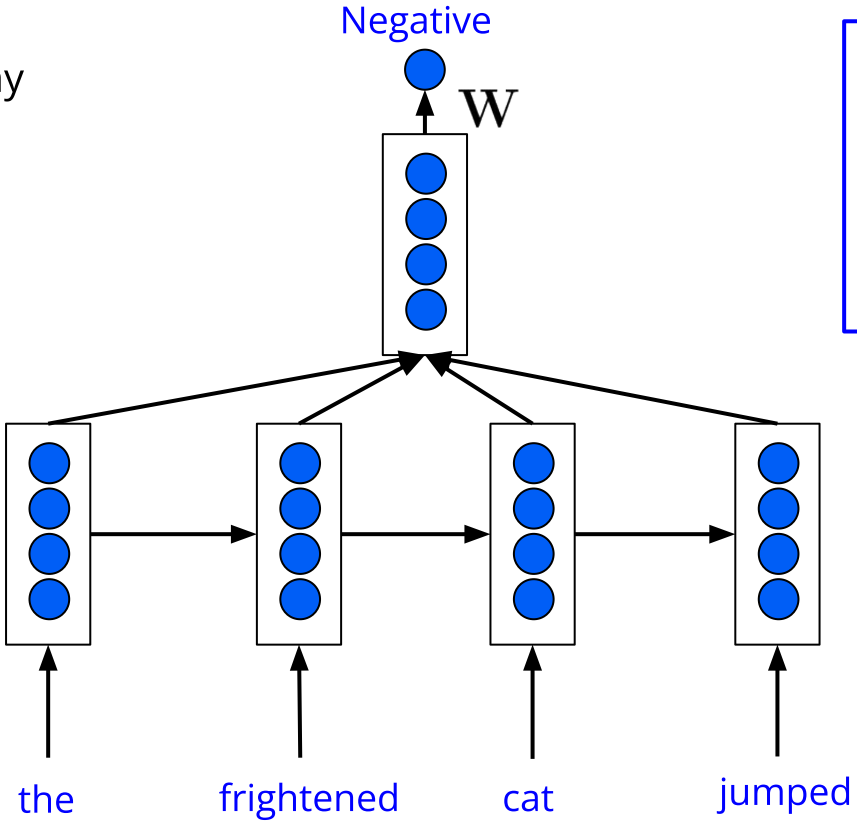
RNN can be useful for many purposes, e.g., **sentiment analysis**



Basic way is to use the **final hidden state**, and then simply project to 1

RNN is NOT LM - Sentiment Analysis

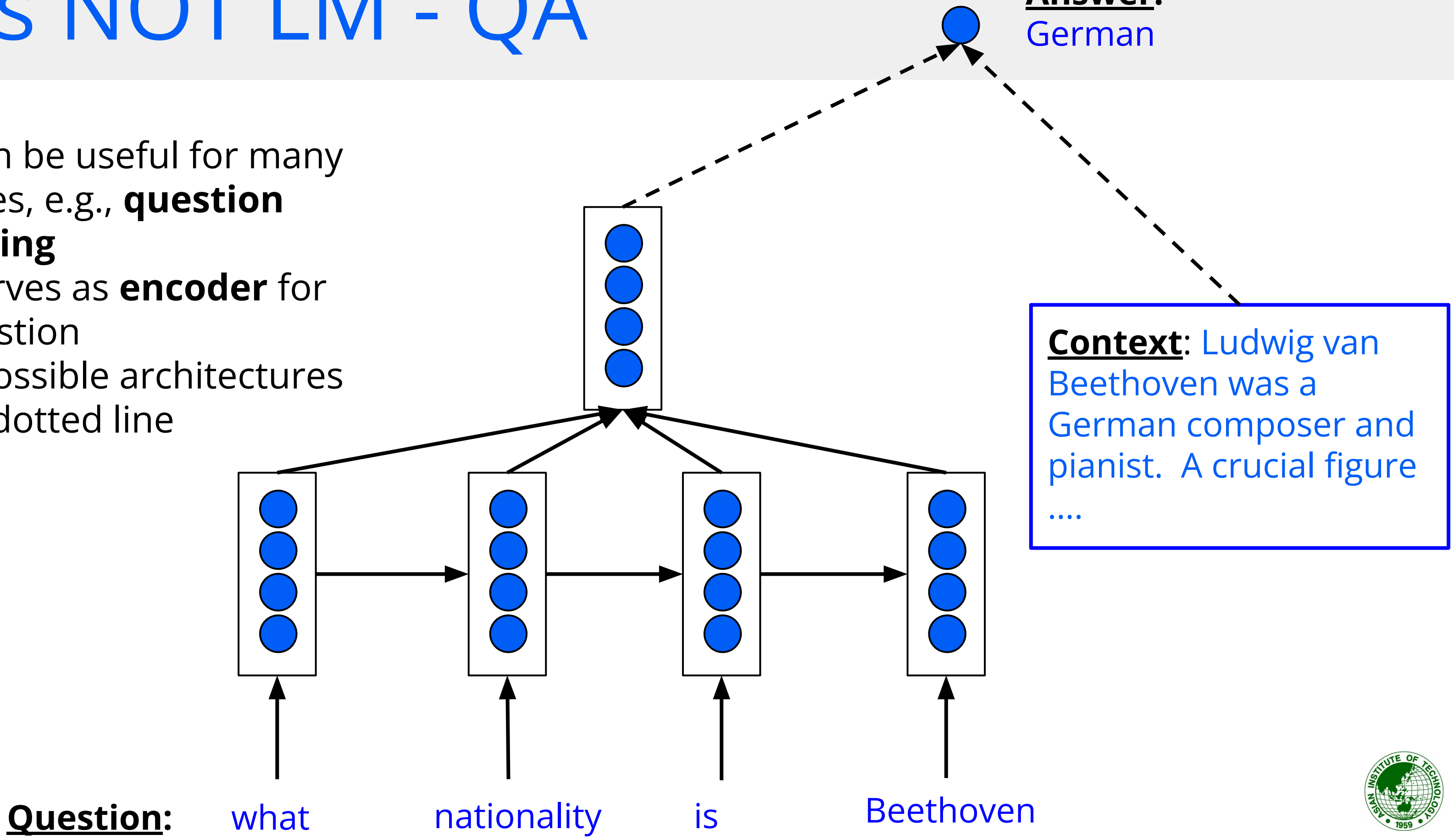
RNN can be useful for many purposes, e.g., **sentiment analysis**



A better way for NLP is to usually use the **element-wise mean or max of all hidden states**, and then simply project to 1

RNN is NOT LM - QA

- RNN can be useful for many purposes, e.g., **question answering**
- RNN serves as **encoder** for the question
- Many possible architectures for the dotted line



Terminology

- RNN in this lecture = simple/**vanilla**/Elman RNN
- More variants of RNN (all interchangeable)
  - LSTM
  - GRU
  - Stacked LSTM
  - Stacked Bidirectional LSTM
  - Stacked Bidirectional LSTM with attention
  - Stacked Bidirectional LSTM with attention with residual connections
- Next lecture!

Summary

- LM: a system that predicts the next word
  - Can be used for **generating text**
  - Not easy to notice that it's a **benchmark task** because is a subcomponent of so so many NLP tasks
- A **n-gram** LM suffers from sparsity and size problems
- **RNN**
  - Takes any length input
  - Apply same weights
  - Can product output on each step
- RNN is NOT LM, RNN are actually even more useful!
  - POS tagging,
  - Sentence classification,
  - Question answering,
  - …