CSC-496/696: Natural Language Processing and Text as Data

Lecture 13: Recurrent Neural Networks

Patrick Wu

Friday, October 18, 2024

Lecture Contents

Announcements

Announcements

- Assignment 3 has been posted
- It is due on **Tuesday, October 29 at 11:59pm**
- Two problems — upcoming assignments seem shorter, but are more open ended

Final Project

I wanted to take a poll: how many people are planning to complete the research project vs. the task-driven project?

Final Project

- The format of the paper will follow the Association for Computational Linguistics (ACL) paper format
- This is a common format used to write NLP papers
- Example: https://aclanthology.org/2022.emnlp-main.696.pdf

Final Project

- The format of the paper will follow the Association for Computational Linguistics (ACL) paper format
- This is a common format used to write NLP papers
- Example: https://aclanthology.org/2022.emnlp-main.696.pdf
- You can work on the project either yourself or with a co-author
- Single-authored project — minimum 3 pages
- Co-authored project — minimum 5 pages
- You can go over these page limits if you want, but there is a maximum of 8 pages
- No extra credit for going over minimum page counts

Midterm

- Midterm grades were posted
- Mean Score: 59.6
- Standard Deviation: 21.3
- I will hand back midterms at the end of class
- I will post solutions by the end of the day
- The midterm itself will not be curved

Reviewing Usage of LLMs in This Course

- There are appropriate and inappropriate usages for using generative LLMs such as ChatGPT
- These are outlined in the class syllabus, but of course, I cannot enforce it

Reviewing Usage of LLMs in This Course

Some tips if you choose to use it

- Treat it as a lab partner rather than a solutions generator. You could use it to clarify concepts, help get you started in the right direction on the assignments, etc.
- Set a fixed amount of time to try a problem yourself without any LLM help. Use that time to carefully read the problem and outline a solution, review class notes and readings, etc.
- If you do generate code with an LLM, walk through the code. Does every line make sense? What does each line do?

Discussing Word Embedding Project Ideas From Last Class

We’ll now discuss the word embedding project ideas you came up with from the last class

Review Questions

What is a word embedding?

(A) A vector representation of a document
(B) A vector representation of a word
(C) A vector representation of an idea
(D) A vector representation of a concept

Review Questions

If I ask you to describe the word embedding of the word “lizard,” what would that be?

(A) A scalar (a number)
(B) A matrix with shape $n \times m$
(C) A vector of n-dimension
(D) Several different possible vectors of n-dimension

Review Questions

We get to control the dimensions of a word embedding.

(A) True

(B) False

Review Questions

What is one way to create a *document* embedding from word embeddings?

(A) Take the word embedding for each word and average all the numbers together to get a single scalar number
(B) Take the word embedding for each word and average all the embeddings by collapsing the rows, so we're left with a vector that has a dimension equal to the number of words
(C) Take the word embedding for each word and average all the embeddings by collapsing the columns, so we're left with a vector that has a dimension equal to the number of dimensions for each word embedding

Intuition Behind Recurrent Neural Networks

Today's Topic

- *Recurrent* neural networks
- What does the word "recurrent" mean?

Today's Topic

- *Recurrent* neural networks
- What does the word "recurrent" mean?
- We now move to a different type of neural network
- While fully connected neural networks are good at recognizing patterns (think of the MNIST digits example), they aren't great at understanding *sequences*

Why a *Recurrent* Neural Network?

Imagine you want to understand a story or conversation

Why a *Recurrent* Neural Network?

Imagine you want to understand a story or conversation

- You will read the story from left to right (if it's in English)

Why a *Recurrent* Neural Network?

Imagine you want to understand a story or conversation

- You will read the story from left to right (if it's in English)
- You need the previous context to understand the next words or ideas

Why a *Recurrent* Neural Network?

