

STOR566: Introduction to Deep Learning

Lecture 9: Recurrent Neural Networks

Yao Li
UNC Chapel Hill

Sep 19, 2024

Materials are from *Deep Learning (UCLA)*

Representation for sentence/document

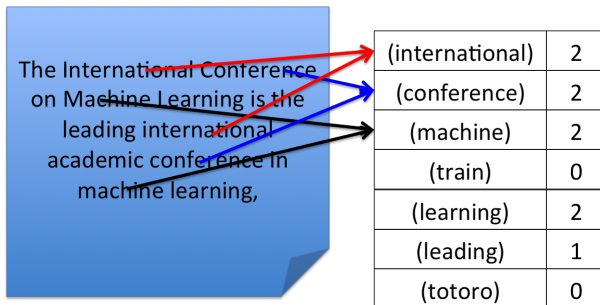
Bag of Words

- A classical way to represent NLP data
- Text \rightarrow Vector/Matrices
- Problem: text length not fixed

Bag of Words

- A classical way to represent NLP data
- Text \rightarrow Vector/Matrices
- Problem: text length not fixed
- Bag of words:

Sentence $\rightarrow d$ -dimensional vector \mathbf{x}



d = number of potential words (very large)

Bag of Words: Processing Steps

- Step 1: Collect Data

*It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,*

Bag of Words: Processing Steps

- Step 1: Collect Data

*It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,*

- Step 2: Build the Vocabulary

{it, was, the, best, of, times, worst, age, wisdom, foolishness}

Bag of Words: Processing Steps

- Step 1: Collect Data

*It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,*

- Step 2: Build the Vocabulary

{it, was, the, best, of, times, worst, age, wisdom, foolishness}

- Step 3: Create Document/Sentence Vectors

	it	was	the	best	of	times	worst	age	wisdom	foolishness
d_1	1	1	1	1	1	1	0	0	0	0
d_2	1	1	1	0	1	1	1	0	0	0
d_3	1	1	1	0	1	0	0	1	1	0
d_4	1	1	1	0	1	0	0	1	0	1

Bag of n -gram

- Bag of n -gram features ($n = 2$):

The international conference on Machine Learning is the leading international academic conference in machine learning,

(international)	2
(conference)	2
(machine)	2
(train)	0
(learning)	2
(leading)	1
(totoro)	0

(international conference)	1
(machine learning)	2
(leading international)	1
(totoro tiger)	0
(tiger woods)	0
(international academic)	1
(academic conference)	1

TF-IDF

- Use the bag-of-words matrix or the normalized version (TF-IDF) for a dataset (denoted by D):

$$\text{tfidf}(\text{doc}, \text{word}, D) = \text{tf}(\text{doc}, \text{word}) \cdot \text{idf}(\text{word}, D)$$

- $\text{tf}(\text{doc}, \text{word})$: term frequency

(word count in the document)/(total number of terms in the document)

- $\text{idf}(\text{word}, \text{Dataset})$: inverse document frequency
 $\log((\text{Number of documents})/(\text{Number of documents with this word}))$

Data Matrix (document)

$$\text{tfidf}(\text{doc}, \text{word}, D) = \text{tf}(\text{doc}, \text{word}) \cdot \text{idf}(\text{word}, D)$$

TF = (word count in the doc)/(total number of terms in the doc)

IDF = $\log((\text{Number of docs})/(\text{Number of docs with this word}))$

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

tf-idf

	angeles	los	new	post	times	york
d1	0	0	$\frac{1}{3} \times \log\left(\frac{3}{2}\right) = 0.135$	0	0.135	0.135
d2	0	0	0.135	$\frac{1}{3} \times \log(3) = 0.366$	0	0.135
d3	0.366	0.366	0	0	0.135	0

Bag of word + linear model

- Example: text classification (e.g., sentiment prediction, review score prediction)
- Linear model: $y \approx \text{sign}(\mathbf{w}^T \mathbf{x})$
(e.g., by linear SVM/logistic regression)
- w_i : the “contribution” of each word

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
-------	-------	-------	-------	-------	-------	-------	-------	-------

