Contextualized Word Embeddings

Kenneth Lai

Brandeis University

November 16, 2022

Contextualized Word Embeddings

Kenneth Lai

Brandeis University

November 16, 2022

Source 1

Source 2

Contextualized Word Embeddings

Kenneth Lai

Brandeis University

November 16, 2022

Source 1

Source 3

Announcements

- By 11:59pm today
  - HW4 due
- Paper presentations begin Monday
- Feedback on final project ideas also hopefully by Monday
- No (additional) HW5
  - Instead, your final project progress report will serve as HW5
    - Due 12/7

Paper Presentation Guidelines

- Groups should aim for around 20 minutes for summary and analysis, and around 5 minutes for questions and discussion
- Presentations should cover the following themes
  - Describe the problem, and why it is interesting/important
    - What dimensions of meaning are the authors interested in (e.g., expression meaning vs. speaker meaning, meaning as truth vs. meaning as use, etc.)?
  - How do the authors try to solve the problem?
    - Methods, data, evaluation, etc.
  - What are the results and conclusions?
    - For someone interested in (a) different dimension(s) of meaning than the authors, what lessons can they learn from the paper?

Paper Presentation Guidelines

- Everyone: please attend class in person if you can!
- Presenters: please upload your slides or other materials to LATTE in advance

Today's Plan

- Contextualized Word Embeddings
  - Issues with word2vec
  - 
  - 
  - If time: Issues with large language models

word2vec

- Two issues with word2vec:
  - One vector per word type
  - Limited (fixed-length) context
    - e.g., ± 2 words, etc.

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?

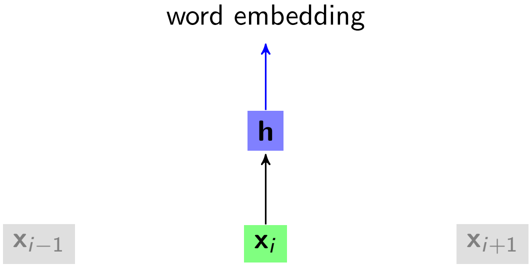
Polysemy

- One vector per word type
- But words have multiple senses
  - ... mouse^1 ... computer ...
  - ... animal ... mouse^2 ...
- Should mouse^1 and mouse^2 have the same word embedding?
  - Embeddings of computer and animal wind up closer than they "should" be

Polysemy

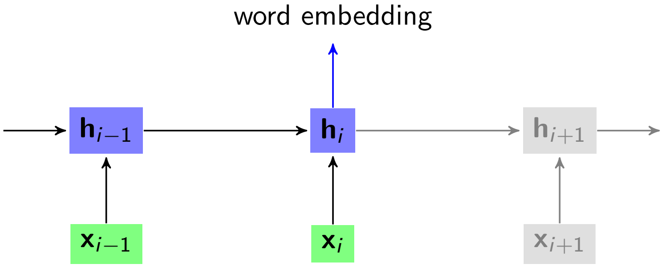
- One vector per word type
- But words have multiple senses
  - ... mouse^1 ... computer ...
  - ... animal ... mouse^2 ...
- Should mouse^1 and mouse^2 have the same word embedding?
  - Embeddings of computer and animal wind up closer than they "should" be
- How can we distinguish between mouse^1 and mouse^2?
  - Context!

Word Embeddings



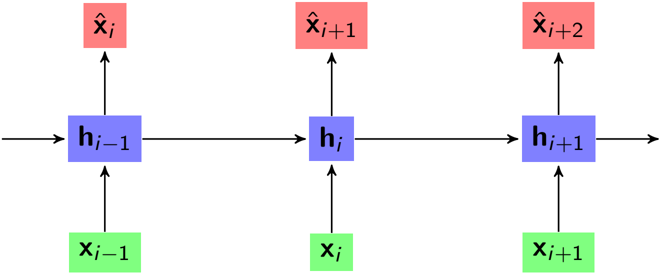
- $\mathbf{h}$ is an embedding of $\mathbf{x}_i$ only
  - How can we embed context information in $\mathbf{h}$?

