Vector Embeddings

CSE 447 / 517
February 3rd, 2022 (Week 5)

Logistics

- A3 is due **tomorrow (Friday, 2/4)**

Agenda

- Quiz 4 Solutions
- Vector embeddings
  - “Static” word embeddings
  - Contextualized word embeddings
- Q & A

Quiz 4 - Question 1 Setup

Consider three sample documents, x_1 x_2 x_3 that are similar to the ones in the lecture.

x_1: great , we love NLP
x_2: say yes to NLP quizzes
x_3: great , no quizzes , we say

Tokens are separated by whitespace.

Compute the count matrix (see Lecture slide 8).

Quiz 4 - Question 1

x_1: great , we love NLP
x_2: say yes to NLP quizzes
x_3: great , no quizzes , we say

	Vector for x_1	Vector for x_2	Vector for x_3
great	1	0	1
we	1	0	1
love	1	0	0
NLP	1	1	0
say	0	1	1
yes	0	1	0
to	0	1	0
quizzes	0	1	1
no	0	0	1
,	1	0	2

Quiz 4 - Question 2 Setup

Consider three sample documents, x_1 x_2 x_3 that are similar to the ones in the lecture.

x_1: great , we love NLP
x_2: say yes to NLP quizzes
x_3: great , no quizzes , we say

Tokens are separated by whitespace.

Compute the positive pointwise mutual information (see Lecture Slide 13) $[A]_{v,c}$ (word v for c-th document). Round to 2 decimal places.

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

$$[\mathbf{A}]_{v,c} = \left[\log \frac{\text{count}_{\boldsymbol{x}_c}(v)}{\frac{\text{count}_{\boldsymbol{x}_{1:C}}(v)}{N} \cdot \ell_c}\right]_+$$

N: the total number of tokens

l_c: the length of document c

c: the index of the document

$[x]_+$: max(0, x)

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

The count of token v in document c.

$$[\mathbf{A}]_{v,c} = \left[\log \frac{\boxed{\text{count}_{\boldsymbol{x}_c}(v)}}{\frac{\text{count}_{\boldsymbol{x}_{1:C}}(v)}{N} \cdot \ell_c} \right]_+$$

N: the total number of tokens
l$_c$: the length of document c
c: the index of the document
[x]$_+$: max(0, x)

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

The count of token v in document c.

$$[\mathbf{A}]_{v,c} = \left[\log \frac{\boxed{\text{count}_{\boldsymbol{x}_c}(v)}}{\boxed{\frac{\text{count}_{\boldsymbol{x}_{1:C}}(v)}{N}} \cdot \ell_c}\right]_+$$

How likely is token v to appear in the corpus assuming tokens are independent (unigram).

N: the total number of tokens

l$_c$: the length of document c

c: the index of the document

[x]$_+$: max(0, x)

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

The count of token v in document c.

$$[\mathbf{A}]_{v,c} = \left[\log \frac{\boxed{\mathrm{count}_{\boldsymbol{x}_c}(v)}}{\boxed{\frac{\mathrm{count}_{\boldsymbol{x}_{1:C}}(v)}{N} \cdot \ell_c}}\right]_+$$

How many token v should we expect to see in document c?

N: the total number of tokens
l_c: the length of document c
c: the index of the document
$[x]_+$: max(0, x)

Review: Positive PMI

Pointwise mutual information: a measurement of association (in this case, token and documents).

$$[\mathbf{A}]_{v,c} = \left[\log \frac{\text{count}_{\boldsymbol{x}_c}(v)}{\frac{\text{count}_{\boldsymbol{x}_{1:C}}(v)}{N} \cdot \ell_c}\right]_+$$

$$= \left[\log \frac{N \cdot \text{count}_{\boldsymbol{x}_c}(v)}{\text{count}_{\boldsymbol{x}_{1:C}}(v) \cdot \ell_c}\right]_+$$

N: the total number of tokens
l_c: the length of document c
c: the index of the document
$[x]_+$: max(0, x)

Quiz 4 - Question 2

$$[\mathbf{A}]_{v,c} = \left[\log \frac{N \cdot \text{count}_{\boldsymbol{x}_c}(v)}{\text{count}_{\boldsymbol{x}_{1:C}}(v) \cdot \ell_c}\right]_+$$

$$[A]_{\text{"}NLP\text{"},\,2} = \left[\log \frac{17 \times 1}{2 \times 5}\right]_+ \approx [0.53]_+ = 0.53$$

N: the total number of tokens
l_c: the length of document c
c: the index of the document
$[x]_+$: max(0, x)

	Vector for x_1	Vector for x_2	Vector for x_3
great	1	0	1
we	1	0	1
love	1	0	0
NLP	1	1	0
say	0	1	1
yes	0	1	0
to	0	1	0
quizzes	0	1	1
no	0	0	1
,	1	0	2

Quiz 4 - Question 2

$$[\mathbf{A}]_{v,c} = \left[\log \frac{N \cdot \text{count}_{\boldsymbol{x}_c}(v)}{\text{count}_{\boldsymbol{x}_{1:C}}(v) \cdot \ell_c}\right]_+$$

$$\begin{aligned}[A]_{\text{"}we\text{"},\,2} &= \left[\log \frac{17 \times 0}{2 \times 5}\right]_+ \\ &= [-\infty]_+ \\ &= 0\end{aligned}$$

N: the total number of tokens
l_c: the length of document c
c: the index of the document
$[x]_+$: max(0, x)

	Vector for x_1	**Vector for x_2**	**Vector for x_3**
great	1	0	1
we	1	0	1
love	1	0	0
NLP	1	1	0
say	0	1	1
yes	0	1	0
to	0	1	0
quizzes	0	1	1
no	0	0	1
,	1	0	2

