Natural Language Processing

Deep Learning for Text Classification

BITTIGER

The Lifelong Learning Platform of Silicon Valley

Copyright Policy

Outline

BITTIGER

1. Processing natural language data
2. Recurrent neural networks
3. Sentiment analysis as classification
4. Case study: sentiment of IMDB movie reviews

Processing Language Data

Representation of Words

BITTIGER

- Traditionally: one-hot, tf-idf, …
  - Sparse

0 0 0 …... 0 1 0 0 … 0

Size = vocabulary size ^{2}
(e.g., ~30000^2)

- Recently: word embedding
  - Dense

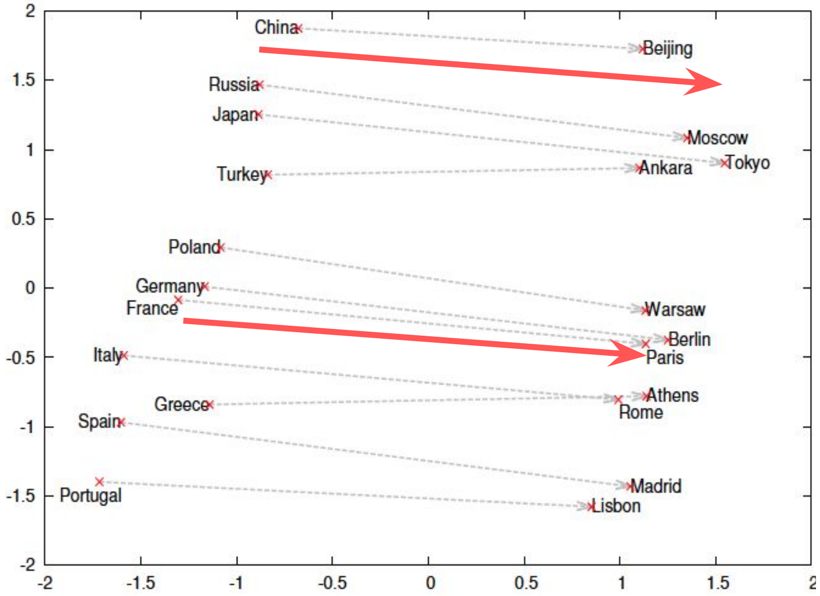
0.04 0.05 …

Size = a user-defined smaller size X vocabulary size
(e.g., ~300 X 30000)

Advantages

BITTIGER

- Not exclusive to deep learning
- Reduced feature size
- Vector arithmetics:
  - “Paris”- “France” + “China” = ?
- Similarity: cosine distance



[Source: Mikolov et al, NIPS’13]

Usage

- Major deep learning toolkits have it conveniently built in!
- Keras (& tensorflow)
  - `keras.layers.Embedding(vocab_size, embedding_size)`
- Input: sequence of word ID
- Output: sequence of word embeddings
- You must create a dictionary & decide its size

Design Challenges (I)

- Size of vocabulary
- Add special words in dictionary: <pad>, <sos>, <eos>, <unk>
- Tokenize and stemming: pros and cons

  Pros: reduce size, group words with similar meanings

  “read”, “reader”, “reading” -> “read”

  Cons: ambiguity, special cases

  “U.S.”, “White House”, “ / and - ”,

Design Challenges (II)

- Length: usually cropped to e.g. 100 words
- Noisy:
  - special invisible symbols → heuristic processing
  - repeated symbols → spaces, !!!!.....,
  - Emoji
  - URLs
- Characters vs. words?

Recurrent Neural Networks (RNN)