Word Embeddings



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

Recurrent Neural Networks



- Neural networks in which the output of a layer in one time step is input to a layer in the next time step
  - Here, time step = word

Recurrent Neural Networks

- RNNs allow for contextualized word embeddings
  - Multiple word senses
  - Arbitrary-length context
- Is this enough?

Context and Long-Distance Dependencies

- $\mathbf{h}_i$ encodes the context $\mathbf{x}_1, ..., \mathbf{x}_i$
  - But mostly $\mathbf{x}_i$, less $\mathbf{x}_{i-1}$, even less $\mathbf{x}_{i-2}$, ..., very little $\mathbf{x}_1$
- Context is local

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline (was/were) cancelling (was/were) full.

Context and Long-Distance Dependencies

- Example: subject-verb agreement
- The flights the airline was cancelling (was/were) full.
  - The context for “was” is mostly “airline”

Context and Long-Distance Dependencies

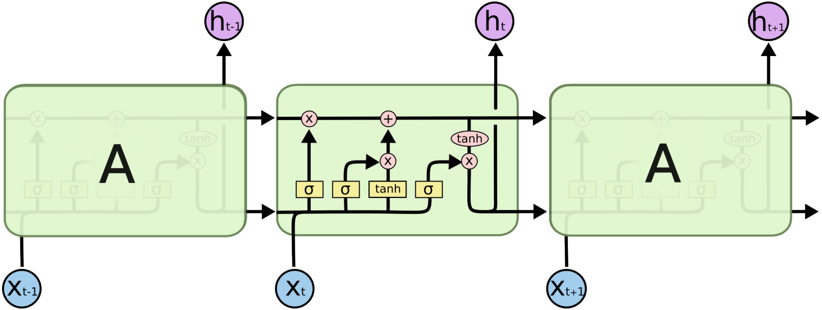
- Example: subject-verb agreement
- The flights the airline was cancelling were full.
  - The context for "was" is mostly "airline"
  - The context for "were" is mostly "cancelling", "was", "airline"
    - Very little "flights"

Context and Long-Distance Dependencies

- Two approaches to handling long-distance dependencies:
  - Memory-based (e.g. long short-term)
    -  does this
  - Attention-based
    - At each time step, the model explicitly computes which other words to pay attention to
    -  does this

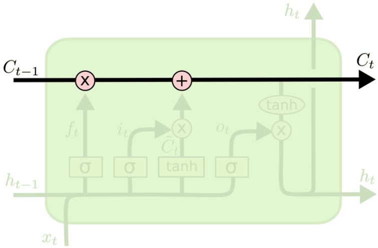
- Embeddings from Language Models
- Based on a bidirectional long short-term memory (LSTM) language model