Quiz 4 - Question 2

$$[\mathbf{A}]_{v,c} = \left[\log \frac{N \cdot \text{count}_{\boldsymbol{x}_c}(v)}{\text{count}_{\boldsymbol{x}_{1:C}}(v) \cdot \ell_c}\right]_+$$

$$[A]_{",",\,3} = \left[\log \frac{17 \times 2}{3 \times 7}\right]_+$$
$$\approx [0.48]_+$$
$$= 0.48$$

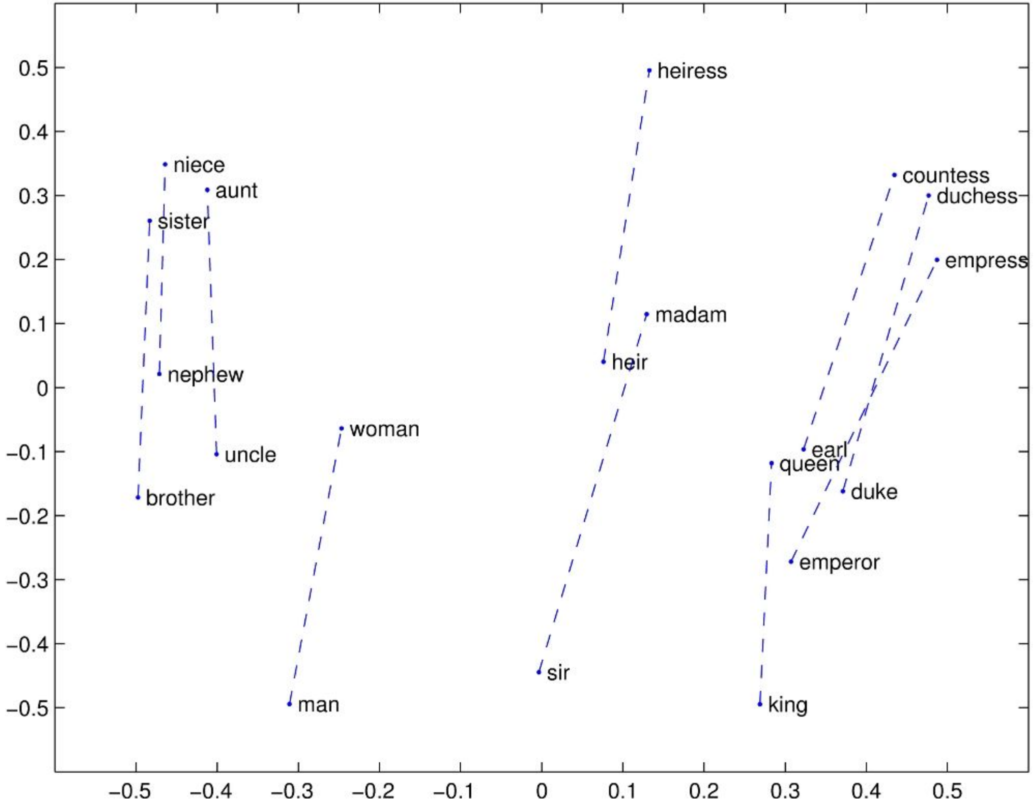
N: the total number of tokens
l_c: the length of document c
c: the index of the document
$[x]_+$: max(0, x)

	Vector for x_1	Vector for x_2	Vector for x_3
great	1	0	1
we	1	0	1
love	1	0	0
NLP	1	1	0
say	0	1	1
yes	0	1	0
to	0	1	0
quizzes	0	1	1
no	0	0	1
,	1	0	2

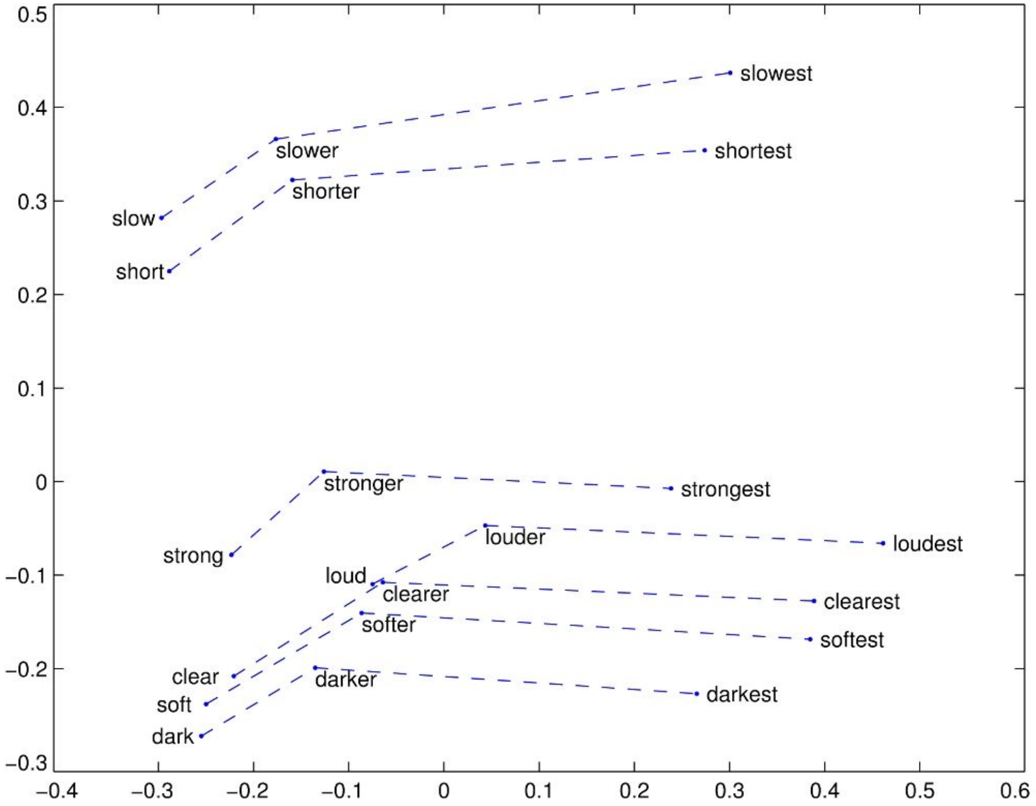
Word Embeddings: A Quick Review

- Motivation:
  - Represent words in a computationally efficient and semantically meaningful way
- Evaluation:
  - Intrinsic: word similarities, TOEFL-like synonyms, analogies, etc.
  - Extrinsic: do the embeddings improve system performance?
- Using embeddings in your model:
  - *Freeze* embeddings and use as-is in your model
  - *Fine-tune* embeddings, updating them as you train

