

# Contextualized Word Embeddings

## CS114B Lab 12

Kenneth Lai

April 30, 2021

# Contextualized Word Embeddings

## CS114B Lab 12

Kenneth Lai

April 30, 2021



Source 1



Source 2

# Contextualized Word Embeddings

## CS114B Lab 12

Kenneth Lai

April 30, 2021



Source 1



Source 3

# Word Embeddings

- ▶ Distributed representations of words

# Word Embeddings

- ▶ Distributed representations of words
  - ▶ Sparse vectors

# Word Embeddings

- ▶ Distributed representations of words
  - ▶ Sparse vectors
    - ▶ One-hot, tf-idf, PPMI, etc.

# Word Embeddings

- ▶ Distributed representations of words
  - ▶ Sparse vectors
    - ▶ One-hot, tf-idf, PPMI, etc.
  - ▶ Dense vectors

# Word Embeddings

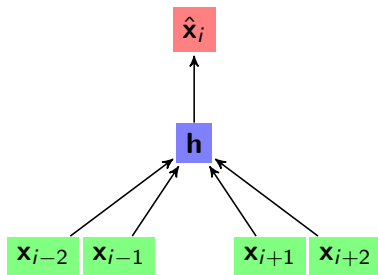
- ▶ Distributed representations of words
  - ▶ Sparse vectors
    - ▶ One-hot, tf-idf, PPMI, etc.
  - ▶ Dense vectors
    - ▶ SVD, word2vec, etc.



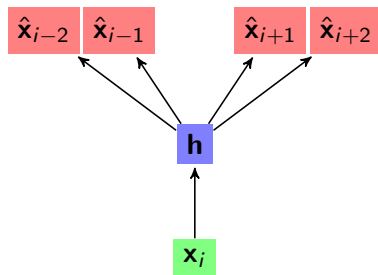
# word2vec

- ▶ Based on a feedforward neural network language model

- Based on a feedforward neural network language model

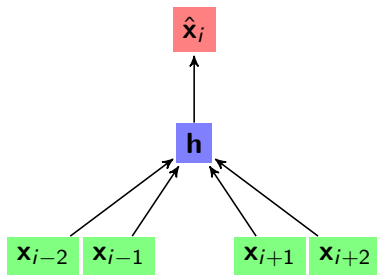


CBOW

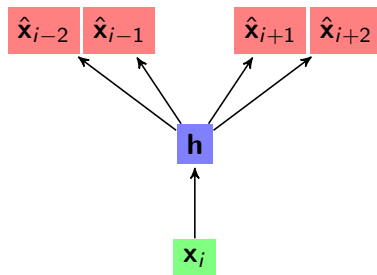


Skip-gram

- Based on a feedforward neural network language model



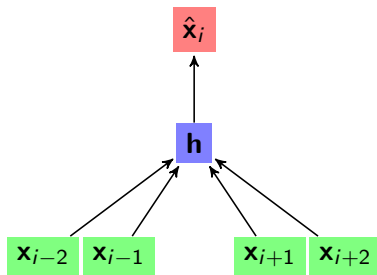
CBOW



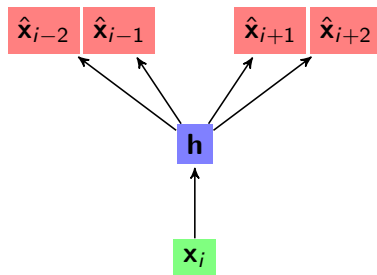
Skip-gram

- CBOW: use context to predict current word

- Based on a feedforward neural network language model



CBOW



Skip-gram

- CBOW: use context to predict current word
- Skip-gram: use current word to predict context

## word2vec

- ▶ Input layer: one-hot word vectors

## word2vec

- ▶ Input layer: one-hot word vectors
- ▶ Hidden (projection) layer: identity (linear!) activation function, no bias

## word2vec

- ▶ Input layer: one-hot word vectors
- ▶ Hidden (projection) layer: identity (linear!) activation function, no bias
  - ▶ Input  $\rightarrow$  hidden = table lookup (in weight matrix)

## word2vec

- ▶ Input layer: one-hot word vectors
- ▶ Hidden (projection) layer: identity (linear!) activation function, no bias
  - ▶ Input  $\rightarrow$  hidden = table lookup (in weight matrix)
- ▶ Output layer: softmax activation function



