Contextualized Word Embeddings

CS114 Lab 12

Kenneth Lai

April 24, 2020

Contextualized Word Embeddings

CS114 Lab 12

Kenneth Lai

April 24, 2020

Source 1

Source 2

Contextualized Word Embeddings

CS114 Lab 12

Kenneth Lai

April 24, 2020

Source 1

Source 3

Word Embeddings

- Distributed representations of words

Word Embeddings

- Distributed representations of words
  - What is the difference between distributed and distributional representations?

Word Embeddings

- Sparse vectors

Word Embeddings

- Sparse vectors
  - One-hot, tf-idf, PPMI, etc.

Word Embeddings

- Sparse vectors
  - One-hot, tf-idf, PPMI, etc.
- Dense vectors

Word Embeddings

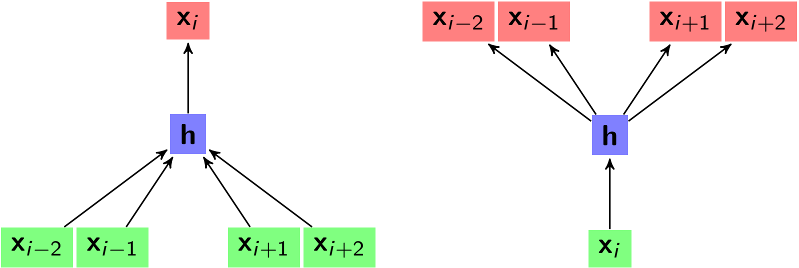
- Sparse vectors
  - One-hot, tf-idf, PPMI, etc.
- Dense vectors
  - SVD, word2vec, etc.

word2vec

- Based on a feedforward neural network language model

word2vec

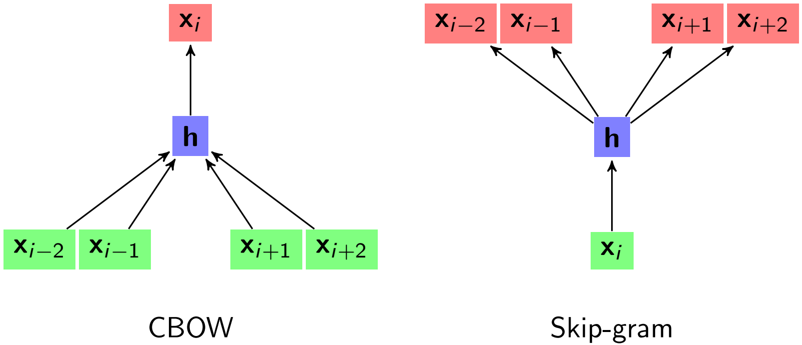
- Based on a feedforward neural network language model



CBOW　　　　Skip-gram

word2vec

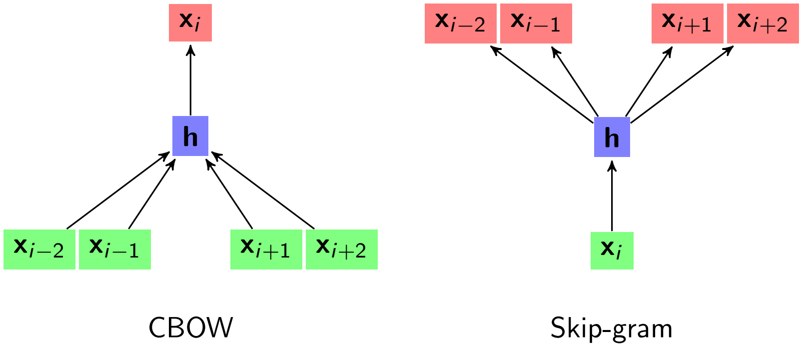
- Based on a feedforward neural network language model



- CBOW: use context to predict current word

word2vec

- Based on a feedforward neural network language model



- 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

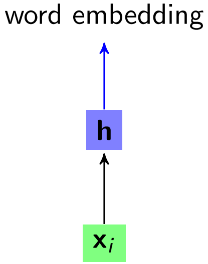
- Input layer: one-hot word vectors
- Hidden (projection) layer: identity (linear!) activation function, no bias
  - Input $\rightarrow$ hidden $=$ table lookup (in weight matrix)
- Output layer: softmax activation function

