Natural Language Processing

Info 159/259

Lecture 5: Attention and transformers

Many slides & instruction ideas borrowed from:

David Bamman & Mohit Iyyer

Logistics

- Quiz 2 will be out this Friday (due next Monday Feb 5).
- Homework 1 is out & due next Tuesday, Feb 6 (11:59 pm)
    - Homework 2 will be out early next week.

- Tonight:
    - Recap on Dense Representation and Word Embeddings
    - Text Classification with Attention, Transformers, etc.

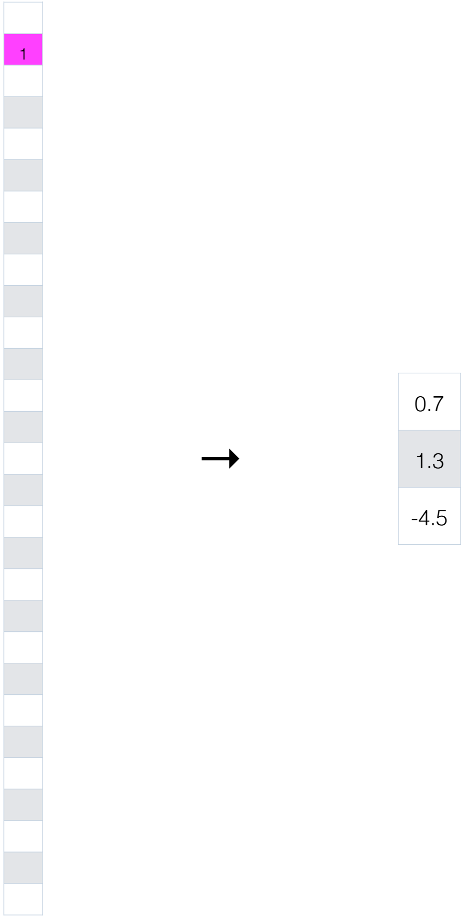
Sparse vectors

“aardvark”

V-dimensional vector, single 1 for the identity of the element

A	0
a	0
aa	0
aal	0
aalii	0
aam	0
Aani	0
aardvark	1
aardwolf	0
...	0
zymotoxic	0
zymurgy	0
Zyrenian	0
Zyrian	0
Zyryan	0
zythem	0
Zythia	0
zythum	0
Zyzomys	0
Zyzzogeton	0

Dense vectors



Dimensionality reduction

…	…
the	1
a	0
an	0
for	0
in	0
on	0
dog	0
cat	0
…	…

the is a point in V-dimensional space

the
4.1
-0.9

the is a point in 2-dimensional space

Dense vectors from prediction

- Learning low-dimensional representations of words by framing a predicting task: using context to predict words in a surrounding window
- Transform this into a supervised prediction problem; similar to language modeling but we're ignoring order within the context window

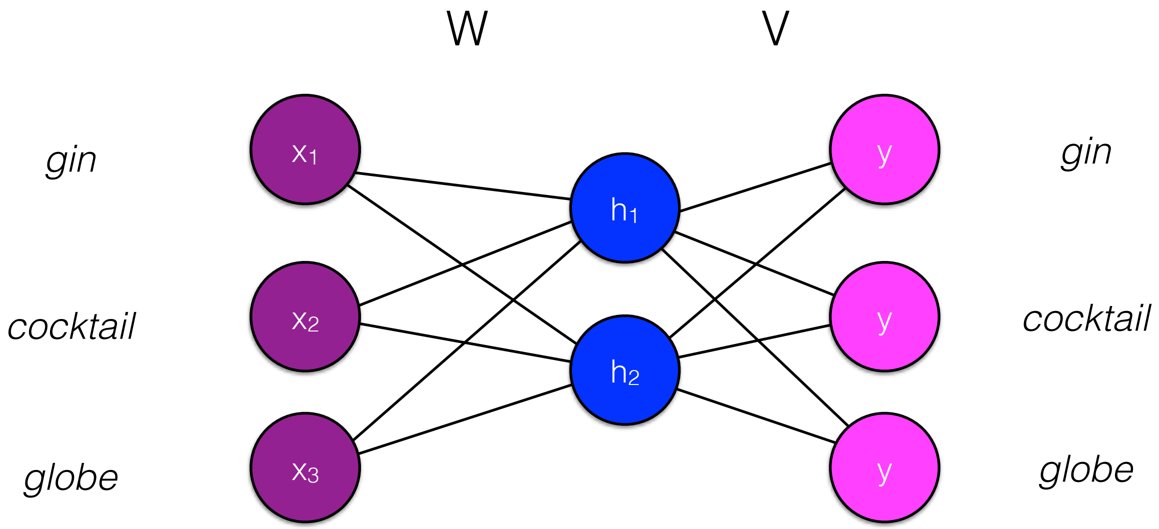
Dense vectors from prediction

Word2vec Skipgram model (Mikolov et al. 2013): given a single word in a sentence, predict the words in a context window around it.

a cocktail with gin and seltzer

x	y
gin	a
gin	cocktail
gin	with
gin	and
gin	seltzer

Window size = 3

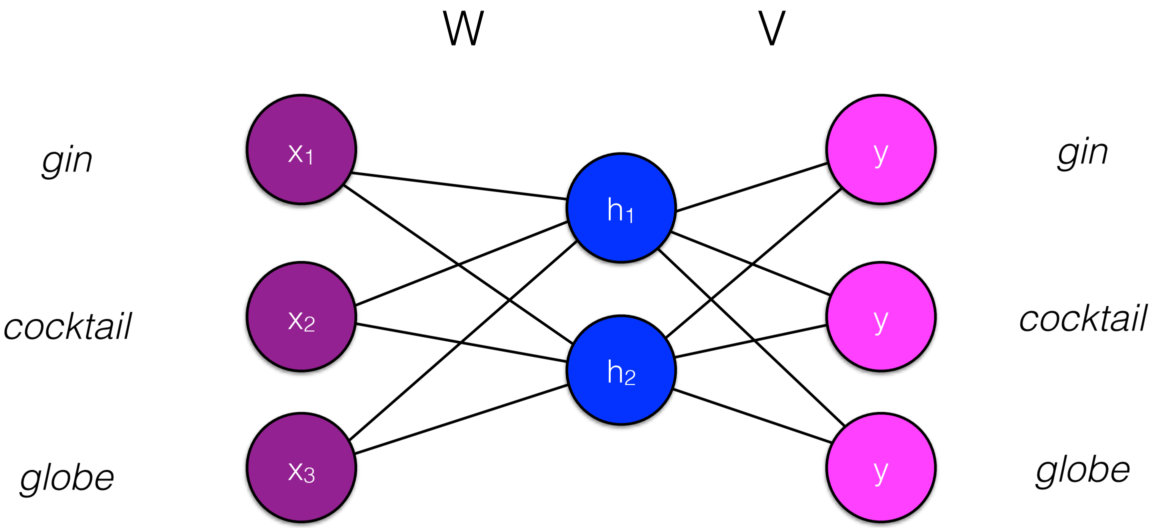


	x
gin	1
cocktail	0
globe	0

W	
-0.5	1.3
0.4	0.08
1.7	3.1

V		
4.1	0.7	0.1
-0.9	1.3	0.3

y
0
1
0



Only one of the inputs is nonzero.