Imagine you want to understand a story or conversation

- You will read the story from left to right (if it's in English)
- You need the previous context to understand the next words or ideas
- A *recurrent* neural network (RNN) operationalizes this intuition

Intuition Behind RNNs

- You can think of RNNs as having a memory when reading a book
- Unlike fully connected neural networks, RNNs can "remember" past information
- They read one word (or character) at a time and have a mechanism to carry information from what they previously encountered
- This memory mechanism allows RNNs to make predictions or assessments over text more accurately

Intuition Behind RNNs

- Stands in contrast to a fully connected neural network
- FCNN: accepts a fixed-sized vector as input, produces a fixed-sized output (e.g., probabilities of different classes)
- Again, think back to the MNIST example: it accepts as input a 784-dimensional vector (the image flattened) and outputs 10 probabilities

Intuition Behind RNNs

- We could use an FCNN with images because we knew that every observation was 1 image that was 28×28

Intuition Behind RNNs

- We could use an FCNN with images because we knew that every observation was 1 image that was 28×28
- We could also use an FCNN with word embeddings because we can take every word's embedding in a document and average the embeddings to create a document embedding

Intuition Behind RNNs

- We could use an FCNN with images because we knew that every observation was 1 image that was 28×28
- We could also use an FCNN with word embeddings because we can take every word's embedding in a document and average the embeddings to create a document embedding
- In other words, it could only take in a fixed input

Intuition Behind RNNs

- We could use an FCNN with images because we knew that every observation was 1 image that was 28×28
- We could also use an FCNN with word embeddings because we can take every word's embedding in a document and average the embeddings to create a document embedding
- In other words, it could only take in a fixed input
- But what if we wanted to create to input an arbitrary-length sequence? What if we wanted to input a *sequence of vectors*?

Intuition Behind RNNs

- What if we wanted to also have an output that is a sequence of vectors?
- For example, if we're doing translation, we need to input a sequence of words and get out a sequence of words
- You can't just directly translate everything word for word
  - Word ordering differ across languages
  - Sometimes several words in language A can be expressed as one word in language B
  - Sometimes one word in language A is expressed as several words in language B

Intuition Behind RNNs

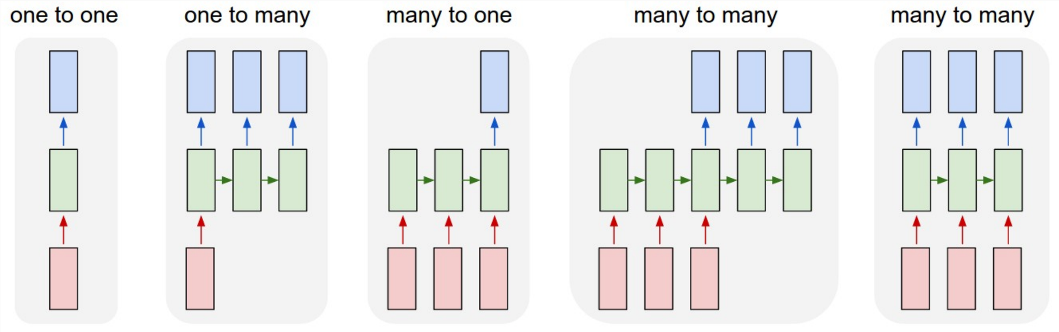


Figure taken from
https://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Intuition of the Mechanism

- At its very core, an RNN operates the same way as a FCNN: it accepts an input vector $\mathbf{x}$ and transforms it into a different set of features; those features and then transformed further until we get an output vector $\mathbf{y}$
- But instead of just transforming features using only the input, we're now transforming an input into a different set of features using previous inputs as well.

Why use RNNs?

- We can now think about modeling *outputs* from a language model
  - With word embeddings, we could only model meaning of words, but we couldn't have word2vec *generate* an output
- Useful for tasks such as translation
- Can also be used for tasks such as classification
- Beyond NLP, it can also be used for tasks such as time-series data
  - Anything that involves a temporal dimension or is sequential

Review Questions

How many inputs can an RNN take?

(A) 1
(B) 2
(C) 3
(D) Arbitrary – this can be determined by the researcher

Review Questions

How many inputs can an RNN take?