Man-woman relations in embeddings



Comparative-superlative relations in embeddings



Distributional Hypothesis, again

- **A word's meaning is given by words that appear frequently close by**
- When a word *w* appears in text, its context is the set of words that appear nearby (in some window).
- Dense Vectors From 10,000 feet:
  - Find a bunch of times that *w* occurs in text.
  - Use the many contexts of *w* to build a vector.

…government debt problems turning into ***banking*** *crises as happened in 2009…*
…saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge…*
…India has just given its ***banking*** *system a shot in the arm…*

These context words define *banking*.

Dense Word Vectors

- Let’s assign each word a dense word vector
- But each word’s vector should be similar to vectors of words that appear in similar contexts.
- Example:

$$\textit{U.S.} = \begin{pmatrix} 0.281 \\ 0.129 \\ 0.312 \\ -1.29 \\ -0.21 \end{pmatrix} \qquad \textit{Washington} = \begin{pmatrix} 0.271 \\ 0.110 \\ 0.311 \\ -1.33 \\ -0.11 \end{pmatrix} \qquad \textit{grass} = \begin{pmatrix} -0.121 \\ 0.930 \\ 0.121 \\ 1.53 \\ -0.51 \end{pmatrix}$$

If words appear in similar contexts, they have similar vectors!

“U.S.” and “Washington” occur in similar contexts!

WORLD NEWS FEBRUARY 2, 2018 / 1:09 AM / 11 DAYS AGO

Exclusive: U.S. to impose arms embargo on South Sudan to end conflict - sources

Washington imposes weapons embargo on South Sudan

US maintains pressure on central and regional governments to end conflict

"Static" Word Embeddings

Each word maps to a single vector, based on their occurrence with other words in a large corpus.

Connects to LSA/I, parallels to LMs

Examples of popular pretrained word embeddings:

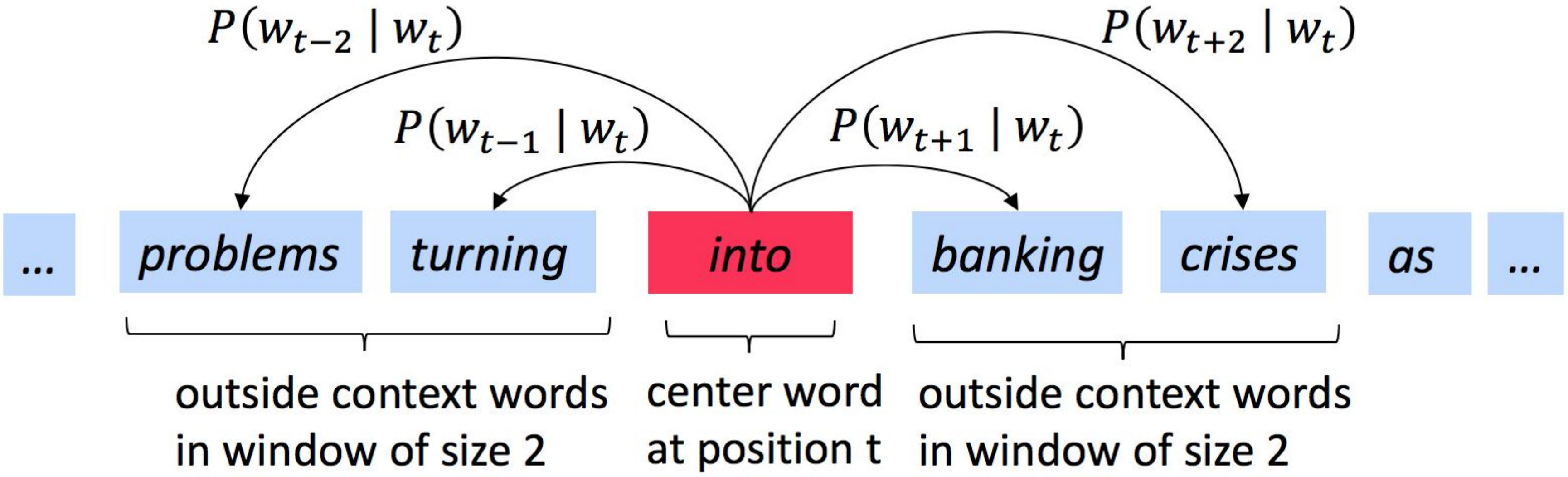
- **word2vec**: Trained on Google News
- **GloVe**: Trained on Wikipedia, Gigaword, Common Crawl, or Twitter
- **FastText**: Trained on Wikipedia or Common Crawl

Word2Vec: Overview

- Word2Vec is a framework for learning word vectors. Basic Idea:
- We have a large corpus of text.
- Every word in a fixed vocabulary is assigned a vector.
- Go through each position *t* in the text, which has a center word *c* and outside (context) words *o.*
- **Use the similarity of the word vectors for *c* and *o* to calculate the probability of *o* given *c.***
- Training: Continuously adjust the word vectors to maximize this probability.