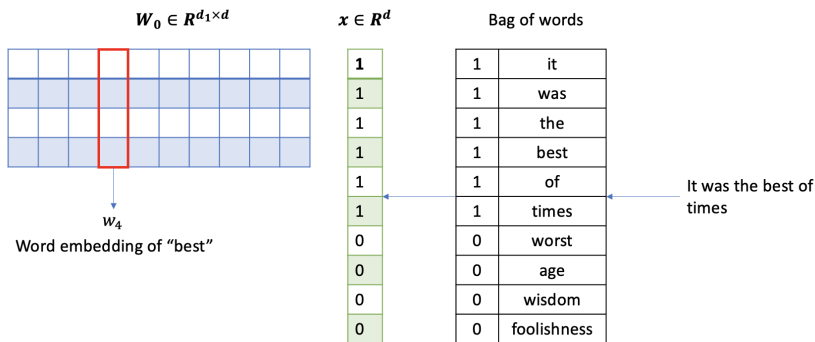
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9

Bag of word + Fully connected network

- $f(\mathbf{x}) = W_L \sigma(W_{L-1} \cdots \sigma(W_0 \mathbf{x}))$

Bag of word + Fully connected network

- $f(\mathbf{x}) = W_L \sigma(W_{L-1} \cdots \sigma(W_0 \mathbf{x}))$



- W_0 is also called the **word embedding matrix**
- \mathbf{w}_i : d_1 dimensional representation of i -th word
- $W_0 \mathbf{x} = x_1 \mathbf{w}_1 + x_2 \mathbf{w}_2 + \cdots + x_d \mathbf{w}_d$
is a **linear combination** of these vectors

Recurrent Neural Network

Time Series/Sequence Data

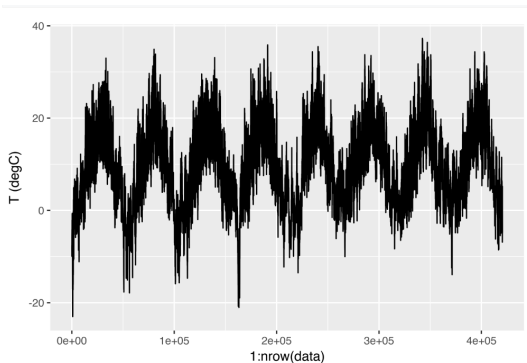
- Input: $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$
 - Each \mathbf{x}_t is the feature at time step t
 - Each \mathbf{x}_t can be a d -dimensional vector
- Output: $\{y_1, y_2, \dots, y_T\}$
 - Each y_t is the output at step t
 - Multi-class output or Regression output:

$$y_t \in \{1, 2, \dots, L\} \quad \text{or} \quad y_t \in \mathbb{R}$$

- Translation: $\mathbf{y}_t \in \mathbb{R}^d$

Example: Time Series Prediction

- Climate Data:
 - x_t : temperature at time t
 - y_t : temperature (or temperature change) at time $t + 1$
- Stock Price: Predicting stock price



Example: Language Modeling

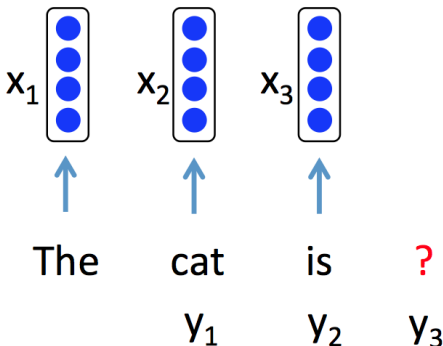
The cat is ?

Example: Language Modeling

The cat is ?

- \mathbf{x}_t : one-hot encoding to represent the word at step t
([0, ..., 0, 1, 0, ..., 0])
- y_t : the next word

$y_t \in \{1, \dots, V\}$ V : Vocabulary size

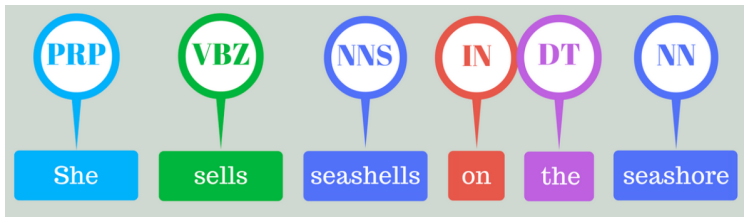


Example: POS Tagging

- Part of Speech Tagging:

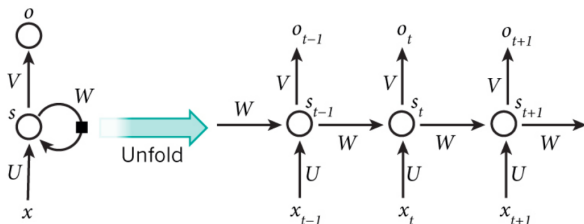
Labeling words with their Part-Of-Speech (Noun, Verb, Adjective, ...)