- 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)



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[1] controlling a computer system in 1968

Polysemy

- One vector per word type
- But words have multiple senses
  - a mouse^1 controlling a computer system in 1968
  - a quiet animal like a mouse^2

Polysemy

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

Polysemy

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

Polysemy

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

Polysemy

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

Polysemy

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

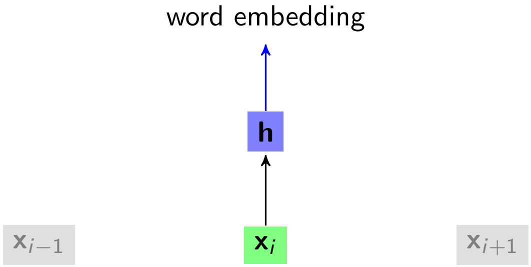
Polysemy

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

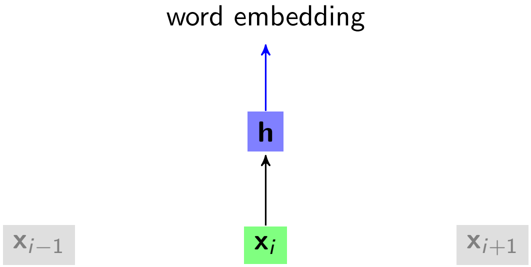
Polysemy

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

Word Embeddings

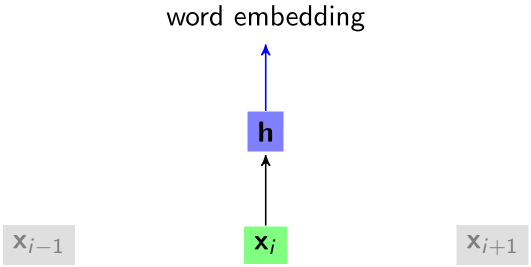


Word Embeddings



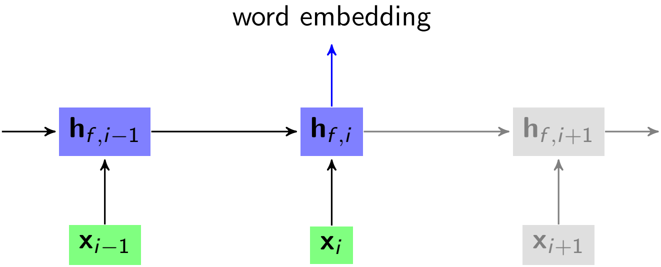
- **h** is an embedding of $\mathbf{x}_i$ only

Word Embeddings



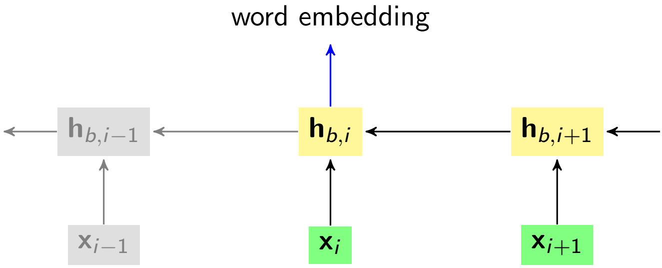
- $\mathbf{h}$ is an embedding of $\mathbf{x}_i$ only
  - How can we embed context information in $\mathbf{h}$?