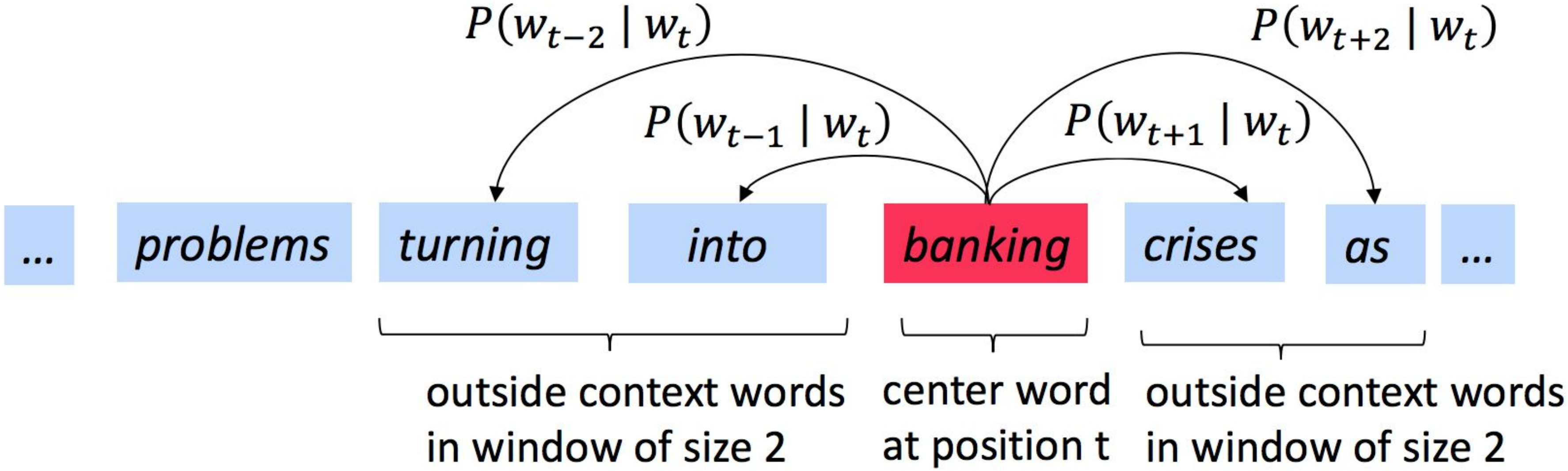
Word2Vec: Overview

- Example for computing $P(w_{t+j} \mid w_t)$



Word2Vec: Overview

- Example for computing $P(w_{t+j} \mid w_t)$



Word2Vec: Loss Function

- For each position $t = 1 \ldots T$, predict context words within a fixed-size window of size m, given the center word w_t
- Likelihood (θ = parameters of the model, or things we want to optimize):

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

For each position in the text.

For each word within the window

Probability of word in window given center word.

Word2Vec: Loss Function

- Loss function *J*: Averaged negative log-likelihood
  - Work in logspace!
  - Negative to turn the problem from a maximization problem into a minimization problem
- If we minimize the loss function *J*, then we maximize the predictive accuracy!

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Word2Vec: Loss Function

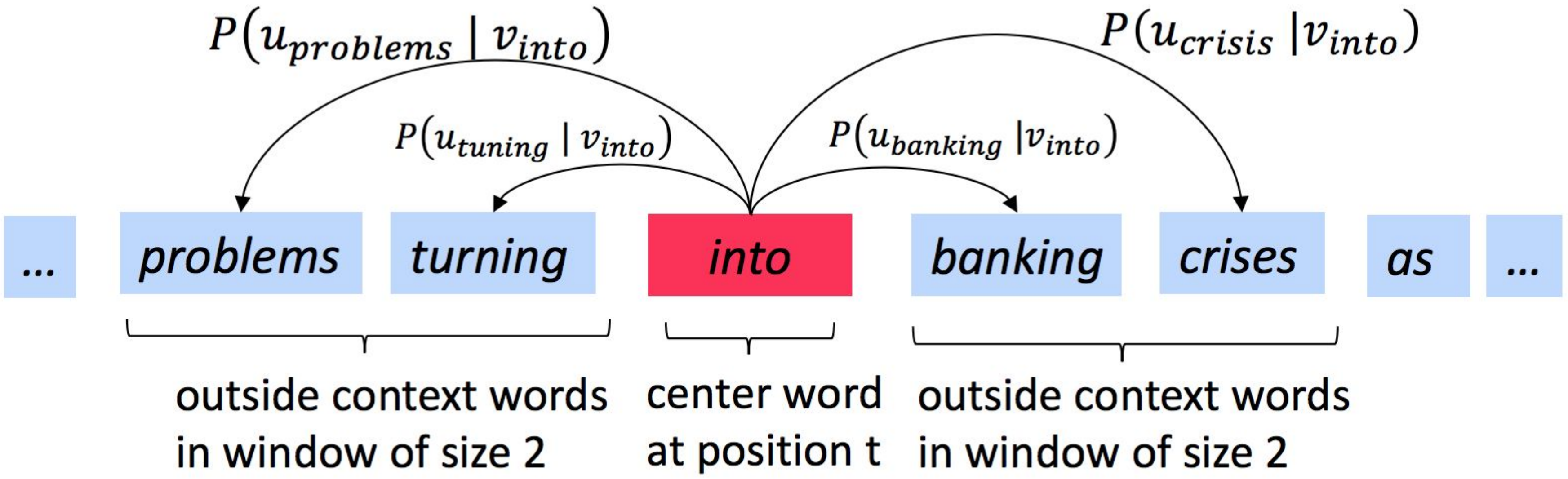
- Question: How do we calculate $P(w_{t+j} \mid w_t)$?
- Answer: Use two vectors per word *w*.
    - Use the vector v_w when w is the center word.
    - Use the vector u_w when w is the context word.
- Thus, for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

- Look familiar?