(A) 1

(B) 2

(C) 3

(D) Arbitrary – this can be determined by the researcher

The Mechanism of RNNs

The Mechanism of RNNs

- Key idea: RNNs have an "internal state" that is updated as a sequence is processed
- Let $\mathbf{x}$ be a sequence of vectors. You can imagine x_1 being the embedding for the first word, x_2 being the embedding for the second word, x_3 being the embedding for the third word, etc.
- Then,

$$h_t = f_W(h_{t-1}, x_t)$$

Breaking down h_t

$$h_t = f_W\left(h_{t-1}, x_t\right)$$

h_t: the new state

f_W: some function with parameters W

h_{t-1}: the previous state

x_t: input vector at some time step or word step t

Breaking down h_t

$$h_t = f_W(h_{t-1}, x_t)$$

Notice that the same function and the same set of parameters are used at each step

Breaking down h_t: The "Vanilla" RNN

Specifically, we can define h_t as

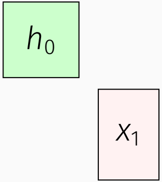
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

We can use this hidden state to then produce an output vector using a linear transformation

$$y_t = W_{hy}h_t + b_y$$

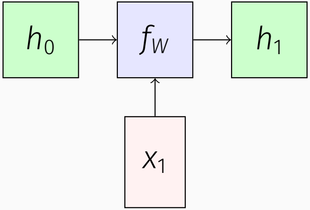
y_t might be, for example, predicting a word