Word Embeddings



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

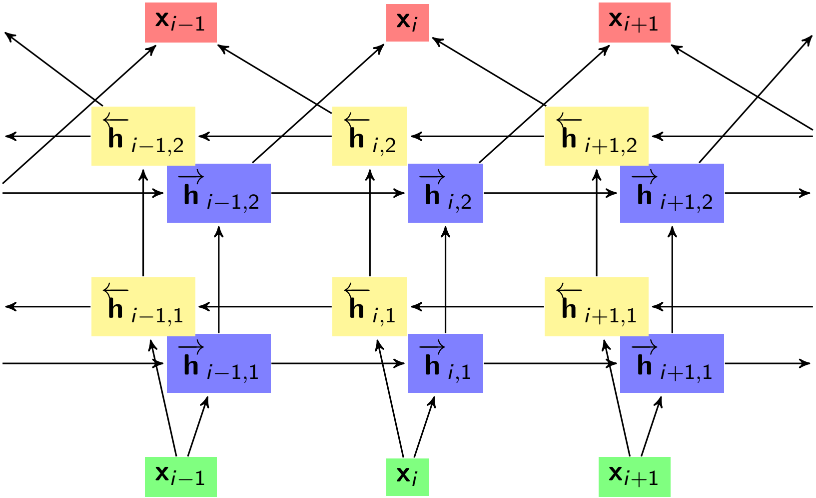
Word Embeddings



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

- Embeddings from Language Models

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


$\mathbf{x}_{i-1}$
$\mathbf{x}_{i}$
$\mathbf{x}_{i+1}$
$\overleftarrow{\mathbf{h}}_{i-1,2}$
$\overleftarrow{\mathbf{h}}_{i,2}$
$\overleftarrow{\mathbf{h}}_{i+1,2}$
$\overrightarrow{\mathbf{h}}_{i-1,2}$
$\overrightarrow{\mathbf{h}}_{i,2}$
$\overrightarrow{\mathbf{h}}_{i+1,2}$
$\overleftarrow{\mathbf{h}}_{i-1,1}$
$\overleftarrow{\mathbf{h}}_{i,1}$
$\overleftarrow{\mathbf{h}}_{i+1,1}$
$\overrightarrow{\mathbf{h}}_{i-1,1}$
$\overrightarrow{\mathbf{h}}_{i,1}$
$\overrightarrow{\mathbf{h}}_{i+1,1}$
$\mathbf{x}_{i-1}$
$\mathbf{x}_{i}$
$\mathbf{x}_{i+1}$

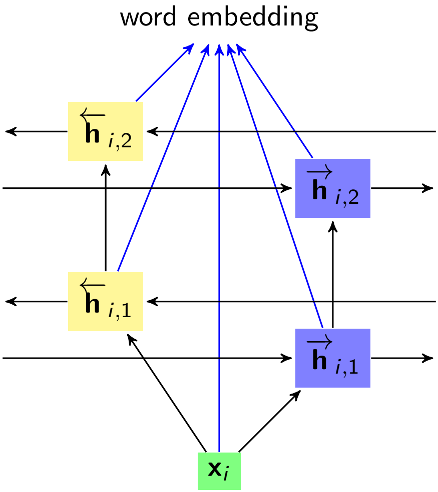
- Input layer: pre-trained word vectors (e.g. from word2vec)

- 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)


word embedding
$\overleftarrow{\mathbf{h}}_{i,2}$
$\overrightarrow{\mathbf{h}}_{i,2}$
$\overleftarrow{\mathbf{h}}_{i,1}$
$\overrightarrow{\mathbf{h}}_{i,1}$
$\mathbf{x}_i$

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

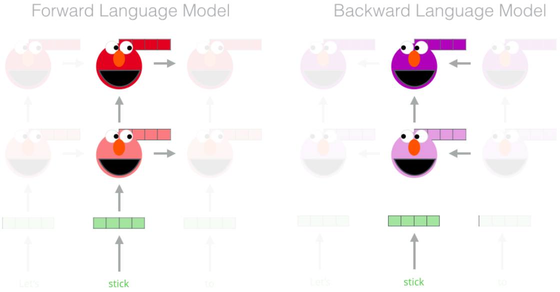
x s_2

x s_1

x s_0

3- Sum the (now weighted) vectors

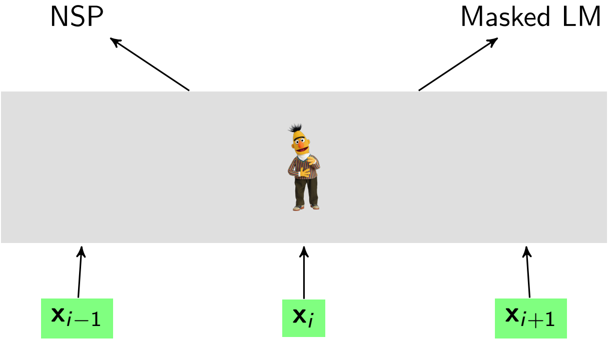
ELMo embedding of “stick” for this task in this context



Source

- Bidirectional Encoder Representations from Transformers

- Bidirectional Encoder Representations from Transformers



Transformers

- An encoder-decoder architecture with attention

Transformers

- An encoder-decoder architecture with attention
- To learn more about encoder-decoder architecture, take StatNLP in the fall!