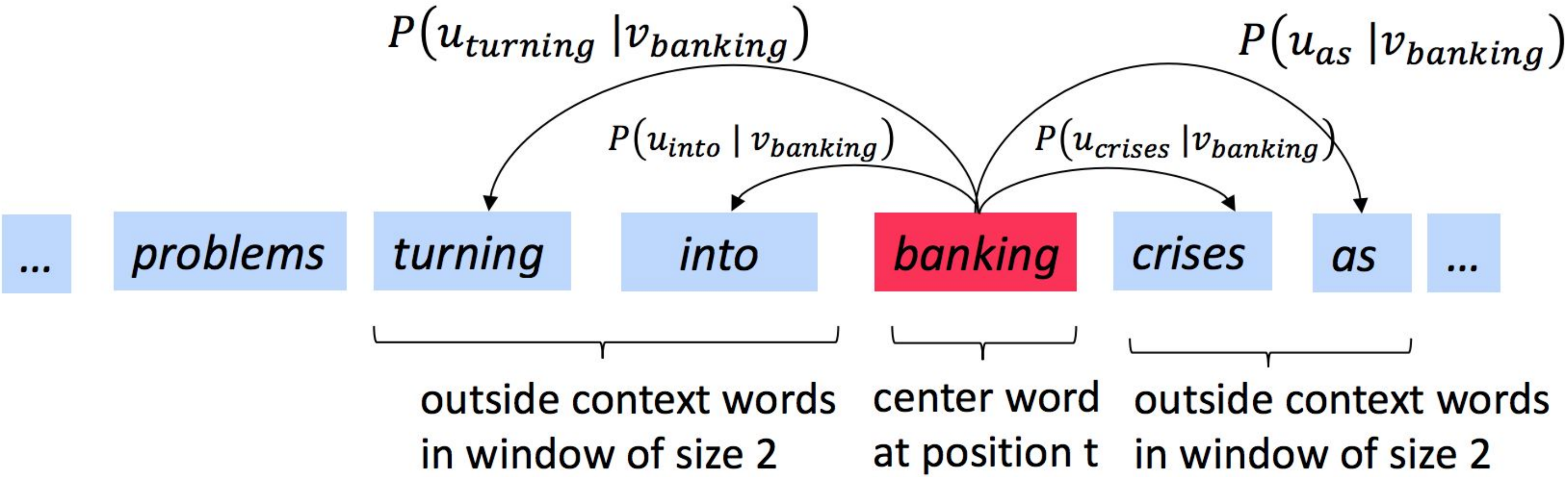
Word2Vec: Now with Vectors!

- Example for computing $P(w_{t+j} \mid w_t)$



Word2Vec: Now with Vectors!

- Example for computing $P(w_{t+j} \mid w_t)$



Word2Vec: Why this prediction function?

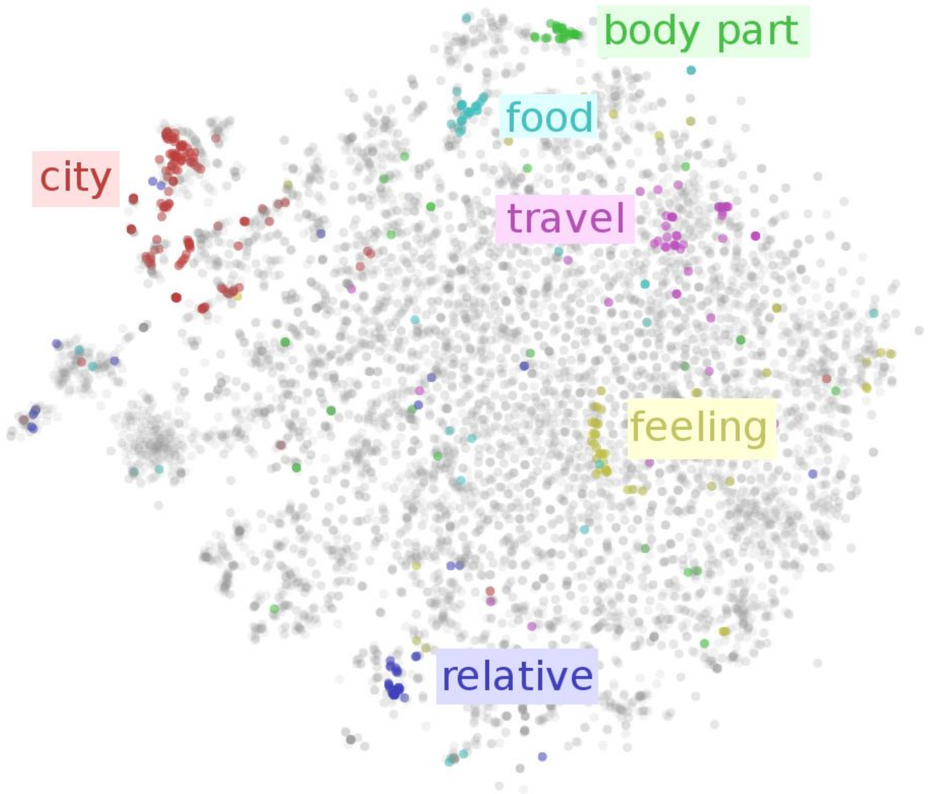
$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Dot product compares similarity of *o* and *c*.
Larger dot product = larger probability

After taking exponent,
normalize over entire vocabulary

- Softmax shows up again.
- We can train this with gradient descent.
- This model puts words that frequently co-occur nearby in vector space (to maximize the dot product).

Clusters of dense word vectors



Why separate center and context vectors?