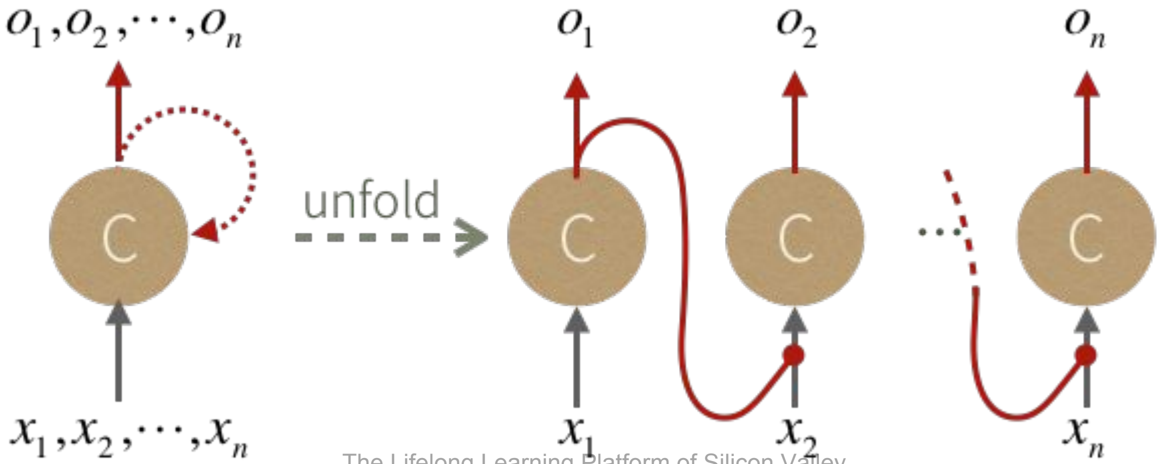
BITTIGER

The Lifelong Learning Platform of Silicon Valley

Recurrent Neural Network: Basics

BITTIGER

- The same cell *C* is used recurrently (repeatedly)
- A memory (state) is kept in *C* that carries information



RNN Applications

BITTIGER

Almost any **sequential** data!

- NLP: NER, POS tagging, translation, sentiment analysis
- Numerical data: temperature, traffic, air quality...

INPUT	今	天	小	明	心	情	很	好
NER	X	X	PER	PER	X	X	X	X
POS	ND	ND	NB	NB	NA	NA	D	VH

RNN Architectures

Major types:

- Many-to-one: Sentiment analysis, topic classification...
- Many-to-many: NER, Translation...
- One-to-many: NER, Translation...
- One-to-one: *are you kidding?*

?? We will explain later!

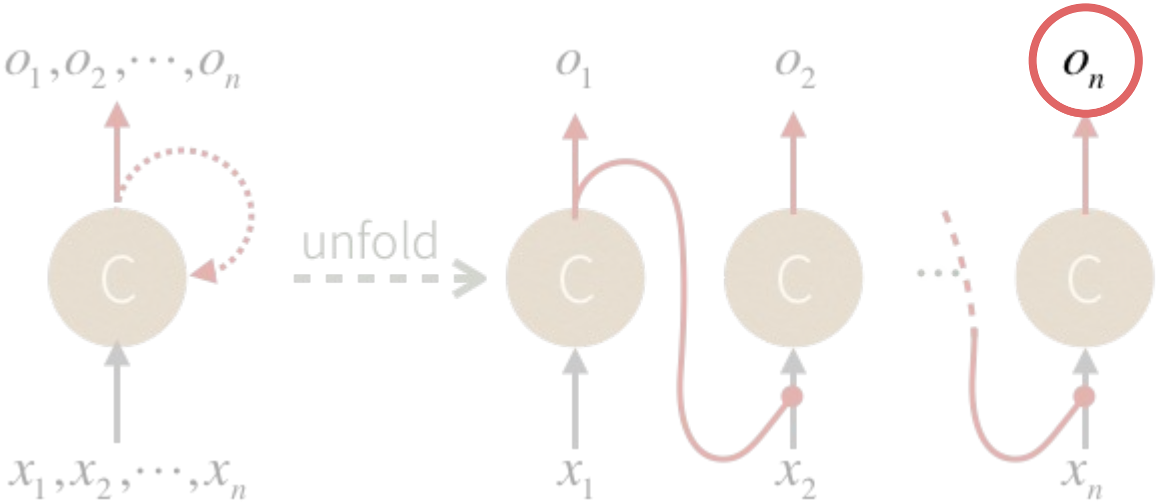
Many-to-one (I)

Major types:

- Many-to-one: Sentiment analysis

INPUT	今	天	小	明	心	情	很	好
OUTPUT								P

Many-to-one (II)



- Only calculate loss against **one** output

Many-to-many (I)