RNN Computational Graph

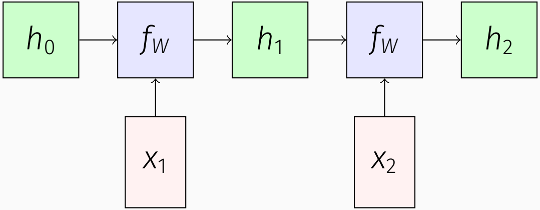


h_0 is the initial state, which can be either set to 0 or learned

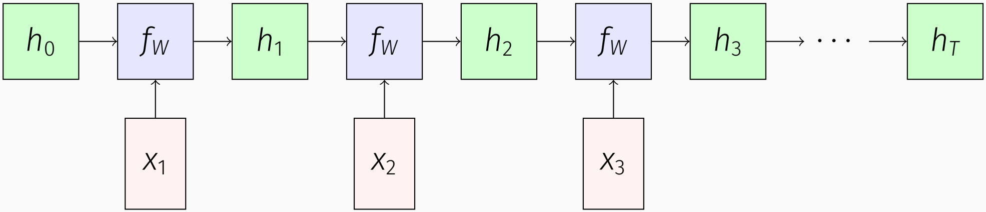
RNN Computational Graph



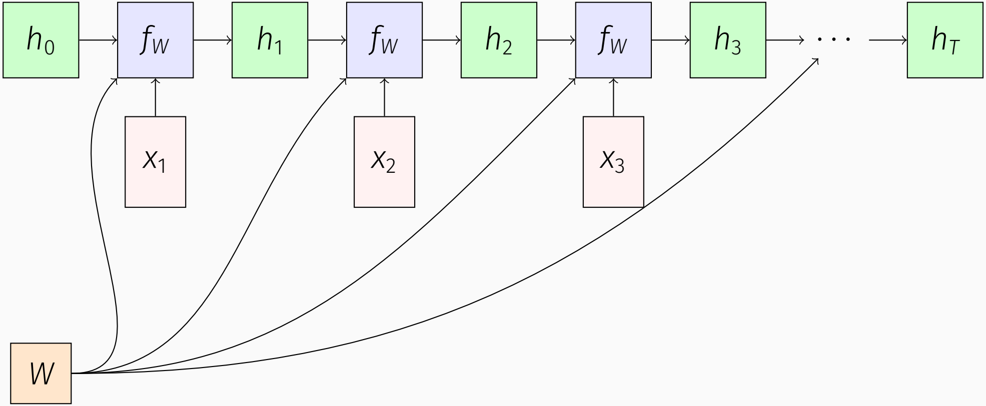
RNN Computational Graph



RNN Computational Graph



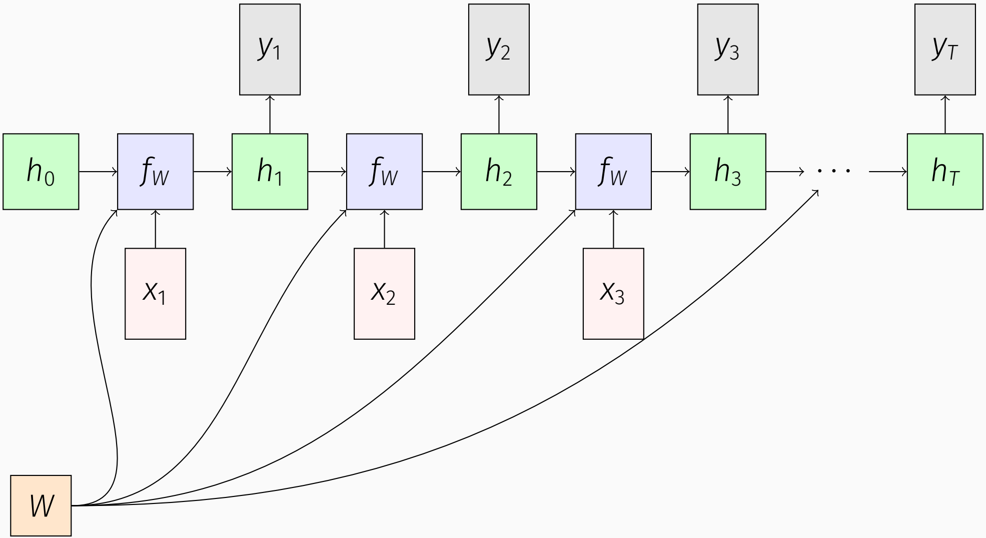
RNN Computational Graph

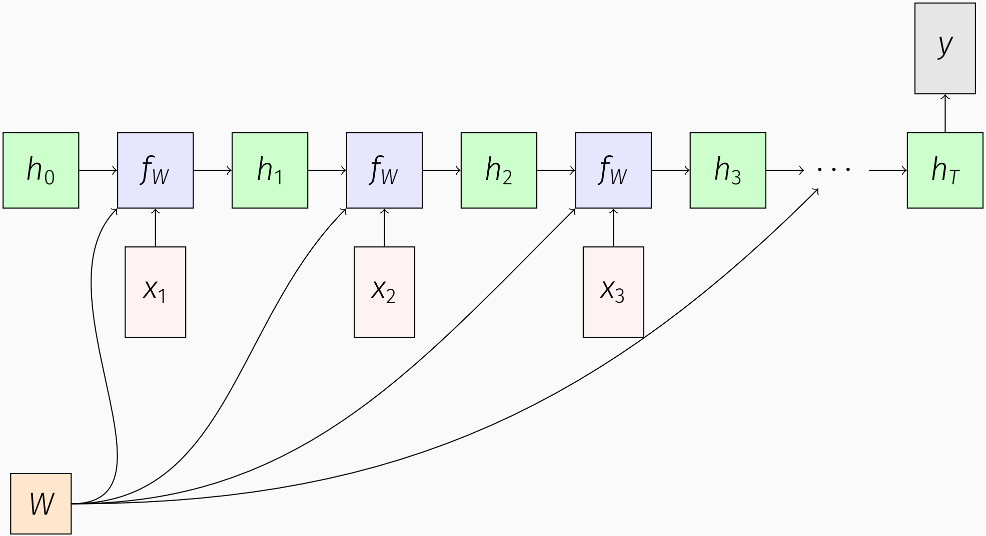


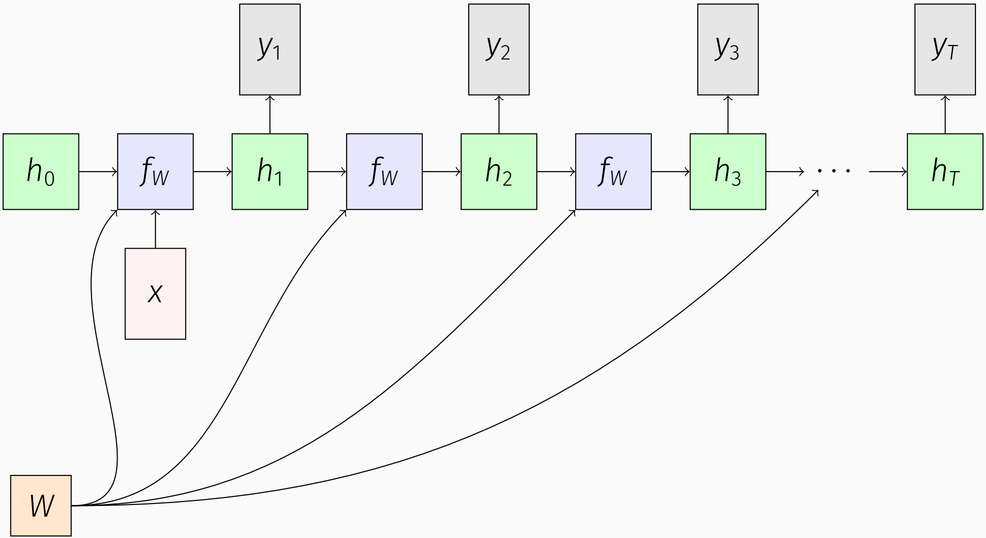
We re-use the **same** weight matrix W_{hh} for each time-step