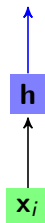
## word2vec

- ▶ Input layer: one-hot word vectors
- ▶ Hidden (projection) layer: identity (linear!) activation function, no bias
  - ▶ Input  $\rightarrow$  hidden = table lookup (in weight matrix)
- ▶ Output layer: softmax activation function
- ▶ Word embeddings: output of hidden layer (= hidden layer weights corresponding to input word)

## word2vec

- ▶ Input layer: one-hot word vectors
- ▶ Hidden (projection) layer: identity (linear!) activation function, no bias
  - ▶ Input  $\rightarrow$  hidden = table lookup (in weight matrix)
- ▶ Output layer: softmax activation function
- ▶ Word embeddings: output of hidden layer (= hidden layer weights corresponding to input word)

word embedding



# Polysemy

- ▶ One vector per word type

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ a **mouse**<sup>1</sup> controlling a computer system in 1968

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ a **mouse**<sup>1</sup> controlling a computer system in 1968
  - ▶ a quiet animal like a **mouse**<sup>2</sup>

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ a **mouse**<sup>1</sup> controlling a computer system in 1968
  - ▶ a quiet animal like a **mouse**<sup>2</sup>
- ▶ Should **mouse**<sup>1</sup> and **mouse**<sup>2</sup> have the same word embedding?

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ a **mouse**<sup>1</sup> controlling a computer system in 1968
  - ▶ a quiet animal like a **mouse**<sup>2</sup>
- ▶ Should **mouse**<sup>1</sup> and **mouse**<sup>2</sup> have the same word embedding?
  - ▶ Why not?



# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ ... mouse<sup>1</sup> ... computer ...
  - ▶ ... animal ... mouse<sup>2</sup> ...
- ▶ Should mouse<sup>1</sup> and mouse<sup>2</sup> have the same word embedding?
  - ▶ Why not?
    - ▶ Syntagmatic association between mouse<sup>1</sup> and computer

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ ... mouse<sup>1</sup> ... computer ...
  - ▶ ... animal ... mouse<sup>2</sup> ...
- ▶ Should mouse<sup>1</sup> and mouse<sup>2</sup> have the same word embedding?
  - ▶ Why not?
    - ▶ Syntagmatic association between mouse<sup>1</sup> and computer
    - ▶ Syntagmatic association between mouse<sup>2</sup> and animal

# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ ... mouse<sup>1</sup> ... computer ...
  - ▶ ... animal ... mouse<sup>2</sup> ...
- ▶ Should mouse<sup>1</sup> and mouse<sup>2</sup> have the same word embedding?
  - ▶ Why not?
    - ▶ Syntagmatic association between mouse<sup>1</sup> and computer
    - ▶ Syntagmatic association between mouse<sup>2</sup> and animal
    - ▶ Paradigmatic association between computer and animal!

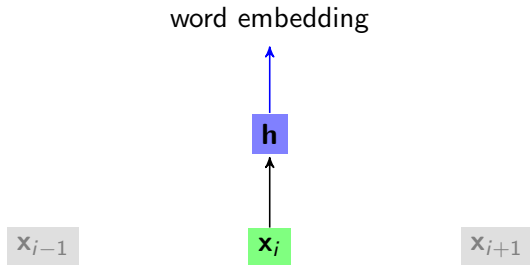
# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ ... mouse<sup>1</sup> ... computer ...
  - ▶ ... animal ... mouse<sup>2</sup> ...
- ▶ Should mouse<sup>1</sup> and mouse<sup>2</sup> have the same word embedding?
  - ▶ Why not?
    - ▶ Syntagmatic association between mouse<sup>1</sup> and computer
    - ▶ Syntagmatic association between mouse<sup>2</sup> and animal
    - ▶ Paradigmatic association between computer and animal!
- ▶ How can we distinguish between mouse<sup>1</sup> and mouse<sup>2</sup>?