BITTIGER

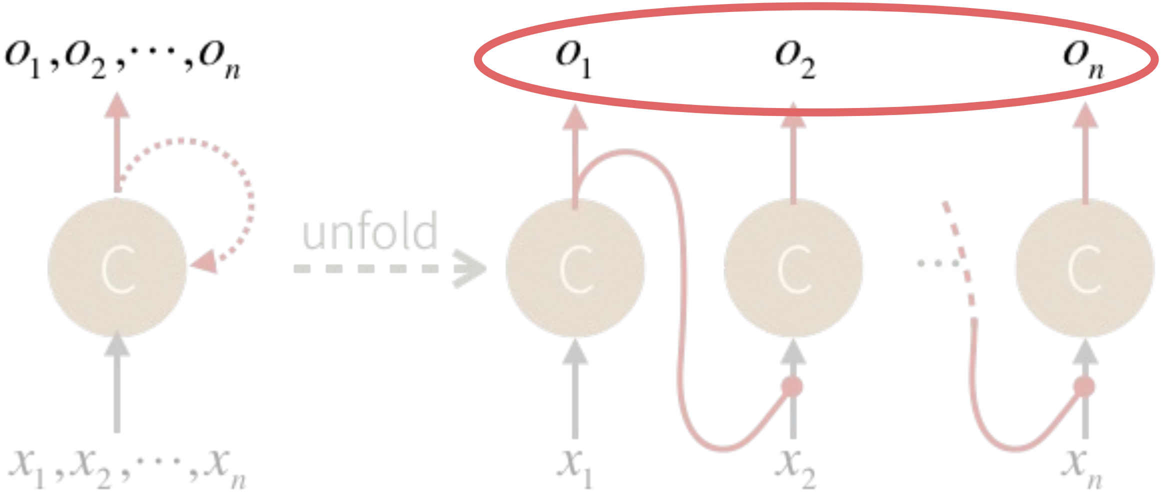
Major types:

- Many-to-many: NER

INPUT	今	天	小	明	心	情	很	好
OUTPUT	X	X	PER	PER	X	X	X	X

Many-to-many (II)