= the inputs are really W_{gin}

W	
-0.5	1.3
0.4	0.08
1.7	3.1

V		
4.1	0.7	0.1
-0.9	1.3	0.3

x	W	
	0.13	0.56
	-1.75	0.07
	0.80	1.19
	-0.11	1.38
	-0.62	-1.46
	-1.16	-1.24
	0.99	-0.26
	-1.46	-0.85
	0.79	0.47
	0.06	-1.21
	-0.31	0.00
1	-1.01	-2.52
	-1.50	-0.14
	-0.14	0.01
	-0.13	-1.76
	-1.08	-0.56
	-0.17	-0.74
	0.31	1.03
	-0.24	-0.84
	-0.79	-0.18

$$x^\top W =$$

-1.01	-2.52

This is the embedding of the context

Word embeddings

- Similarly, V has one H-dimensional vector for each element in the vocabulary (for the words that are being predicted)

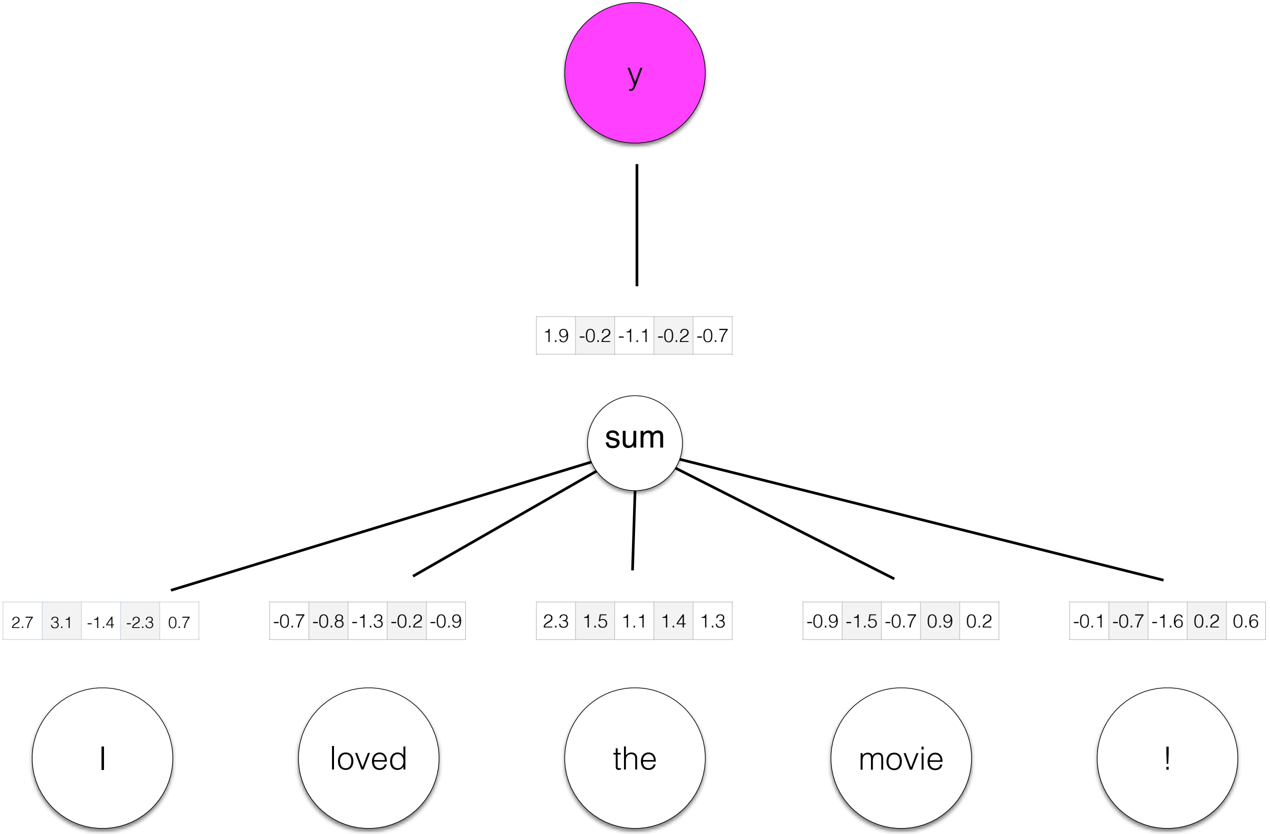
V			
gin	cocktail	cat	globe
4.1	0.7	0.1	1.3
-0.9	1.3	0.3	-3.4