- Why use two vectors (one for when the word is the context, one for when the word is the center)?
  - Makes optimization/training easier in practice.
  - Our final word vector is traditionally average of the context and center vector for a word.

Why separate center and context vectors?

- Another angle:

word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method

Yoav Goldberg and Omer Levy
{yoav.goldberg,omerlevy}@gmail.com

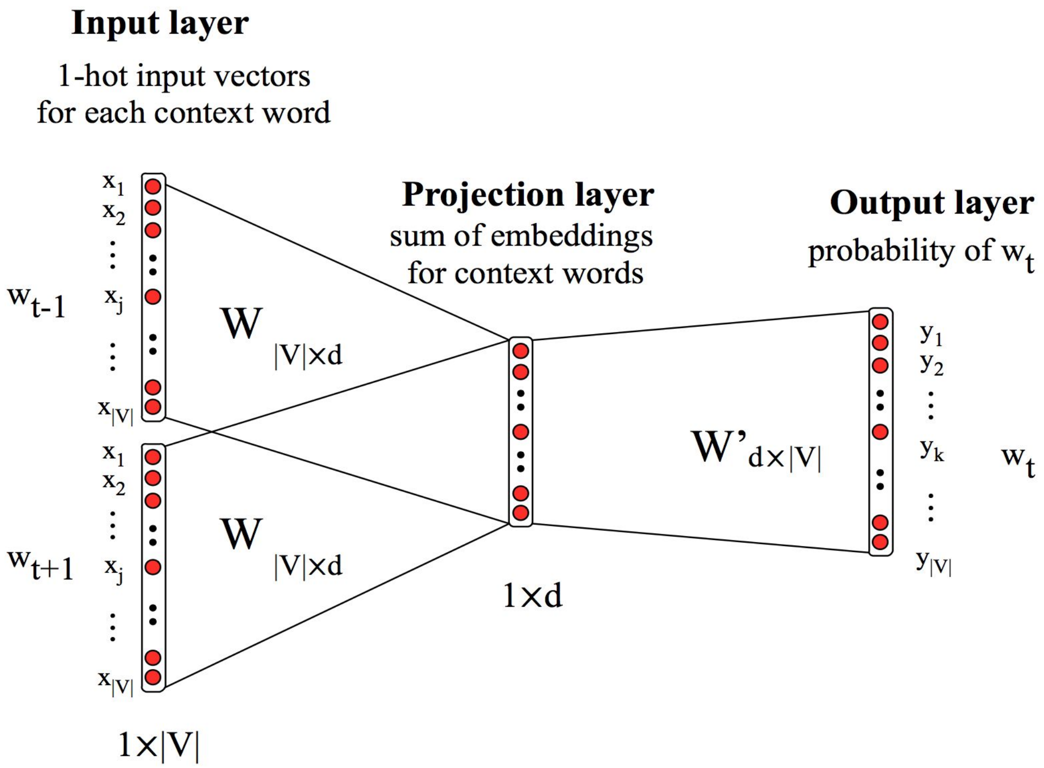
February 14, 2014

[2]Throughout this note, we assume that the words and the contexts come from distinct vocabularies, so that, for example, the vector associated with the word *dog* will be different from the vector associated with the context *dog*. This assumption follows the literature, where it is not motivated. One motivation for making this assumption is the following: consider the case where both the word *dog* and the context *dog* share the same vector v. Words hardly appear in the contexts of themselves, and so the model should assign a low probability to $p(dog|dog)$, which entails assigning a low value to $v \cdot v$ which is impossible.