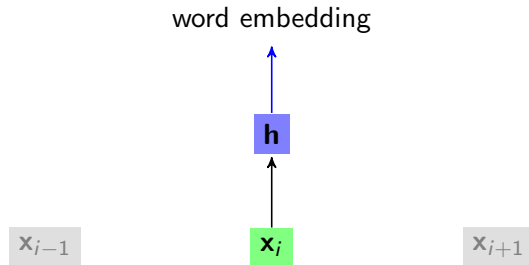
# Polysemy

- ▶ One vector per word type
- ▶ But words have multiple senses
  - ▶ ... mouse<sup>1</sup> ... computer ...
  - ▶ ... animal ... mouse<sup>2</sup> ...
- ▶ Should mouse<sup>1</sup> and mouse<sup>2</sup> have the same word embedding?
  - ▶ Why not?
    - ▶ Syntagmatic association between mouse<sup>1</sup> and computer
    - ▶ Syntagmatic association between mouse<sup>2</sup> and animal
    - ▶ Paradigmatic association between computer and animal!
- ▶ How can we distinguish between mouse<sup>1</sup> and mouse<sup>2</sup>?
  - ▶ Context!

# Word Embeddings

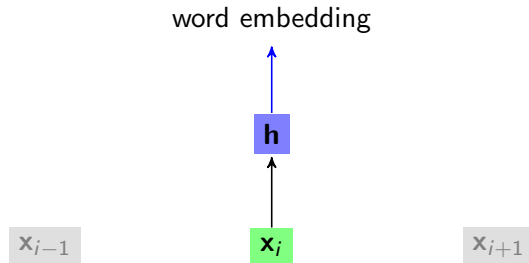


# Word Embeddings



- $h$  is an embedding of  $x_i$  only

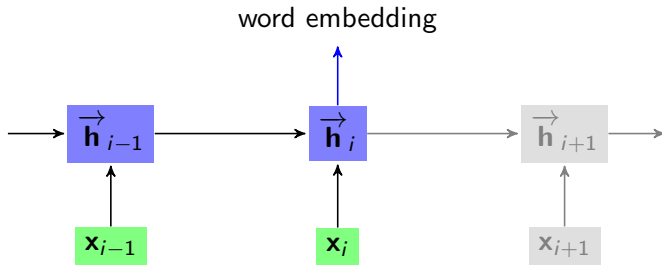
# Word Embeddings



- ▶  $h$  is an embedding of  $x_i$  only
  - ▶ How can we embed context information in  $h$ ?

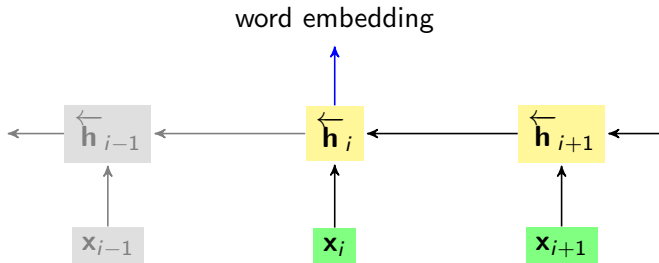


# Word Embeddings



- ▶  $\mathbf{h}$  is an embedding of  $\mathbf{x}_i$  only
  - ▶ How can we embed context information in  $\mathbf{h}$ ?

# Word Embeddings



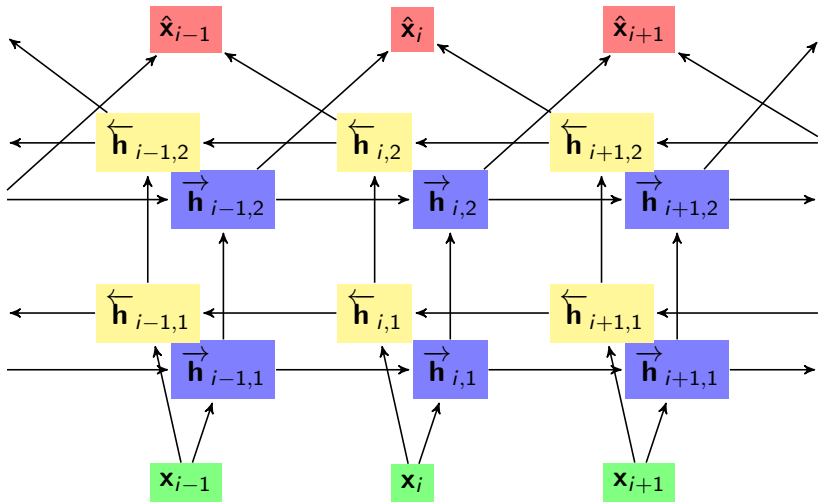
- ▶  $\mathbf{h}$  is an embedding of  $\mathbf{x}_i$  only
  - ▶ How can we embed context information in  $\mathbf{h}$ ?