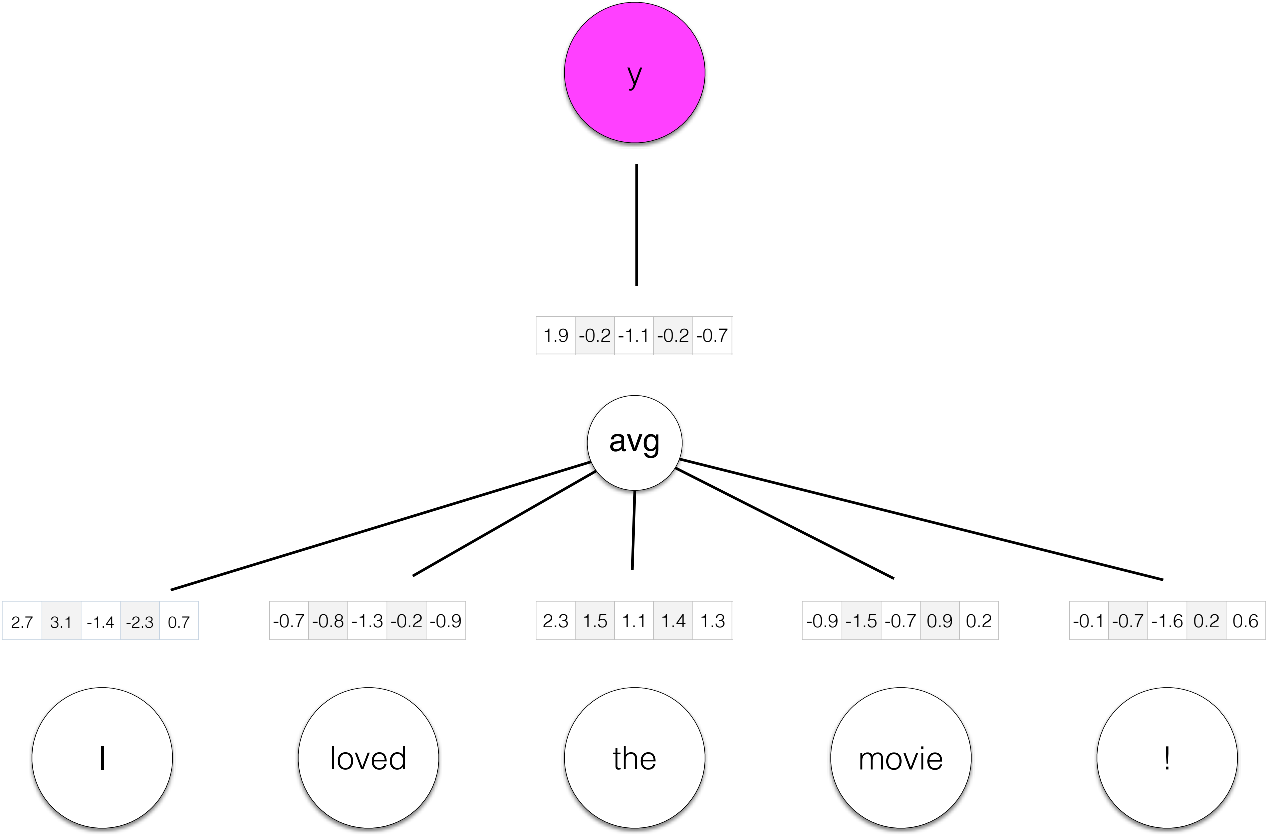
This is the embedding of the word

	1	2	3	4	…	50
the	0.418	0.24968	-0.41242	0.1217	…	-0.17862
,	0.013441	0.23682	-0.16899	0.40951	…	-0.55641
.	0.15164	0.30177	-0.16763	0.17684	…	-0.31086
of	0.70853	0.57088	-0.4716	0.18048	…	-0.52393
to	0.68047	-0.039263	0.30186	-0.17792	…	0.13228
…	…	…	…	…	…	…
chanty	0.23204	0.025672	-0.70699	-0.04547	…	0.34108
kronik	-0.60921	-0.67218	0.23521	-0.11195	…	0.85632
rolonda	-0.51181	0.058706	1.0913	-0.55163	…	0.079711
zsombor	-0.75898	-0.47426	0.4737	0.7725	…	0.84014
sandberger	0.072617	-0.51393	0.4728	-0.52202	…	0.23096

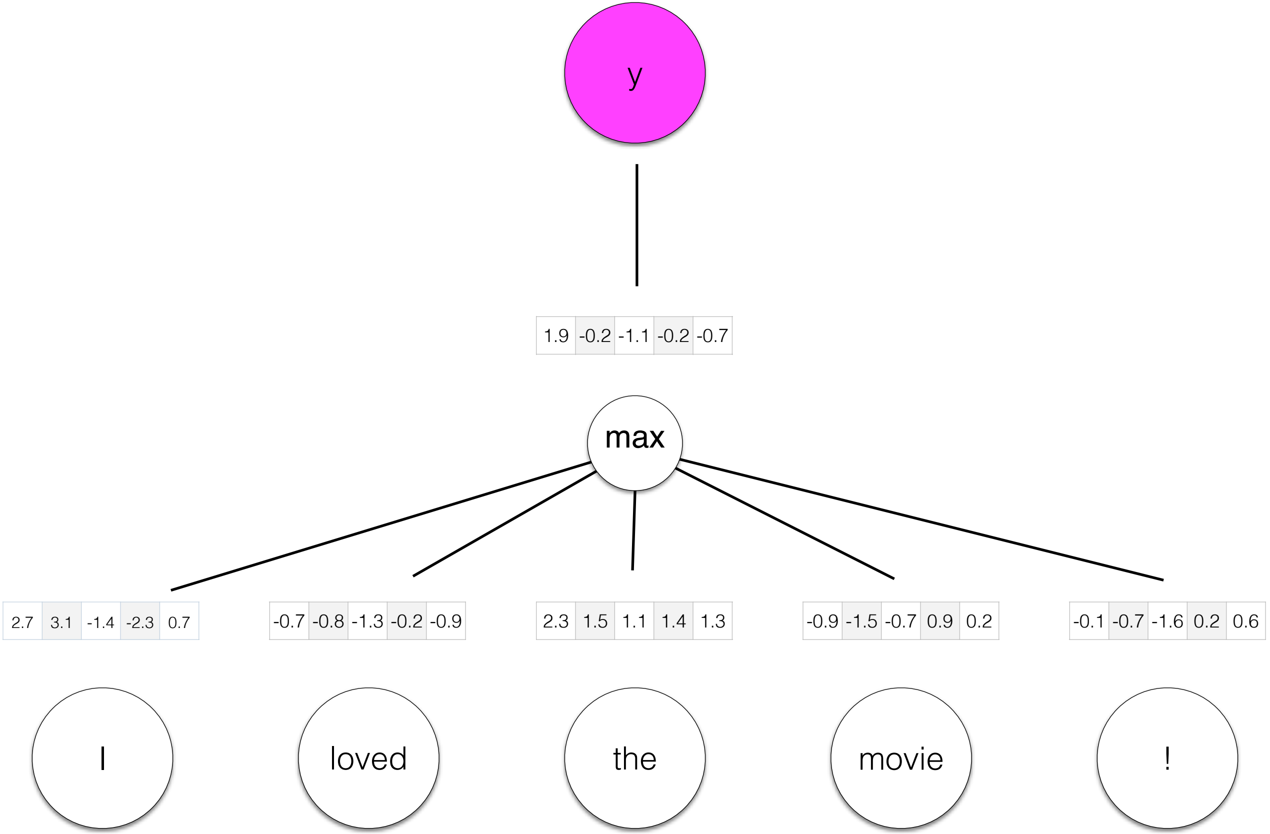
https://nlp.stanford.edu/projects/glove/

How do we use word embeddings for document classification?