Transformers

- An encoder-decoder architecture with attention
- To learn more about encoder-decoder architecture, take StatNLP in the fall!

- Encoders take a sequence as input, output sequence representation

Transformers

- An encoder-decoder architecture with attention
- To learn more about encoder-decoder architecture, take StatNLP in the fall!

- Encoders take a sequence as input, output sequence representation
- Decoders take a sequence representation as input, output sequence

Transformers

- An encoder-decoder architecture with attention
- To learn more about encoder-decoder architecture, take StatNLP in the fall!

- Encoders take a sequence as input, output sequence representation
- Decoders take a sequence representation as input, output sequence

- Useful for machine translation, among other things

Transformers

- An encoder-decoder architecture with attention
- To learn more about encoder-decoder architecture, take StatNLP in the fall!

- Encoders take a sequence as input, output sequence representation
- Decoders take a sequence representation as input, output sequence

- Useful for machine translation, among other things
- BERT only uses the encoder part

Attention

- To learn more about attention, take StatNLP in the fall!

Attention

- To learn more about attention, take StatNLP in the fall!
- For each input i, compute a query, key, and value

Attention

- To learn more about attention, take StatNLP in the fall!
- For each input i, compute a query, key, and value
- The output for each input i is a weighted sum of the values for all inputs j, by how similar the key for j is to the query for i

Attention

- To learn more about attention, take StatNLP in the fall!

- For each input i, compute a query, key, and value
- The output for each input i is a weighted sum of the values for all inputs j, by how similar the key for j is to the query for i

- Result: the output for each input i contains information about the whole sequence

Attention

- To learn more about attention, take StatNLP in the fall!

- For each input i, compute a query, key, and value
- The output for each input i is a weighted sum of the values for all inputs j, by how similar the key for j is to the query for i

- Result: the output for each input i contains information about the whole sequence
  - More information about more relevant parts of the sequence

Attention

- To learn more about attention, take StatNLP in the fall!

- For each input i, compute a query, key, and value
- The output for each input i is a weighted sum of the values for all inputs j, by how similar the key for j is to the query for i

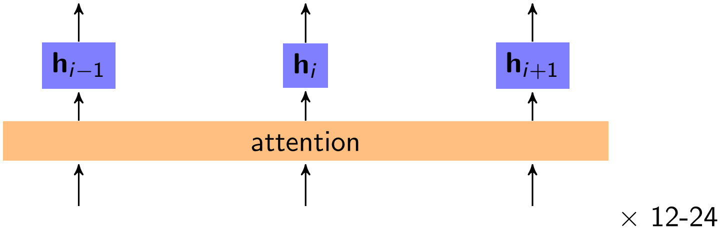
- Result: the output for each input i contains information about the whole sequence
  - More information about more relevant parts of the sequence
- Useful for long-distance dependencies, among other things

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

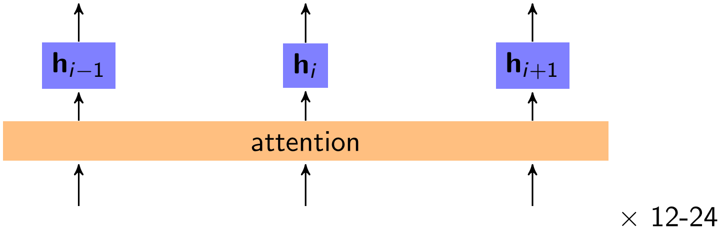
- Input layer: pre-trained word vectors (e.g. from word2vec)
- 12-24 encoder layers