Long Short-Term Memory



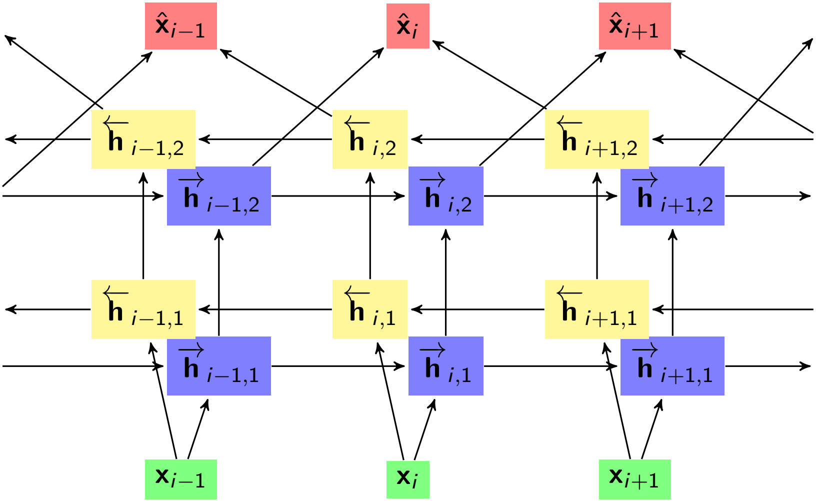
Source

Long Short-Term Memory



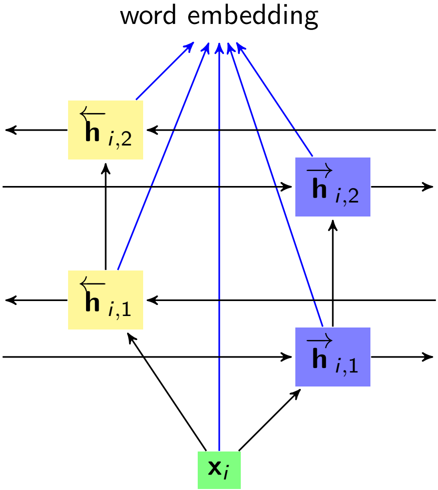
Source

- Separate memory (cell) state
  - Reading from and writing to memory controlled by gates
    - Each gate contains one or two neural network layers
  - State persists across time
    - May remember information from long ago
- See Christopher Olah's Understanding LSTM Networks for more details!

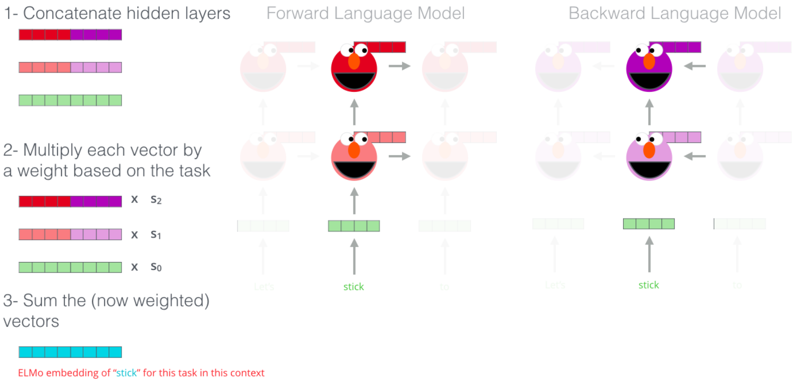

$\hat{\mathbf{x}}_{i-1}$
$\hat{\mathbf{x}}_{i}$
$\hat{\mathbf{x}}_{i+1}$
$\overleftarrow{\mathbf{h}}_{i-1,2}$
$\overleftarrow{\mathbf{h}}_{i,2}$
$\overleftarrow{\mathbf{h}}_{i+1,2}$
$\overrightarrow{\mathbf{h}}_{i-1,2}$
$\overrightarrow{\mathbf{h}}_{i,2}$
$\overrightarrow{\mathbf{h}}_{i+1,2}$
$\overleftarrow{\mathbf{h}}_{i-1,1}$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

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

- Word embeddings: weighted sum of outputs of input and LSTM layers (task dependent)


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

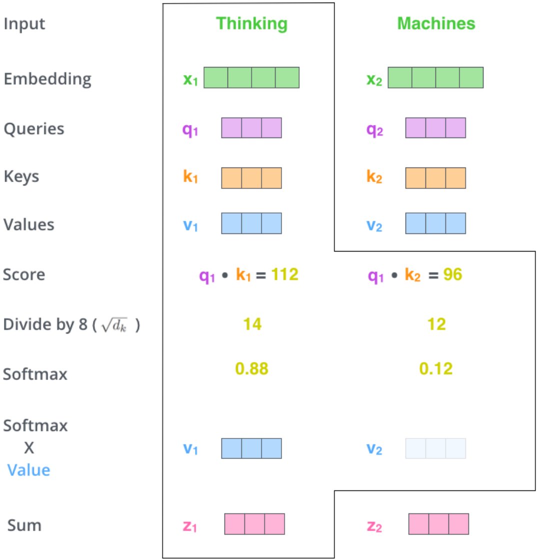
Embedding of "stick" in "Let's stick to" - Step #2



Source

- Bidirectional Encoder Representations from Transformers
- Based on a transformer ("attention is all you need") model
  - See Jay Alammar's The Illustrated Transformer for more details!