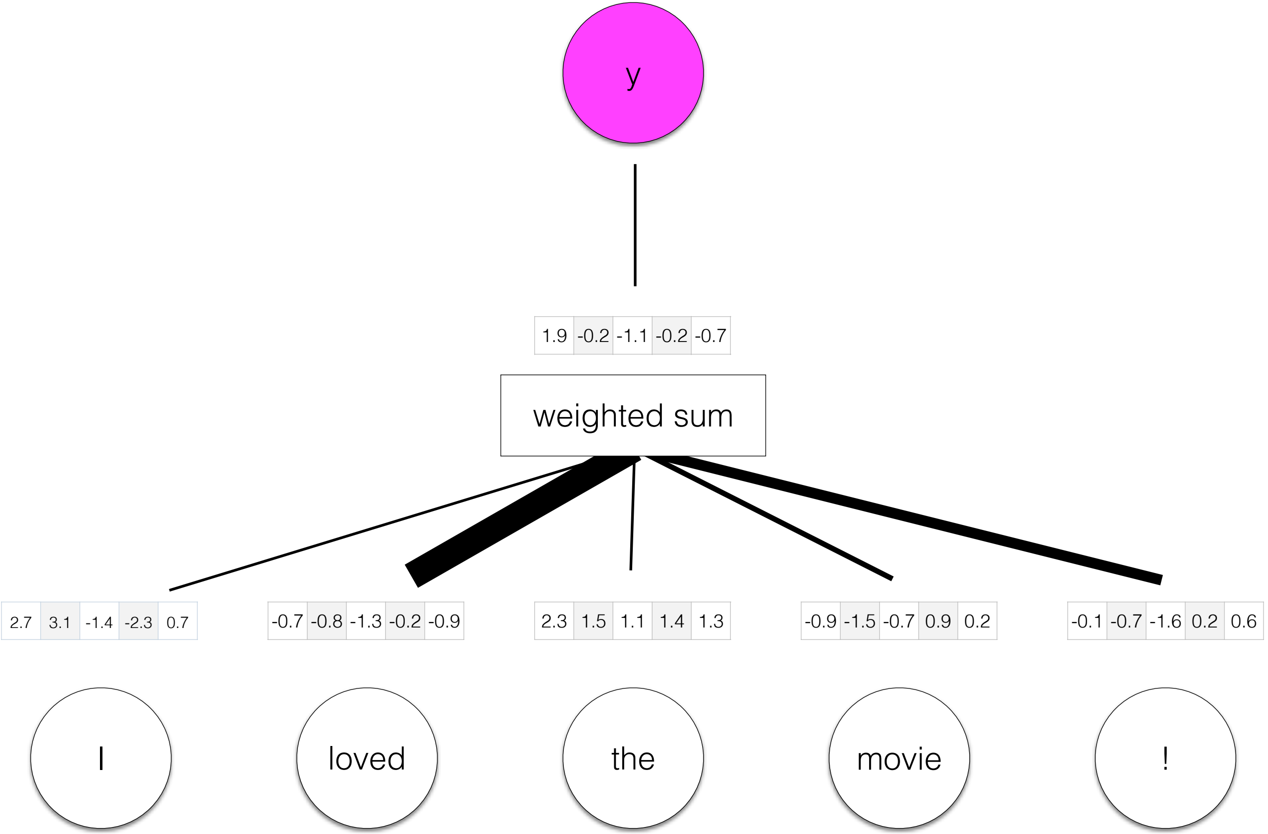

y
1.9 -0.2 -1.1 -0.2 -0.7
sum
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!



Iyyer et al. (2015), "Deep Unordered Composition Rivals Syntactic Methods for Text Classification" (ACL)



y
1.9 -0.2 -1.1 -0.2 -0.7
max
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6
I
loved
the
movie
!



y

1.9	-0.2	-1.1	-0.2	-0.7

weighted sum

2.7	3.1	-1.4	-2.3	0.7

-0.7	-0.8	-1.3	-0.2	-0.9

2.3	1.5	1.1	1.4	1.3

-0.9	-1.5	-0.7	0.9	0.2

-0.1	-0.7	-1.6	0.2	0.6

I

loved

the

movie

!

Attention

- Let's incorporate structure (and parameters) into a network that captures which elements (tokens) in the input we should be attending to (and which we can ignore).

$v \in \mathscr{R}^H$

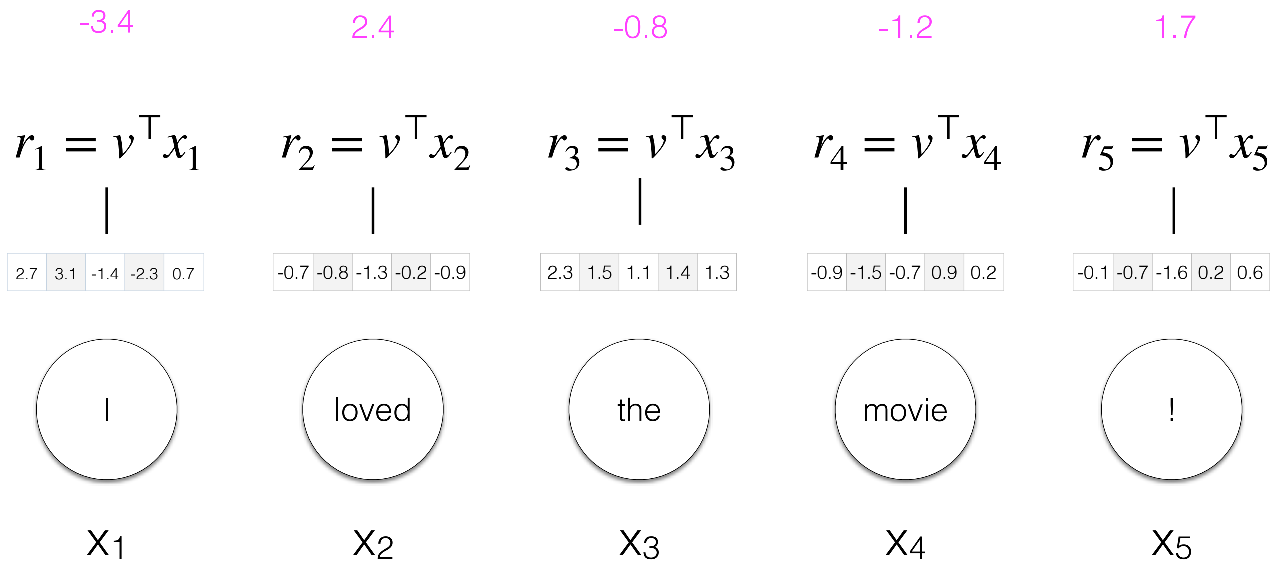
2.7	3.1	-1.4	-2.3	0.7

Define v to be a vector to be learned; think of it as an “important word” vector. The dot product here measures how similar each input vector is to that “important word” vector

2.7	3.1	-1.4	-2.3	0.7	I	X_1
-0.7	-0.8	-1.3	-0.2	-0.9	loved	X_2
2.3	1.5	1.1	1.4	1.3	the	X_3
-0.9	-1.5	-0.7	0.9	0.2	movie	X_4
-0.1	-0.7	-1.6	0.2	0.6	!	X_5