- 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

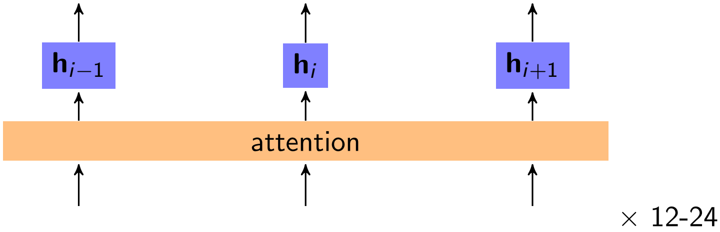


- Input layer: pre-trained word vectors (e.g. from word2vec)
- 12-24 encoder layers
  - Encoder layer = (shared) attention layer + (individual) feedforward layers



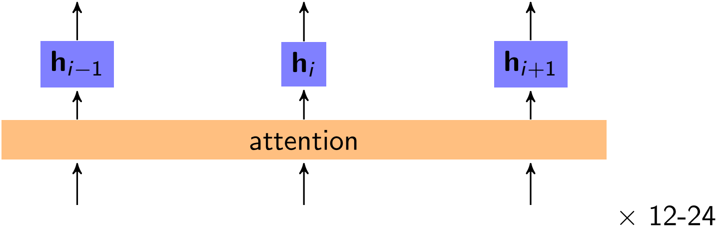
- Output layer: 2 pre-training tasks

- 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

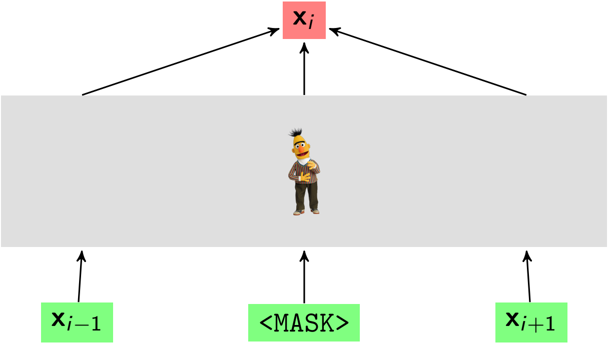
- Mask 15% of input tokens at random

Masked LM

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

Masked LM

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

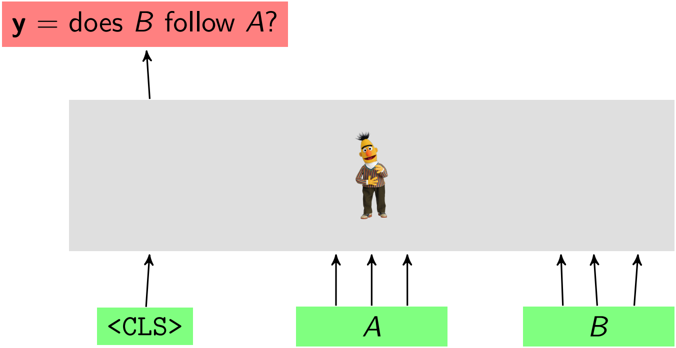


NSP

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

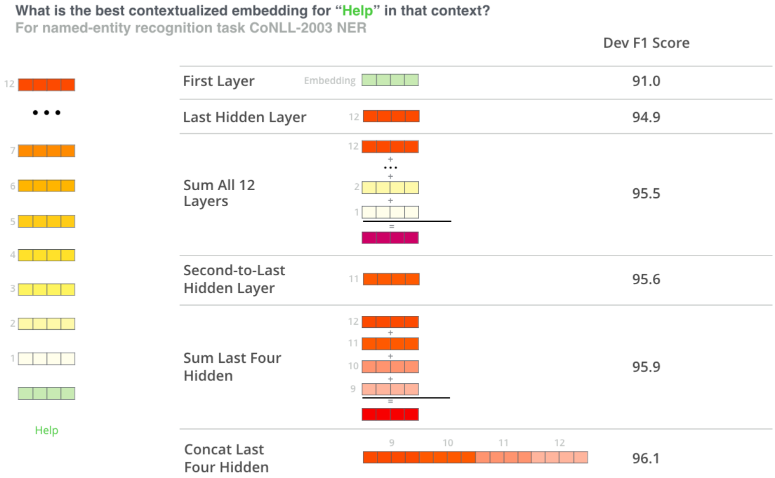
NSP

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



- Word embeddings: combinations of outputs of encoder layers

- Word embeddings: combinations of outputs of encoder layers



Source