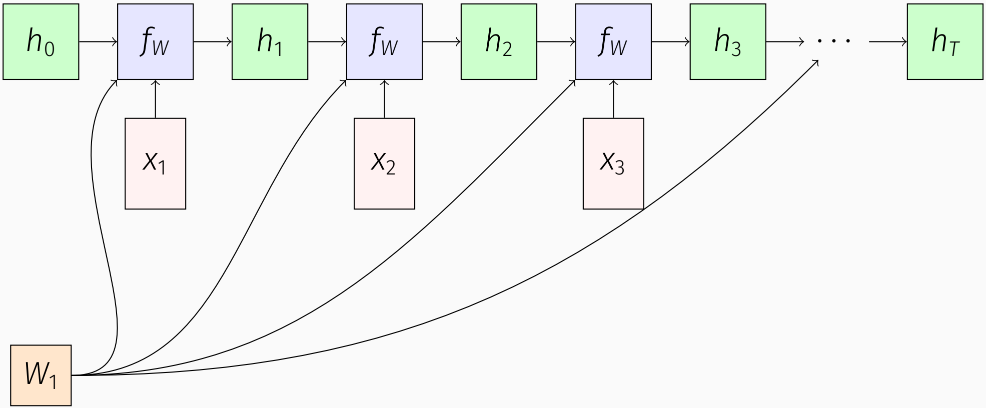
RNNs are Flexible

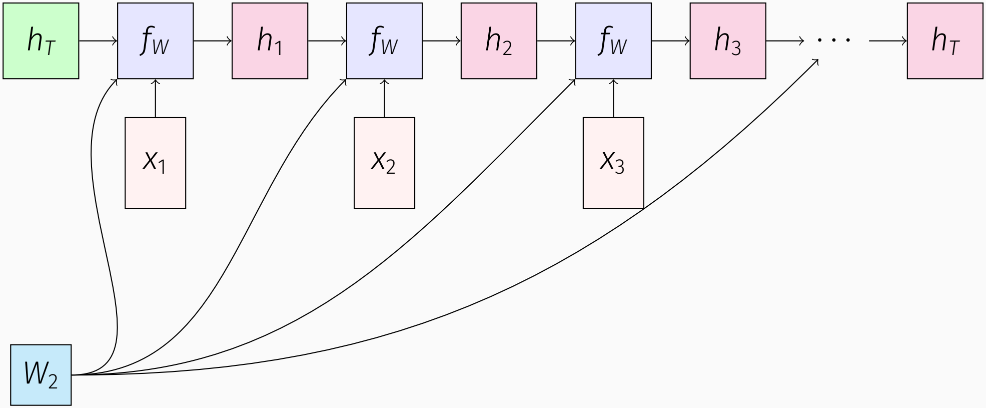
One advantage of RNNs is that they are flexible with inputs and outputs


y_1
y_2
y_3
y_T
h_0
f_W
h_1
f_W
h_2
f_W
h_3
$\cdots$
h_T
x_1
x_2
x_3
W


h_0
f_W
h_1
f_W
h_2
f_W
h_3
$\cdots$
h_T
y
x_1
x_2
x_3
W


y_1
y_2
y_3
y_T
h_0
f_W
h_1
f_W
h_2
f_W
h_3
$\cdots$
h_T
x
W


h_0
f_W
h_1
f_W
h_2
f_W
h_3
$\cdots$
h_T
x_1
x_2
x_3
W_1



$\rightarrow$ so you can stack RNNs, too!