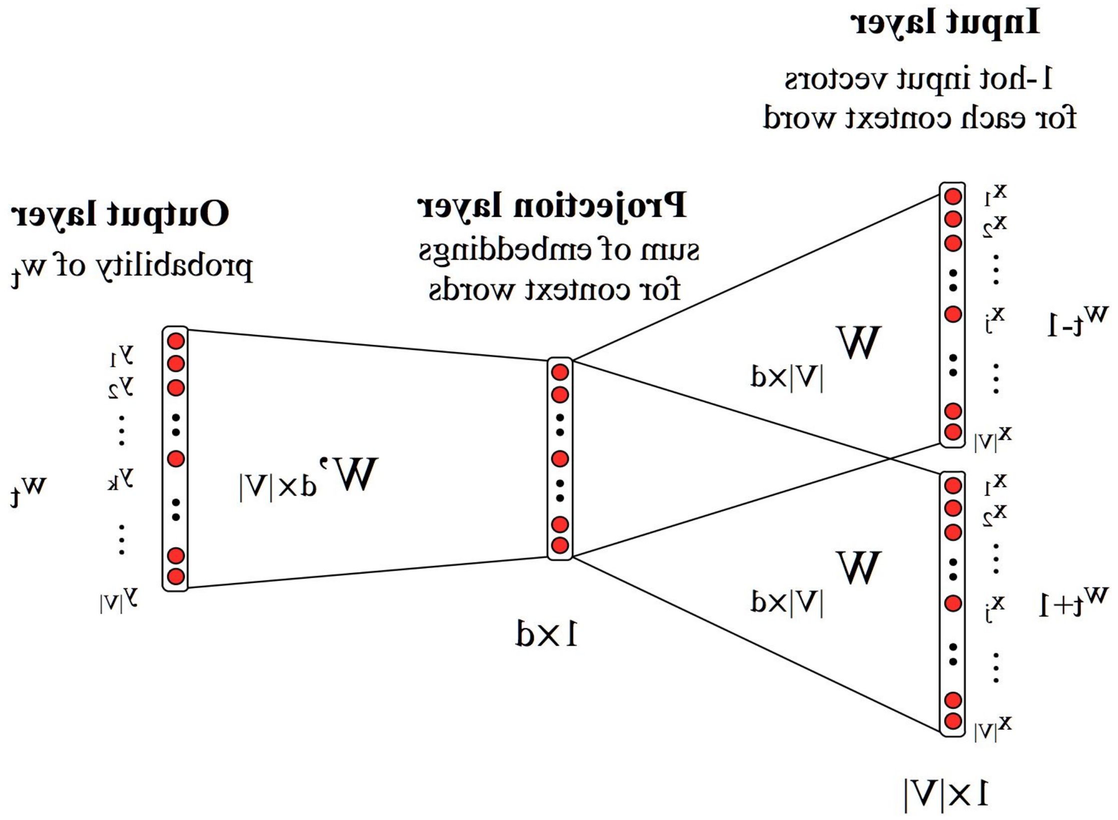
Two Variants of Word2Vec

1. **SkipGram** (what we’ve seen so far): Predict context (outside) words given the center word.
2. **CBOW**: Predict center word from the sum of surrounding word vectors.

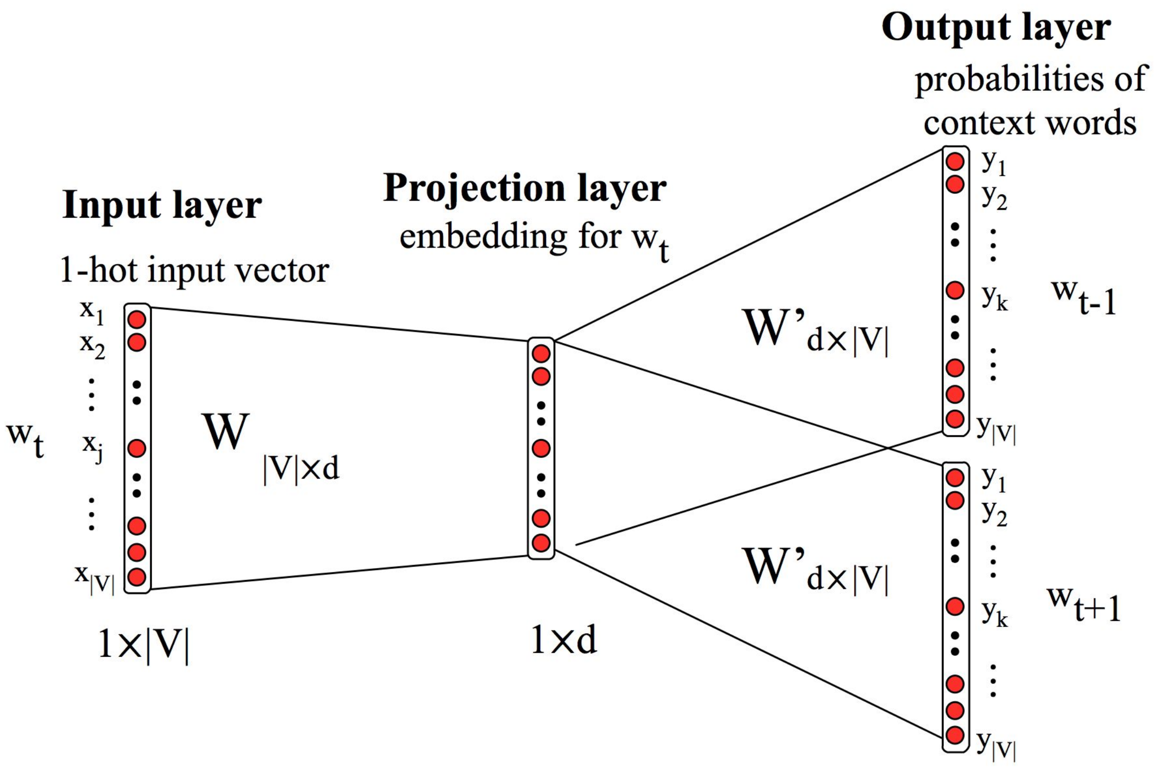
CBOW in practice



Skipgram is like the reverse of CBOW?



Okay, okay just kidding, here's the real SkipGram diagram:



Contextualized Word Embeddings

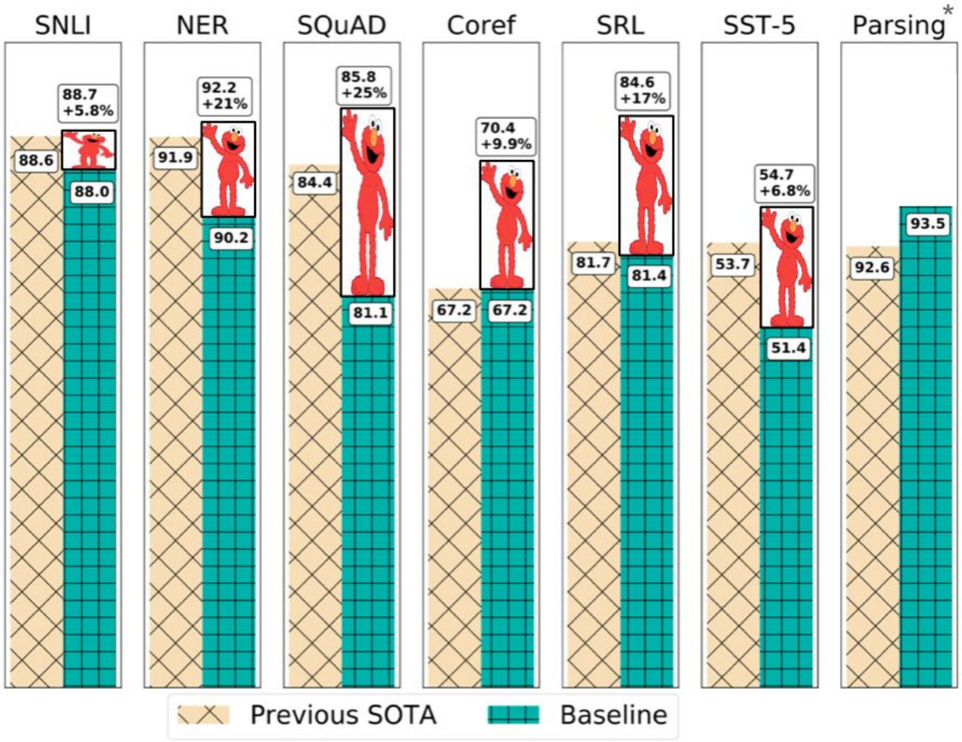
Premise: define a vector for each *token* based its context **in the data**