$v \in \mathscr{R}^H$

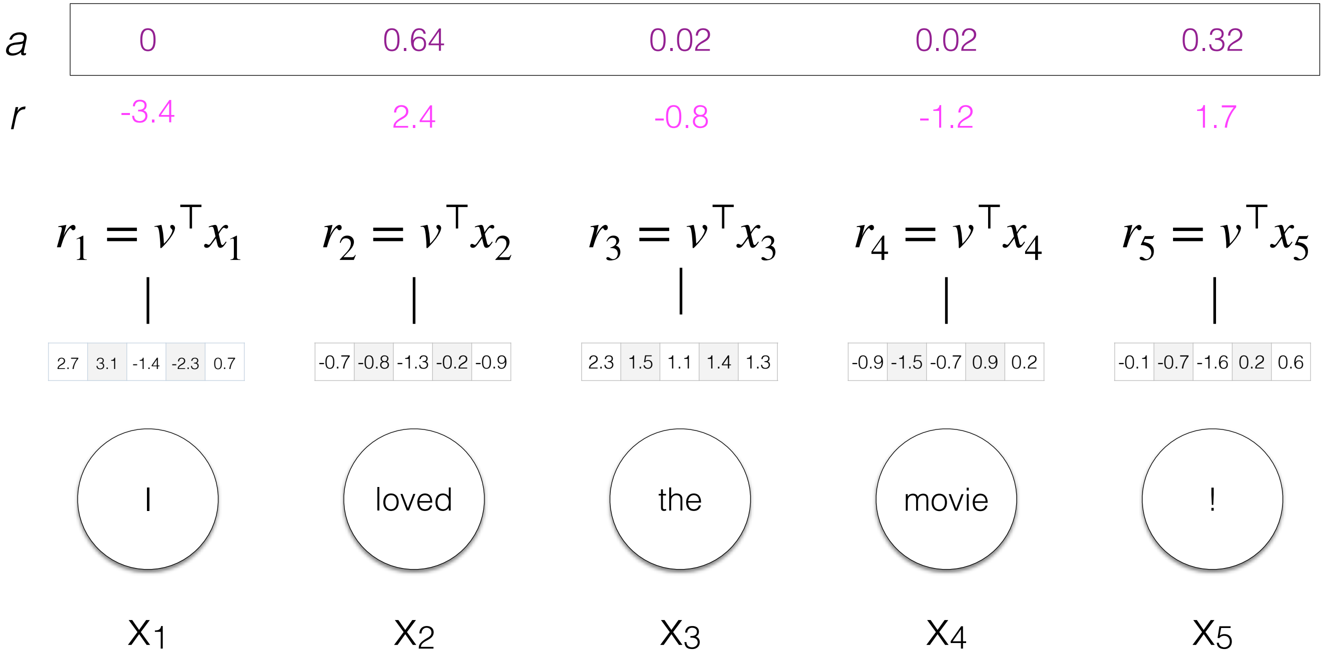
2.7	3.1	-1.4	-2.3	0.7



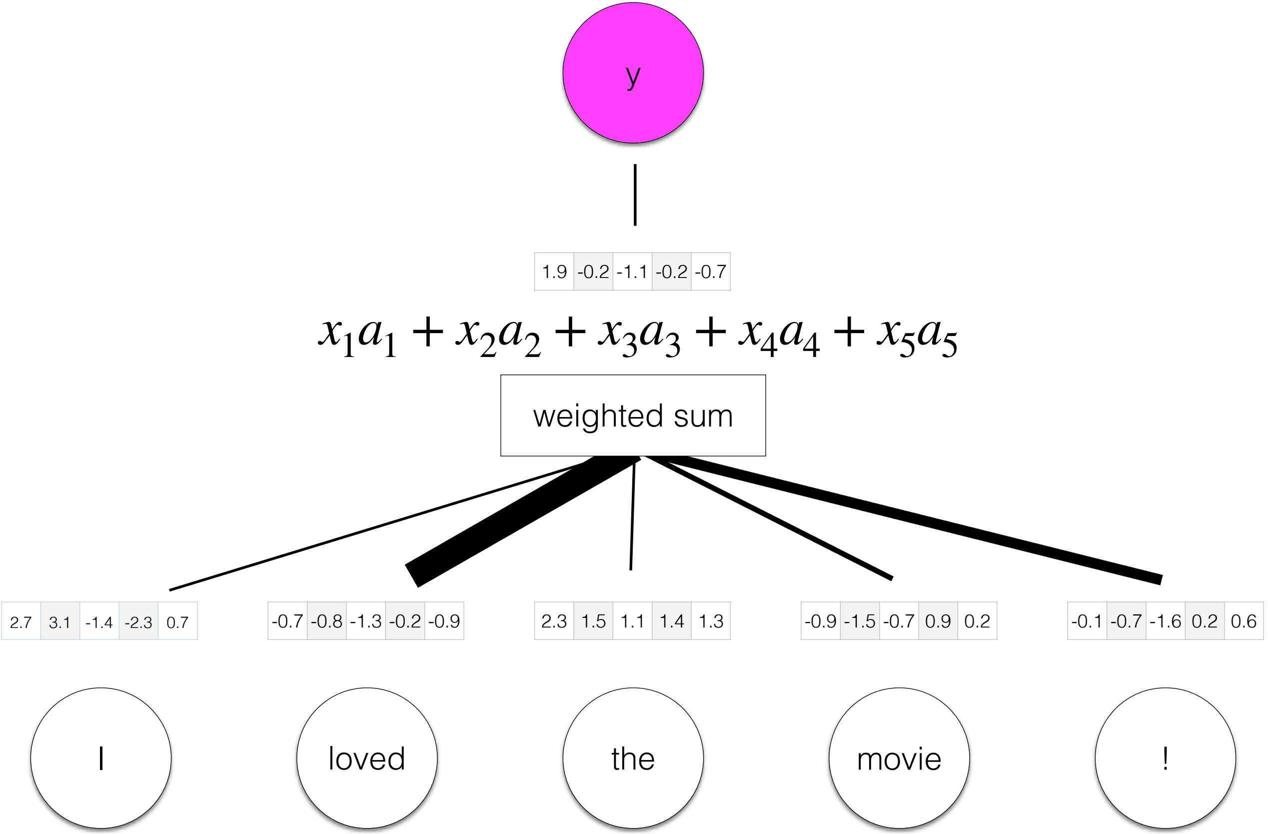
-3.4	2.4	-0.8	-1.2	1.7
$r_1 = v^\top x_1$	$r_2 = v^\top x_2$	$r_3 = v^\top x_3$	$r_4 = v^\top x_4$	$r_5 = v^\top x_5$
2.7 3.1 -1.4 -2.3 0.7	-0.7 -0.8 -1.3 -0.2 -0.9	2.3 1.5 1.1 1.4 1.3	-0.9 -1.5 -0.7 0.9 0.2	-0.1 -0.7 -1.6 0.2 0.6
I	loved	the	movie	!
x_1	x_2	x_3	x_4	x_5