- x_t : a **vector** to represent the word at step t
- y_t : label of word t



picture from <https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31>

Recurrent Neural Network (RNN)



- x_t : t -th input
- s_t : hidden state at time t ("memory" of the network)

$$s_t = f(Ux_t + Ws_{t-1})$$

W : transition matrix, U : word embedding matrix

s_0 usually set to be 0, f : activation function

- Output at time t :

$$o_t = Vs_t$$

Recurrent Neural Network (RNN)

- Training: Find U, W, V to minimize empirical loss:
- Loss of a sequence:

$$\sum_{t=1}^T \text{loss}(V\mathbf{s}_t, y_t)$$

(\mathbf{s}_t is a function of U, W, V)

Recurrent Neural Network (RNN)

- Training: Find U, W, V to minimize empirical loss:
- Loss of a sequence:

$$\sum_{t=1}^T \text{loss}(V\mathbf{s}_t, y_t)$$

(\mathbf{s}_t is a function of U, W, V)

- Loss on a batch:
Average loss over all sequences in a batch
- Solved by SGD/Adam

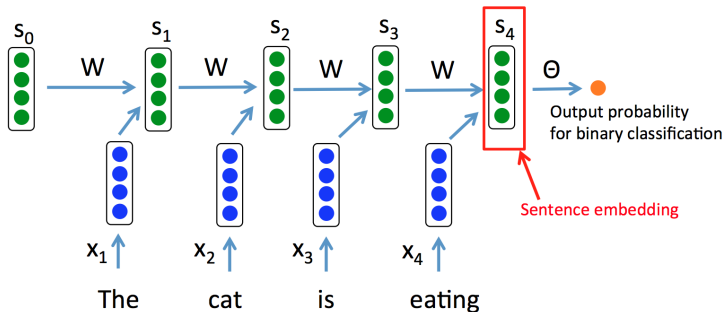
RNN: Text Classification

- Not necessary to output at each step
- Text Classification:

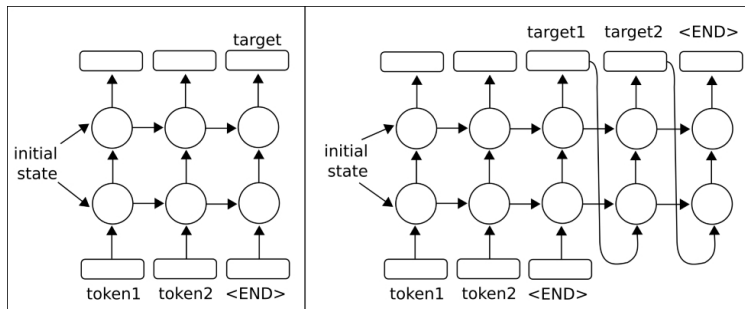
sentence \rightarrow category

Output only at the final step

- Model: add a fully connected network to the final embedding



Multi-layer RNN



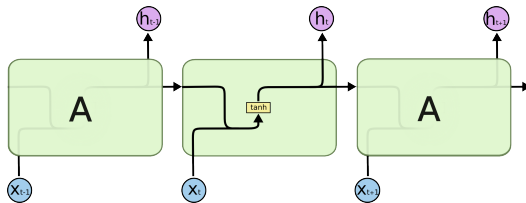
(Figure from https://subscription.packtpub.com/book/big_data_and_business_intelligence)

Problems of Classical RNN

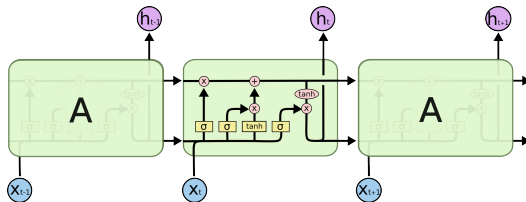
- Hard to capture long-term dependencies
- Hard to solve (vanishing gradient problem)
- Solution:
 - LSTM (Long Short Term Memory networks)
 - GRU (Gated Recurrent Unit)
 - ...