Example: Language Modeling

Training Time

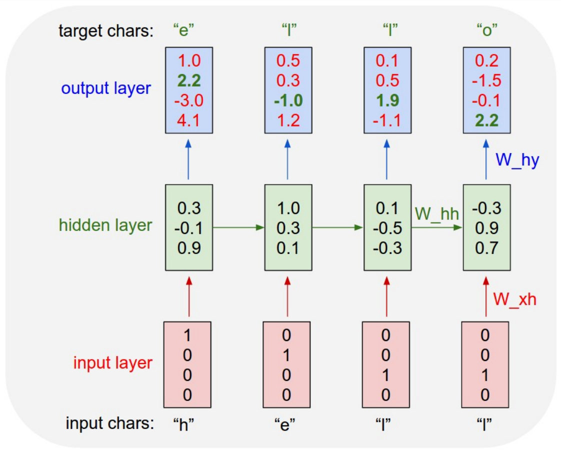
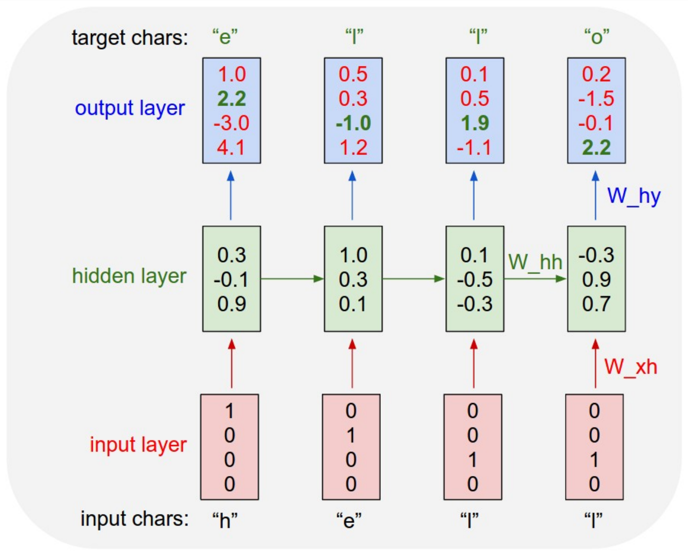
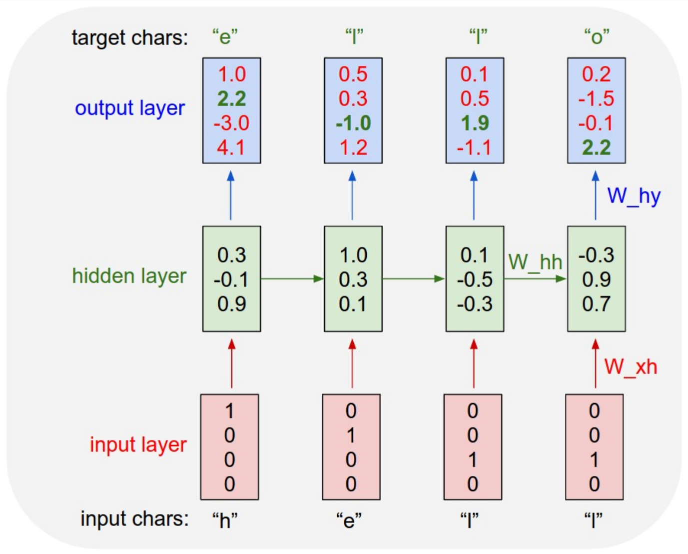


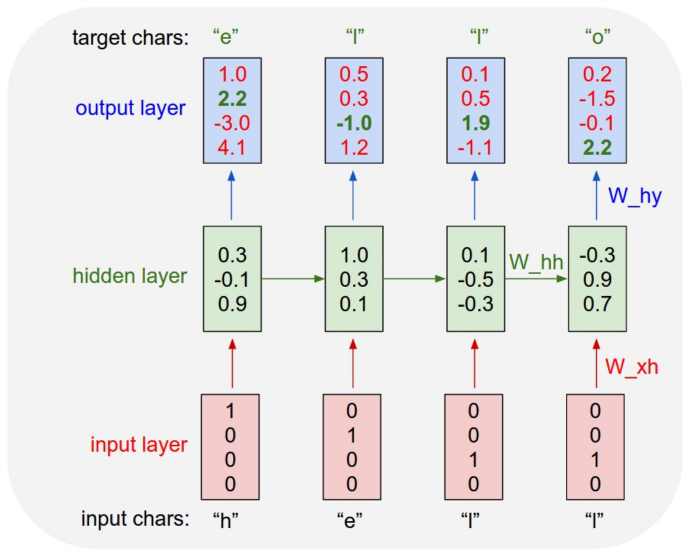
Figure taken from https://karpathy.github.io/2015/05/21/rnn-effectiveness/.