Convert r into a vector of normalized weights that sum to 1.

$$a = \text{softmax}(r)$$



a	0	0.64	0.02	0.02	0.32
r	-3.4	2.4	-0.8	-1.2	1.7
	$r_1 = v^\top x_1$	$r_2 = v^\top x_2$	$r_3 = v^\top x_3$	$r_4 = v^\top x_4$	$r_5 = v^\top x_5$
	2.7 3.1 -1.4 -2.3 0.7	-0.7 -0.8 -1.3 -0.2 -0.9	2.3 1.5 1.1 1.4 1.3	-0.9 -1.5 -0.7 0.9 0.2	-0.1 -0.7 -1.6 0.2 0.6
	I	loved	the	movie	!
	X_1	X_2	X_3	X_4	X_5



y

1.9 -0.2 -1.1 -0.2 -0.7

$$x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5$$

weighted sum

2.7 3.1 -1.4 -2.3 0.7

-0.7 -0.8 -1.3 -0.2 -0.9

2.3 1.5 1.1 1.4 1.3

-0.9 -1.5 -0.7 0.9 0.2

-0.1 -0.7 -1.6 0.2 0.6

I

loved

the

movie

!

Attention

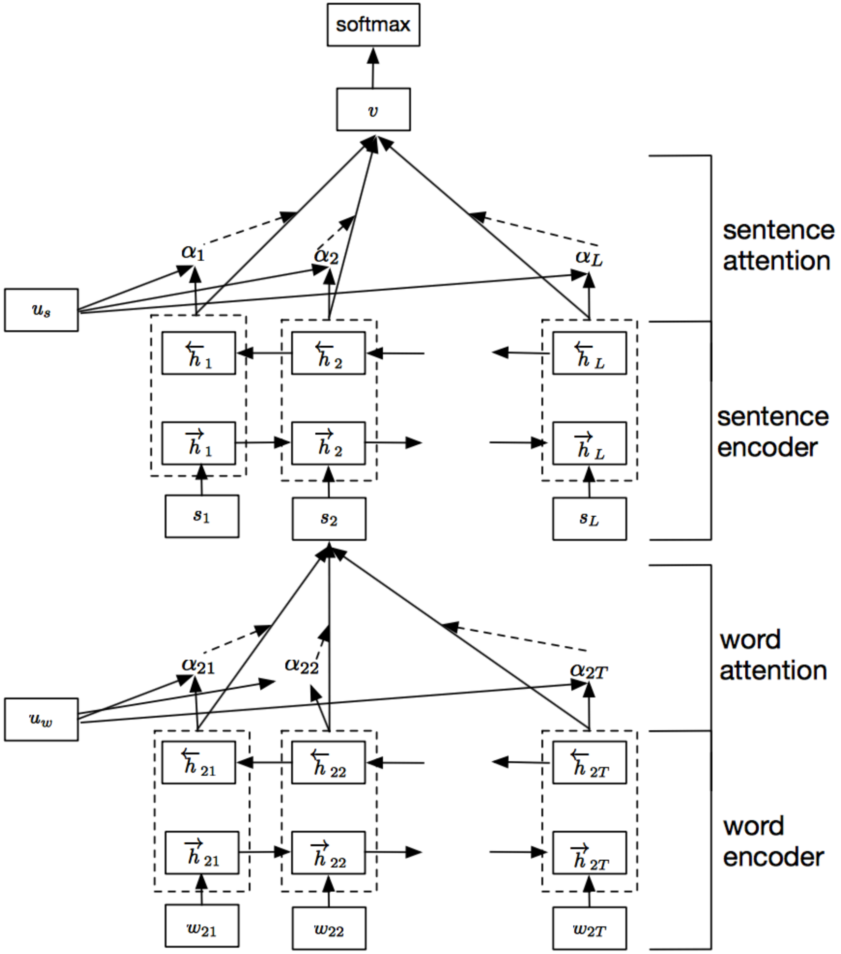
- Lots of variations on attention:
    - Linear transformation of x before dotting with v
    - Non-linearities after each operation.
    - “Multi-head attention”: multiple v vectors to capture different aspects that should be attended to in the input.
    - Hierarchical attention (sentence representation with attention over words + document representation with attention over sentences).

attention over sentences

bidirectional GRU over sentence representations

attention over words

bidirectional GRU over word representations

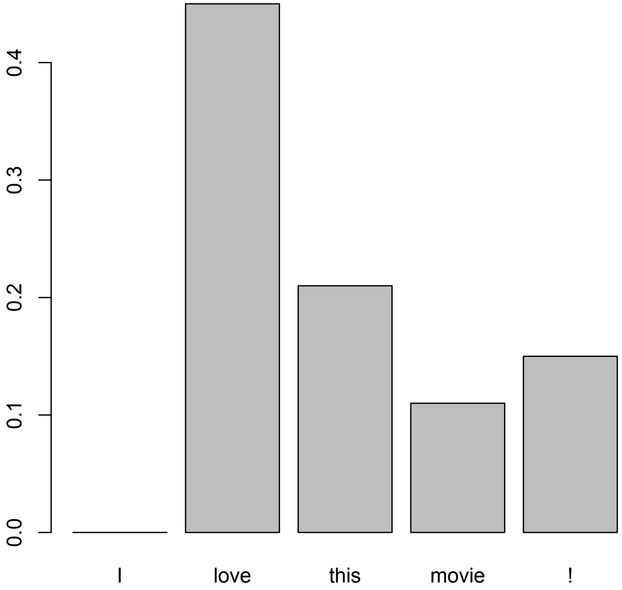


Yang et al. (2016), “Hierarchical Attention Networks for Document Classification”

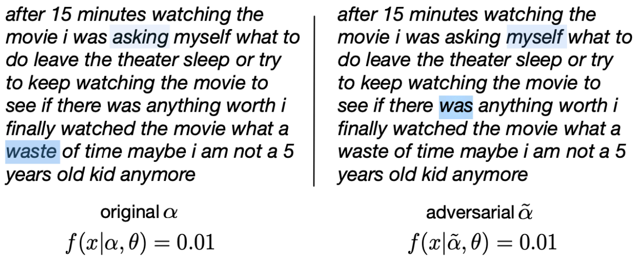
Attention

- Attention gives us a normalized weight for every token in a sequence that tells us how important that word was for the prediction

- This can be useful for visualization



Attention



after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

original α

$$f(x|\alpha,\theta) = 0.01$$

after 15 minutes watching the movie i was asking myself what to do leave the theater sleep or try to keep watching the movie to see if there was anything worth i finally watched the movie what a waste of time maybe i am not a 5 years old kid anymore