LSTM

- RNN:



- LSTM:



Neural Network
Layer

Pointwise
Operation

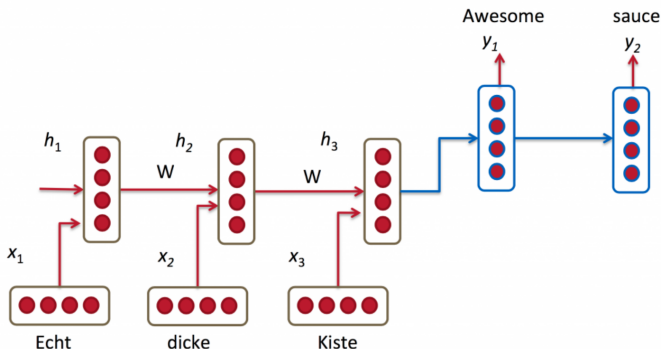
Vector
Transfer

Concatenate

Copy

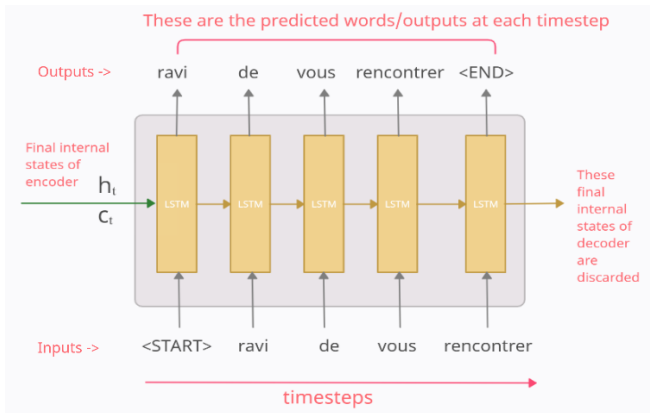
Neural Machine Translation (NMT)

- Out the translated sentence from an input sentence
- Training data: a set of input-output pairs (supervised setting)
- Encoder-decoder approach:
 - Encoder: Use (RNN/LSTM) to encode the input sentence input a latent vector
 - Decoder: Use (RNN/LSTM) to generate a sentence based on the latent vector



RNN: Neural Machine Translation

- Start input of the decoder?
- When to stop?



picture from <https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186bf49b>

Problems

- Only the last hidden state is used in decoding.
- Do not work well on long sequences.
- Solution:
 - Attention Mechanism:

How about if we give a vector representation from every encoder step to the decoder model?

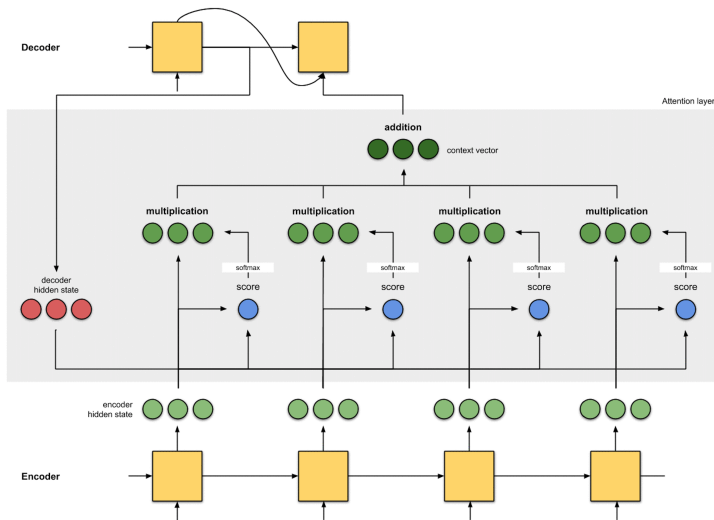
Attention in NMT

- Usually, each output word is only related to a subset of input words (e.g., for machine translation)
- Let \mathbf{u} be the **current decoder hidden state**
 $\mathbf{v}_1, \dots, \mathbf{v}_n$ be the **hidden state for each input word**
- Compute the weight of each state by

$$\mathbf{p} = \text{Softmax}(\mathbf{u}^T \mathbf{v}_1, \dots, \mathbf{u}^T \mathbf{v}_n)$$

- Compute the context vector by $V\mathbf{p} = p_1 \mathbf{v}_1 + \dots + p_n \mathbf{v}_n$

Attention in NMT



(Figure from <https://towardsdatascience.com/neural-machine-translation-nmt-with-attention-mechanism>)

Conclusions

- Bag of words
- RNN
- Attention in NMT

Questions?