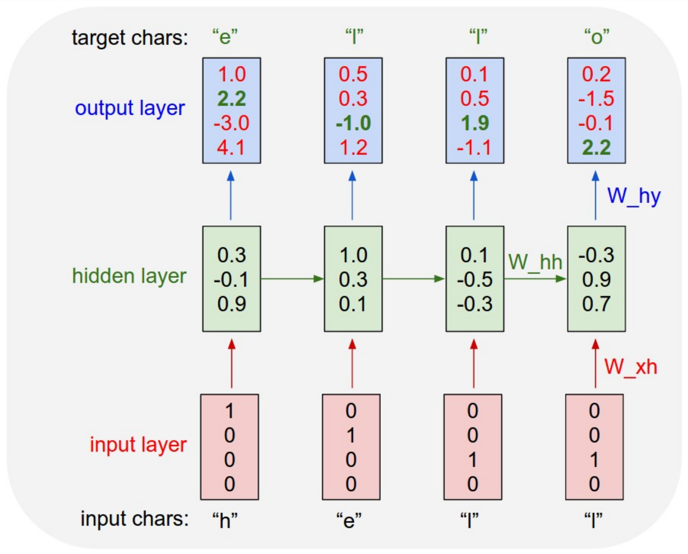
Given ‘h,’ predict ‘e’.



Given ‘he,’ predict ‘l’.

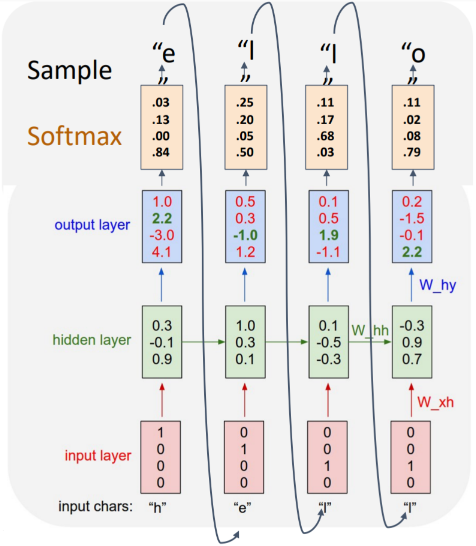


Given 'hel,' predict 'l'.



Given ‘hell,’ predict ‘o’.

Test Time



Sample
"e"
"l"
"l"
"o"
Softmax
.03
.13
.00
.84
.25
.20
.05
.50
.11
.17
.68
.03
.11
.02
.08
.79
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"

Backpropagation

- We can iterate through the entire sequence to calculate the loss (forward)

Backpropagation

- We can iterate through the entire sequence to calculate the loss (forward)
- We can then use backpropagation through the entire sequence to compute the gradients (backward)

Backpropagation

- We can iterate through the entire sequence to calculate the loss (forward)
- We can then use backpropagation through the entire sequence to compute the gradients (backward)
- This can be quite costly with long sequences, so there are techniques that we can use to avoid having to store everything in memory with long sequences

Text Generation Using RNNs

```
QUEENE:
I had thought thou hadst a Roman; for the oracle,
Thus by All bids the man against the word,
Which are so weak of care, by old care done;
Your children were in your holy love,
And the precipitation through the bleeding throne.

BISHOP OF ELY:
Marry, and will, my lord, to weep in such a one were prettiest;
Yet now I was adopted heir
Of the world's lamentable day,
To watch the next way with his father with his face?

ESCALUS:
The cause why then we are all resolved more sons.

VOLUMNIA:
O, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, no, it is no sin it should
And love and pale as any will to that word.

QUEEN ELIZABETH:
But how long have I heard the soul for this world,
And show his hands of life be proved to stand.
```