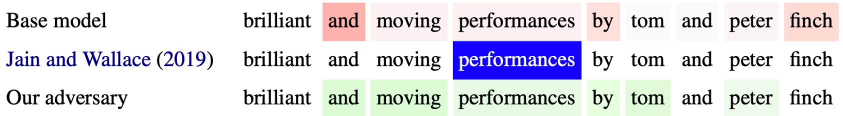
adversarial $\tilde{\alpha}$

$$f(x|\tilde{\alpha},\theta) = 0.01$$

[Submitted on 26 Feb 2019 (v1), last revised 8 May 2019 (this version, v3)]

Attention is not Explanation

Sarthak Jain, Byron C. Wallace



Base model	brilliant	and	moving	performances	by	tom	and	peter	finch
Jain and Wallace (2019)	brilliant	and	moving	performances	by	tom	and	peter	finch
Our adversary	brilliant	and	moving	performances	by	tom	and	peter	finch

Figure 2: Attention maps for an IMDb instance (all predicted as positive with score > 0.998), showing that in practice it is difficult to learn a distant adversary which is consistent on all instances in the training set.

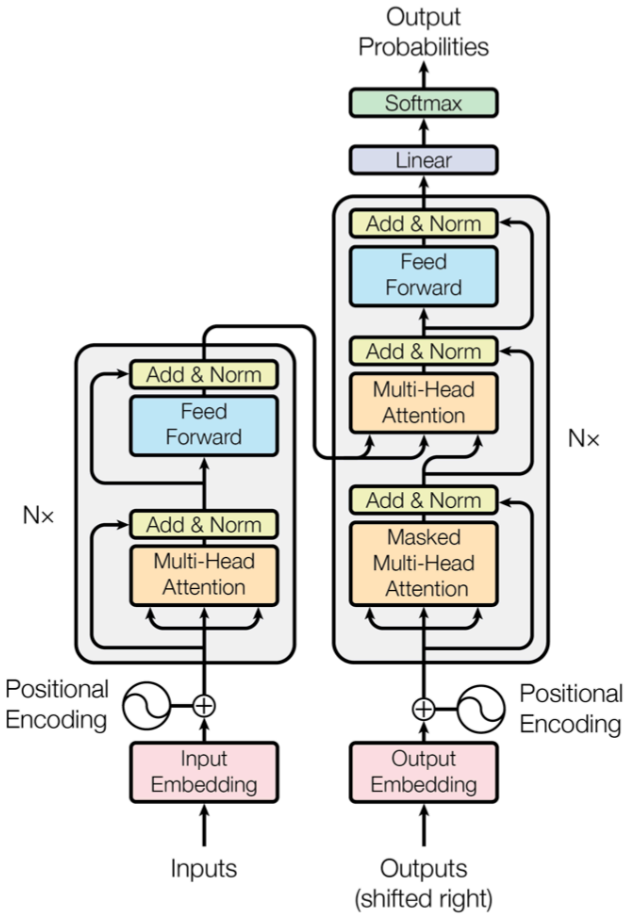
[Submitted on 13 Aug 2019 (v1), last revised 5 Sep 2019 (this version, v2)]

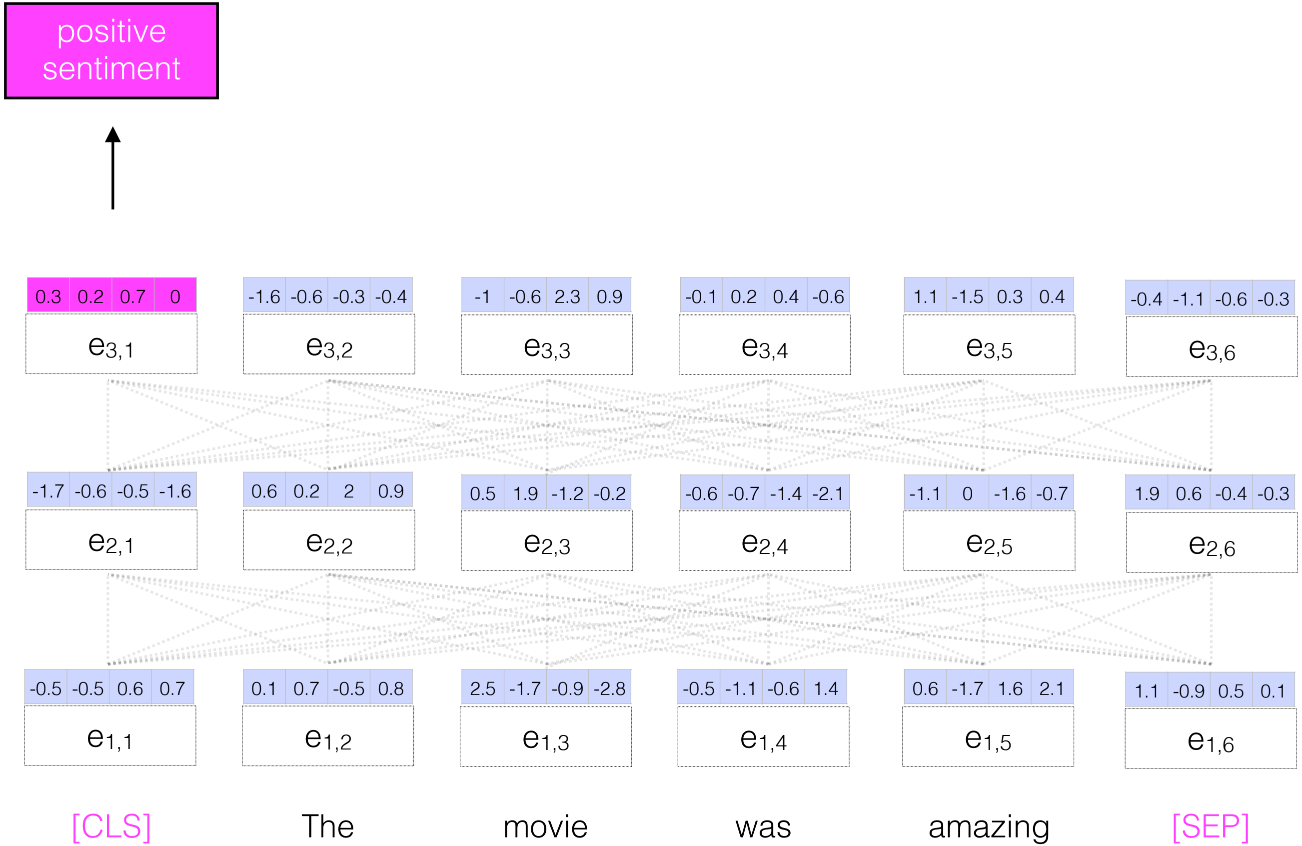
Attention is not not Explanation

Sarah Wiegreffe, Yuval Pinter

Transformers

- Vaswani et al. 2017, “Attention is All You Need”
- Transforms map an input sequence of vectors to an output sequence of vectors of the same dimensionality