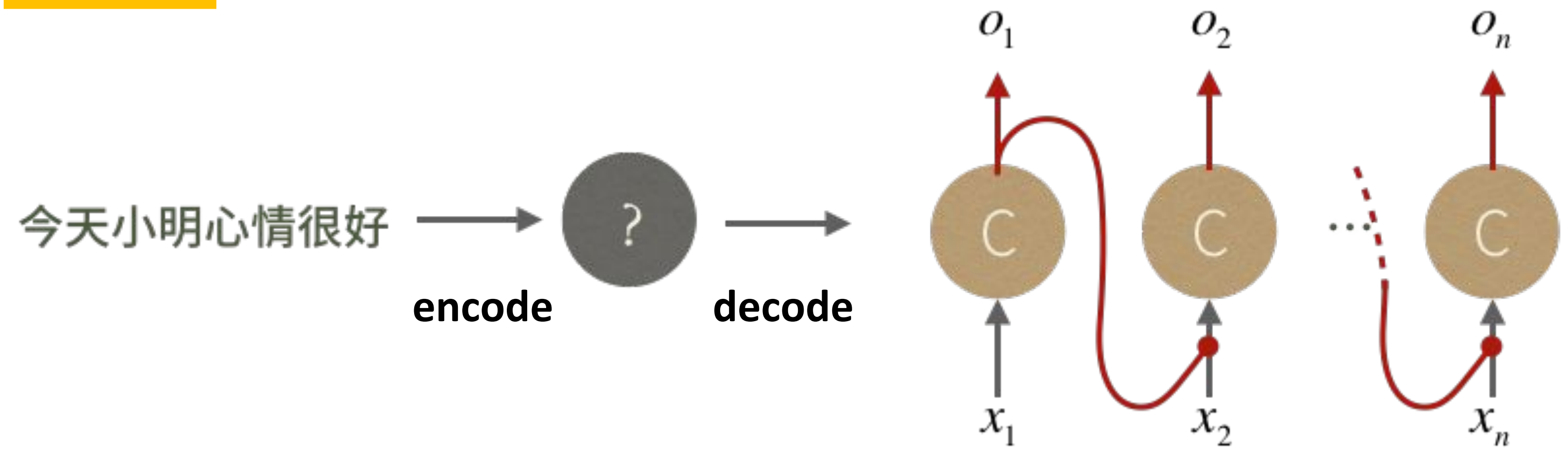
BITTIGER



- Calculate loss against the **entire sequence**

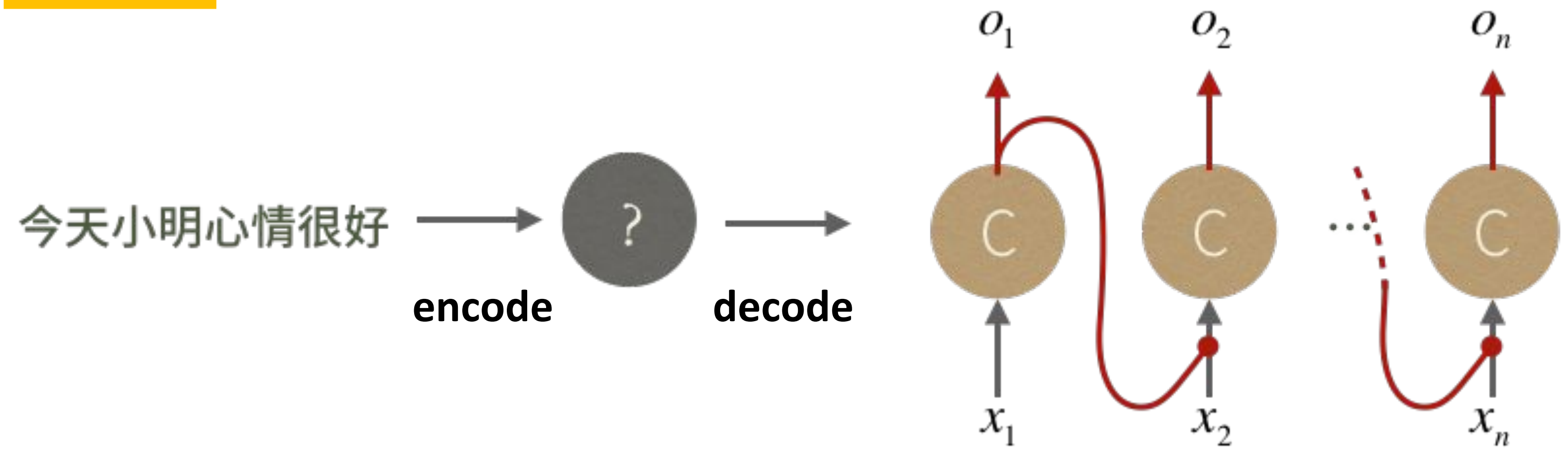
One-to-Many (I)

BITTIGER



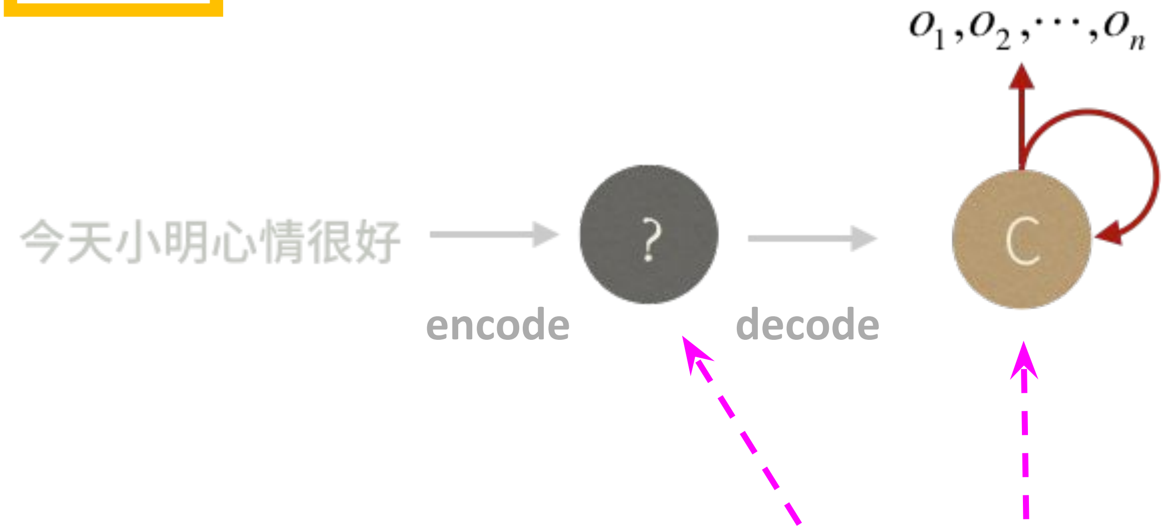
- First, **encode** the sequence into one vector
- Second, **decode** the vector into another sequence

One-to-Many (II)



- Calculate loss against the **entire sequence**
- Back-propagation will also train the encoder

One-to-Many (III)



- Two different networks here, **encoder** and **decoder**
- They can have different architectures! e.g., CNN + RNN

Sequence to sequence (seq2seq)