- How do we get context? RNN-based Neural LM's
  - Hidden state h_i at timestep i represents the left-context of token x_i
  - Compute an analogous right-context by training a right-to-left LM
  - Simplest approach: concatenate the two contexts to get an embedding

Contextualized Word Embeddings

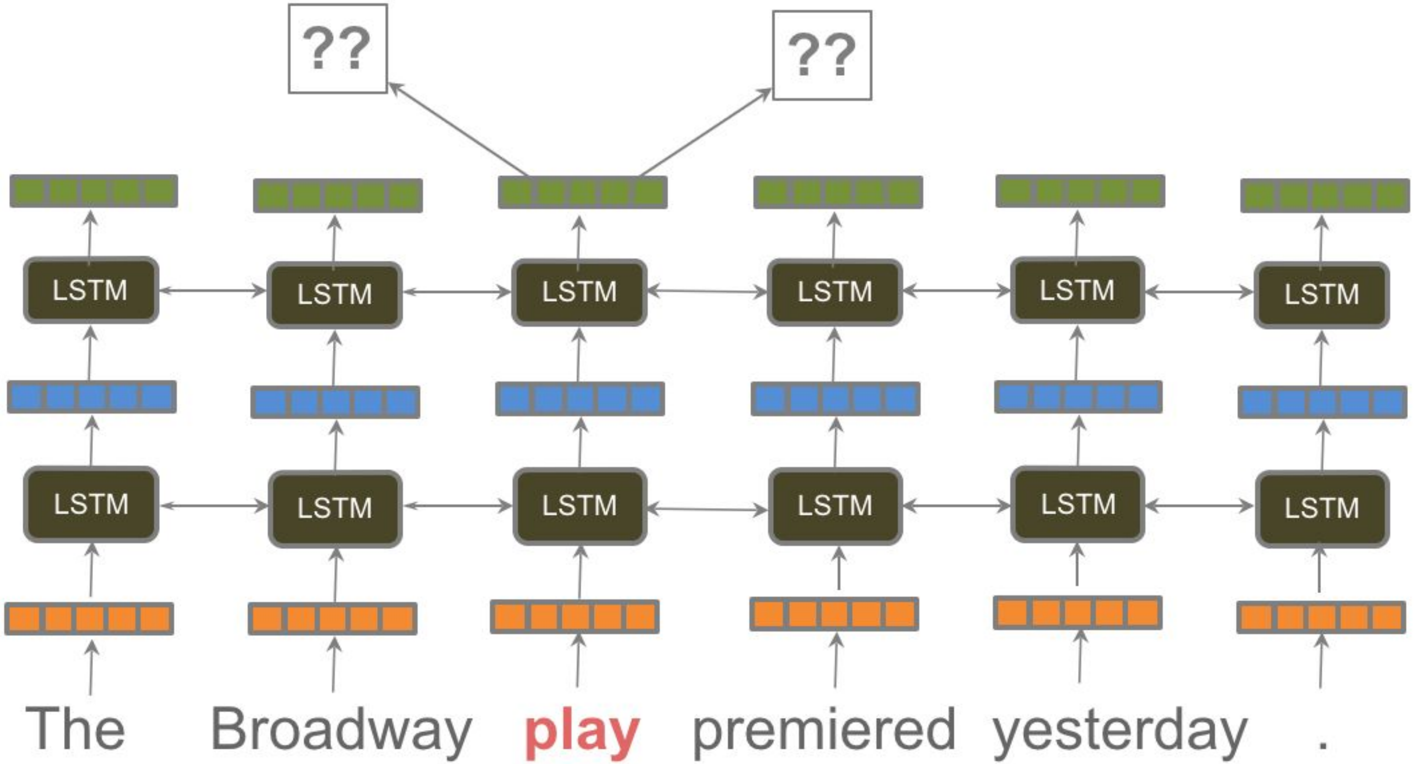
ELMo (Peters et al., 2018)

- Used a multi-layer, bidirectional LSTM
- Using ELMo instead of static vectors: instant SOTA on a lot of benchmark tasks



*Kitaev and Klein, ACL 2018 (see also Joshi et al., ACL 2018)

ELMo, visually



BERT

BERT (Devlin et al., 2019) :

- Instead of RNN, it uses *transformers*.
- Learning objectives:
    - **Masked Language Model** (MLM): randomly mask out words for model to predict.
    - **Next Sentence Prediction** (NSP): given a pair of sentences, does the second sentence follow the first one? Helpful for understanding the relationship between sentences (for QA, NLI, etc.).

BERT

Pretrain + finetune like we discussed!

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT's Performance on GLUE tasks (Devlin et al., 2019)

BERTology

Many many ideas are built on BERT:

- Multilingual BERT (Devlin et al., 2019):
  - pretrained on 104 language.
- RoBERTa (Liu et al., 2019):
  - removed NSP objective;
  - trained with larger mini-batches
  - larger learning rates;
  - more data;
  - longer pretraining time.
- Overview: Rogers et al. (2020)
- T5 (Raffel et al., 2019): model that explored many different options

Q & A