► Embeddings from Language Models



- ▶ Embeddings from Language Models
- ▶ Based on a bidirectional recurrent neural network language model





- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)



- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 2 bidirectional LSTM layers

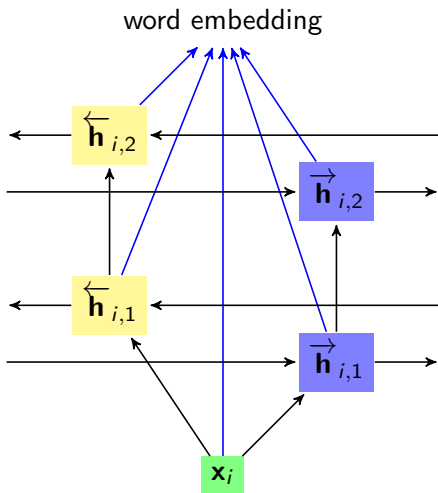


- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 2 bidirectional LSTM layers
- ▶ Output layer: softmax





- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 2 bidirectional LSTM layers
- ▶ Output layer: softmax
  
- ▶ Word embeddings: weighted sum of outputs of input and LSTM layers (task dependent)



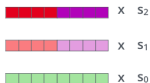


## Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

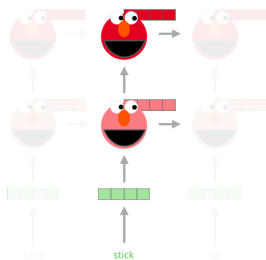


3- Sum the (now weighted) vectors

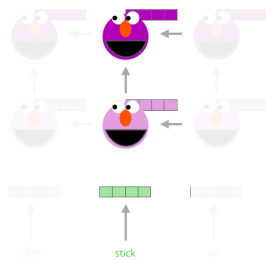


ELMo embedding of “stick” for this task in this context

Forward Language Model



Backward Language Model



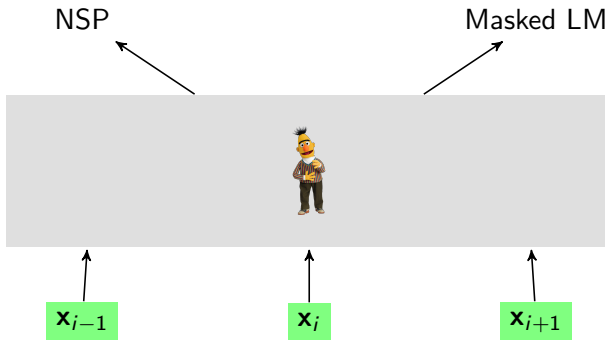
Source



► Bidirectional Encoder Representations from Transformers



► Bidirectional Encoder Representations from Transformers





- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)



- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers

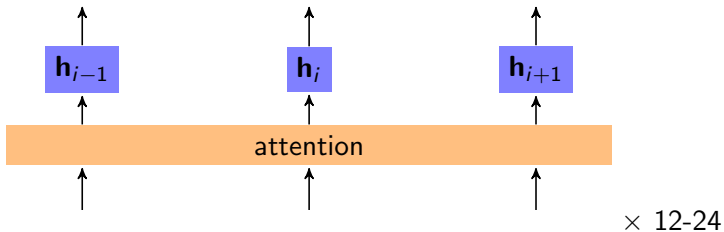


- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers
  - ▶ Encoder layer = (shared) attention layer + (individual) feedforward layers



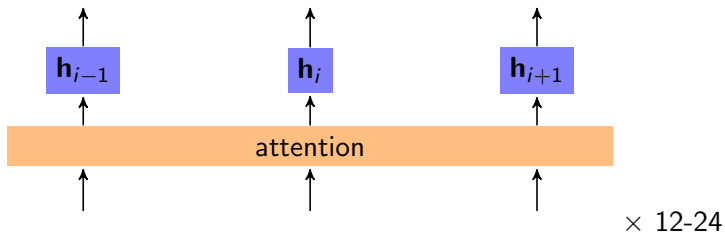


- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers
  - ▶ Encoder layer = (shared) attention layer + (individual) feedforward layers