BITTIGER

- Both input and output are sequences
- Also called “encoder-decoder”
- Can be 1-to-m or m-to-m
- Can have different network architectures
- State-of-the-art in many NLP tasks
- We will talk more about them in Part II and III of this course

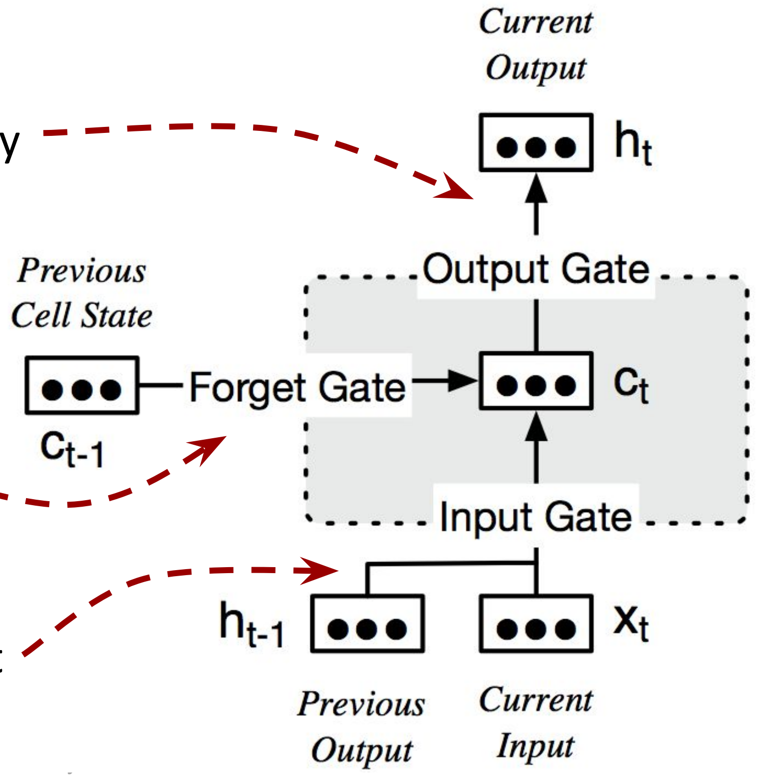
LSTM: concepts

Long Short-Term Memory

- Main concept: learning to forget
- Three **gates**: **input**, **output**, and **forget**
- Gates regulate the values of the input, output, and memory
- Value of the gates are learnable parameters of the model

An LSTM cell

- **Output**: regulate the output from memory
- **Forget**: regulate previous memory
- **Input**: regulate input and previous output



Learning to forget (and more)

BITTIGER

- ***W*** and ***U***: learned weights
- Gate values {0,1}, remember sigmoids?
- So, LSTM “learn” how to modulate gates from the value of input **x** and **h**
- New memory **c** = forget some old memory + input some new memory

$$\left\{\begin{aligned} \mathbf{i}_t &= \sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1} + \mathbf{b}_i) \\ \mathbf{f}_t &= \sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1} + \mathbf{b}_f) \\ \mathbf{o}_t &= \sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1} + \mathbf{b}_o) \end{aligned}\right\}$$

$$\tilde{\mathbf{c}}_t = \tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$