Attention

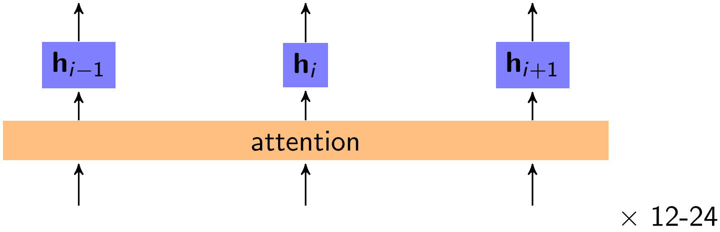


Source

Transformers

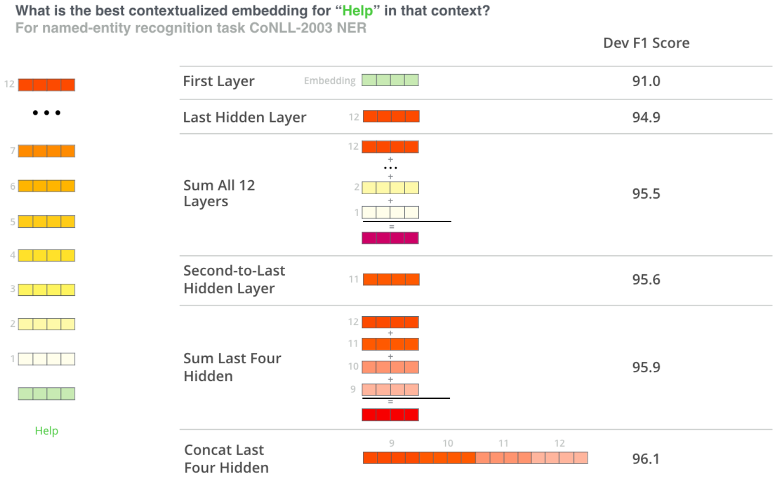
- "Attention Is All You Need" (Vaswani et al., 2017)
- No recurrence, relies entirely on attention (and feedforward layers) to capture global dependencies
  - Recurrent neural networks are inherently sequential, processing one word at a time
  - Transformers are more parallel, looking at the entire sequence at once
    - More efficient, especially on GPUs
    - Also scores better on many NLP 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)
    - Mask 15% of input tokens at random, predict masked words
  - NSP (Next Sentence Prediction)
    - Given sentences A and B, does B follow A?

- Word embeddings: combinations of outputs of encoder layers



Source

Issues with Large Language Models

- Models are expensive (environmentally and financially)! (Strubell et al., 2019)
  - Training  on GPUs in the cloud costs $3,751-$12,571 and generates 1,438 pounds of CO_2 emissions

Issues with Large Language Models

- Recommendations from Strubell et al. (2019)
  - "Authors should report training time and sensitivity to hyperparameters."
  - "Academic researchers need equitable access to computation resources."
  - "Researchers should prioritize computationally efficient hardware and algorithms."

Issues with Large Language Models

- Risks of large language models include (but are not limited to) (Bender et al., 2021):
  - Models can reproduce/amplify biases (or even abusive language) found in their training data
  - Bad actors can use generated text for nefarious purposes
  - One can extract personally identifiable information from large language models! (Carlini et al., 2021)

Issues with Large Language Models

- Recommendations from Bender et al. (2021):
  - Curate your data for your specific task, rather than ingesting all the data you can find on the internet
  - Document your data sources, goals, values, motivations, and potential users/ stakeholders
  - Pre-mortems: before development, identify possible failures and ways to avoid them
  - Value sensitive design: identify stakeholders, work with them, and make sure your system supports their values