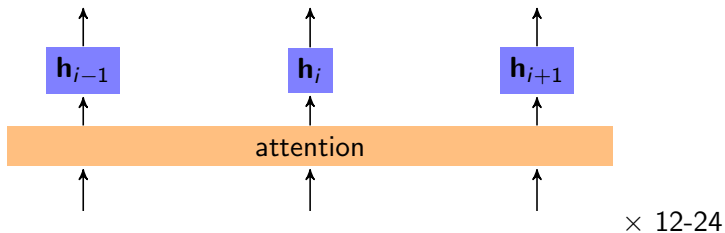
- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers
  - ▶ Encoder layer = (shared) attention layer + (individual) feedforward layers



- ▶ Output layer: 2 pre-training tasks



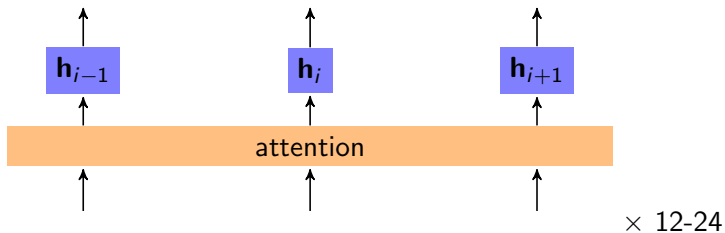
- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers
  - ▶ Encoder layer = (shared) attention layer + (individual) feedforward layers



- ▶ Output layer: 2 pre-training tasks
  - ▶ Masked LM (Cloze)



- ▶ Input layer: pre-trained word vectors (e.g. from word2vec)
- ▶ 12-24 encoder layers
  - ▶ Encoder layer = (shared) attention layer + (individual) feedforward layers



- ▶ Output layer: 2 pre-training tasks
  - ▶ Masked LM (Cloze)
  - ▶ NSP (Next Sentence Prediction)

# Masked LM

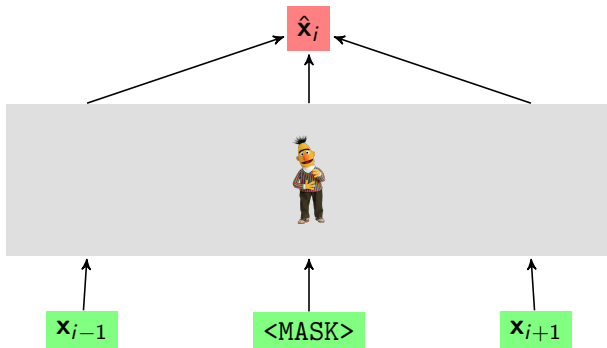
- ▶ Mask 15% of input tokens at random

# Masked LM

- ▶ Mask 15% of input tokens at random
- ▶ Predict masked words

# Masked LM

- ▶ Mask 15% of input tokens at random
- ▶ Predict masked words



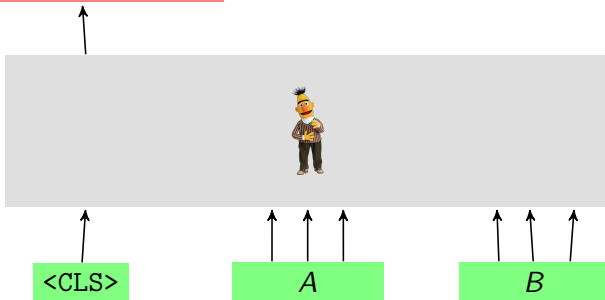
- ▶ Given sentences  $A$  and  $B$ , does  $B$  follow  $A$ ?



# NSP

- ▶ Given sentences  $A$  and  $B$ , does  $B$  follow  $A$ ?

$\hat{y} = \text{does } B \text{ follow } A?$






























- ▶ Word embeddings: combinations of outputs of encoder layers



## ► Word embeddings: combinations of outputs of encoder layers

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
12 	First Layer Embedding 	91.0
• • •	Last Hidden Layer 12 	94.9
7 	12 	
6 	+ • • •	
5 	+ 2 	95.5
4 	+ 1 	
3 	= 	
2 	Second-to-Last Hidden Layer 11 	95.6
1 	12 	
	+ 11 	
Help	+ 10 	95.9
	+ 9 	
	= 	
	Concat Last Four Hidden 9  10  11  12 	96.1