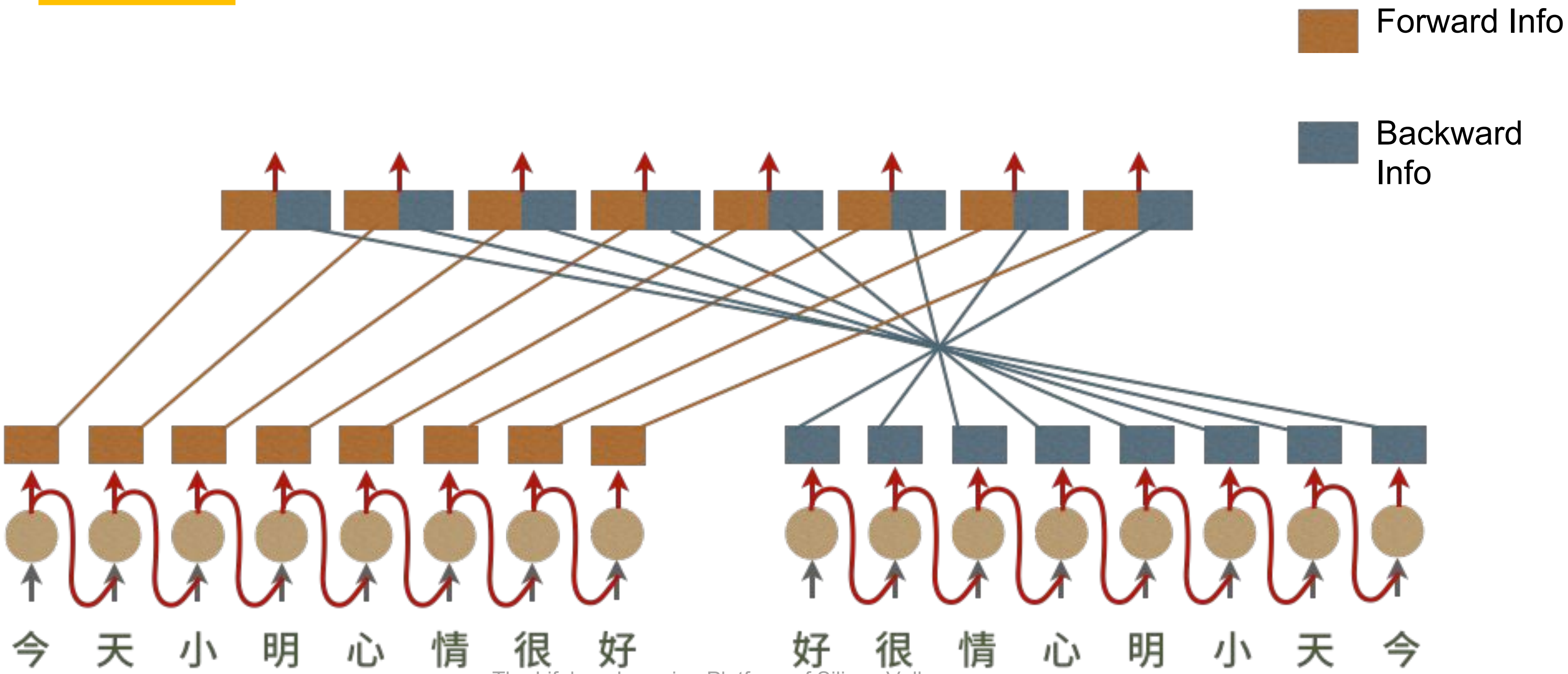
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

Design Challenges: RNN

- Length!
  - Obviously, RNNs will not work for very long sequences
- Direction
  - Use **bidirectional** RNN to attempt to learn long sequences
- Depth
  - Can stack multiple RNNs
- Speed
  - Slow! Why?

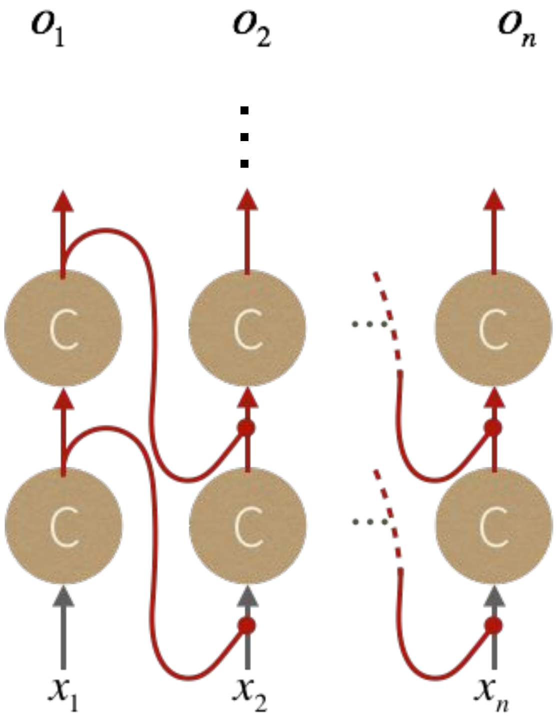
Bidirectional RNN

BITTIGER



Deep RNN

BITTIGER



The Lifelong Learning Platform of Silicon Valley

Speed of RNN

BITTIGER

- Factors: sequence length and size of the RNN cell
    - The step ***t+1*** of the RNN cannot be computed until step ***t*** is completed
    - Size of RNN cell determines the number of parameters

Sentiment Analysis as Classification

Sequence Classification

- Many-to-one architecture
  - Input a sequence, obtain one class label
- In our sentiment analysis project: input a movie review, predict its sentiment as being “positive” or “negative”
- We will go to the code now

Code

BITTIGER

The Lifelong Learning Platform of Silicon Valley