Issues with RNNs

Issues with RNNs

- RNNs struggle to learn long-term dependencies

Issues with RNNs

- RNNs struggle to learn long-term dependencies
- Caused by
  - Vanishing Gradients: as a gradient is backpropagated through many time steps, it tends to get smaller
  - Exploding Gradients: gradients can also grow exponentially, which causes major shifts in weights

Issues with RNNs

- RNNs struggle to learn long-term dependencies
- Caused by
  - Vanishing Gradients: as a gradient is backpropagated through many time steps, it tends to get smaller
  - Exploding Gradients: gradients can also grow exponentially, which causes major shifts in weights
- RNNs also have limited memory: the fixed-sized hidden state can be a bottleneck for storing information

Long Short-Term Memory (LSTM)

- Attempts to alleviate the vanishing gradients problem

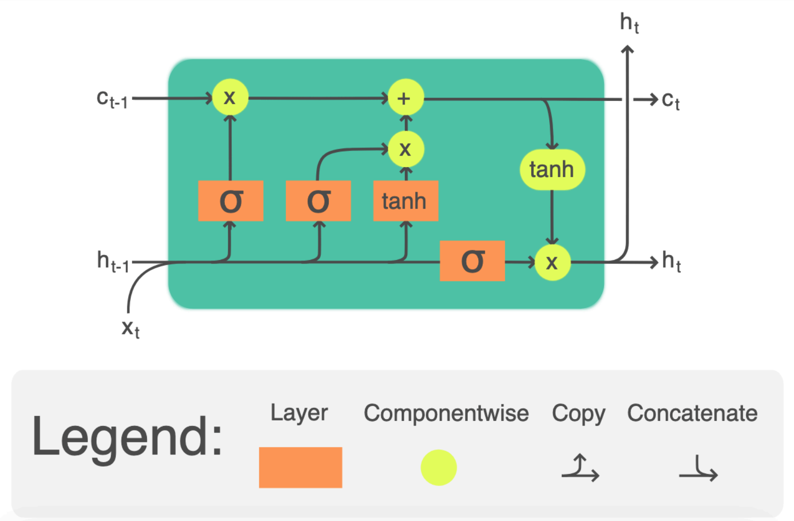
Long Short-Term Memory (LSTM)

- Attempts to alleviate the vanishing gradients problem
- We won't discuss the technical aspects of this, but LSTMs are now the standard when people are using RNNs

Long Short-Term Memory (LSTM)

- Attempts to alleviate the vanishing gradients problem
- We won't discuss the technical aspects of this, but LSTMs are now the standard when people are using RNNs
- Vanilla RNNs are simply not as effective as LSTMs

LSTM



h_t
c_{t-1}
x
+
c_t
x
tanh
σ
σ
tanh
h_{t-1}
σ
x
h_t
x_t
Legend:
Layer
Componentwise
Copy
Concatenate

Concluding Lecture

Takeaways

- RNNs are a way to deal with sequential input data (like text!)

Takeaways

- RNNs are a way to deal with sequential input data (like text!)
- They are, in a way, unreasonably effective
  - You use the same weight matrix W_{hh} and W_{xh} for each time-step
  - There is **not** a specific weight matrix for each step

Takeaways

- RNNs are a way to deal with sequential input data (like text!)
- They are, in a way, unreasonably effective
  - You use the same weight matrix W_{hh} and W_{xh} for each time-step
  - There is **not** a specific weight matrix for each step
- They are very flexible
  - One to one
  - One to many
  - Many to one
  - Many to many

Takeaways

- RNNs are a way to deal with sequential input data (like text!)
- They are, in a way, unreasonably effective
    - You use the same weight matrix W_{hh} and W_{xh} for each time-step
    - There is **not** a specific weight matrix for each step
- They are very flexible
    - One to one
    - One to many
    - Many to one
    - Many to many
- Allows us to start *generating* text

Next Class...

- Showing some code
    - But these days, RNNs aren't used as much now
- LSTMs don't solve everything, which led to the development of the attention mechanism
- Attention is the core mechanism of transformers