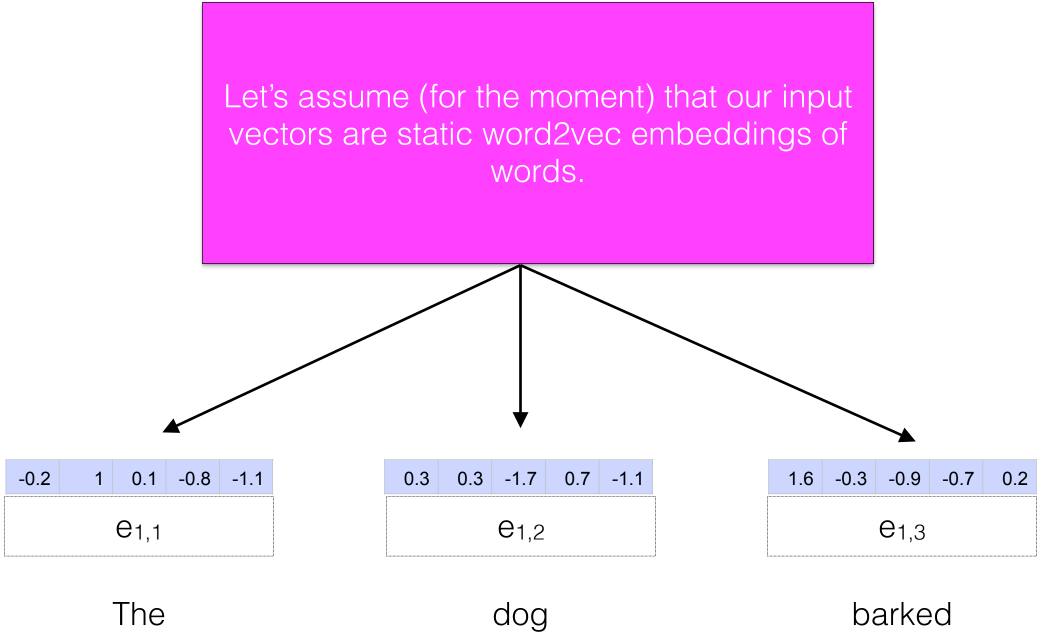




positive sentiment

0.3 0.2 0.7 0	-1.6 -0.6 -0.3 -0.4	-1 -0.6 2.3 0.9	-0.1 0.2 0.4 -0.6	1.1 -1.5 0.3 0.4	-0.4 -1.1 -0.6 -0.3
$e_{3,1}$	$e_{3,2}$	$e_{3,3}$	$e_{3,4}$	$e_{3,5}$	$e_{3,6}$
-1.7 -0.6 -0.5 -1.6	0.6 0.2 2 0.9	0.5 1.9 -1.2 -0.2	-0.6 -0.7 -1.4 -2.1	-1.1 0 -1.6 -0.7	1.9 0.6 -0.4 -0.3
$e_{2,1}$	$e_{2,2}$	$e_{2,3}$	$e_{2,4}$	$e_{2,5}$	$e_{2,6}$
-0.5 -0.5 0.6 0.7	0.1 0.7 -0.5 0.8	2.5 -1.7 -0.9 -2.8	-0.5 -1.1 -0.6 1.4	0.6 -1.7 1.6 2.1	1.1 -0.9 0.5 0.1
$e_{1,1}$	$e_{1,2}$	$e_{1,3}$	$e_{1,4}$	$e_{1,5}$	$e_{1,6}$
[CLS]	The	movie	was	amazing	[SEP]

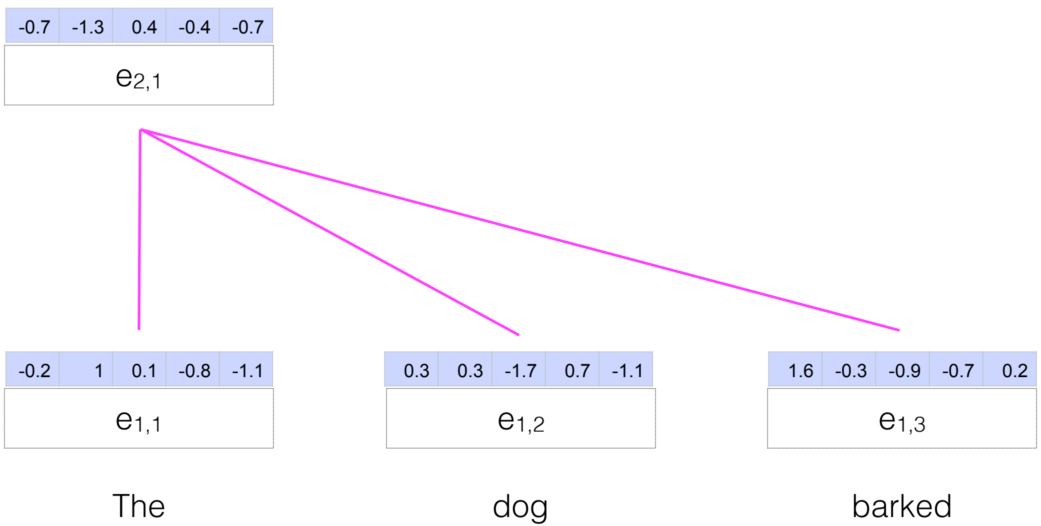
Self-Attention



Let’s assume (for the moment) that our input vectors are static word2vec embeddings of words.

-0.2 1 0.1 -0.8 -1.1	0.3 0.3 -1.7 0.7 -1.1	1.6 -0.3 -0.9 -0.7 0.2
$e_{1,1}$	$e_{1,2}$	$e_{1,3}$
The	dog	barked

The value for time step j at layer i is the result of attention over all time steps in the previous layer i-1



-0.7	-1.3	0.4	-0.4	-0.7

$e_{2,1}$

-0.2	1	0.1	-0.8	-1.1

$e_{1,1}$

The

0.3	0.3	-1.7	0.7	-1.1

$e_{1,2}$

dog

1.6	-0.3	-0.9	-0.7	0.2

$e_{1,3}$

barked

- Let’s separate out the different functions that an input vector has in attention by transforming it into separate representations for its role in a weighted sum (the value) from the roles used to assess compatibility (the query and key).

query $q_{1,1} \in \mathbb{R}^{37} \quad (e_{1,1}W^Q)$

key $k_{1,1} \in \mathbb{R}^{37} \quad (e_{1,1}W^K)$

value $v_{1,1} \in \mathbb{R}^{100} \quad (e_{1,1}W^V)$

original value $e_{1,1} \in \mathbb{R}^{100}$

$e_{1,1}$

The

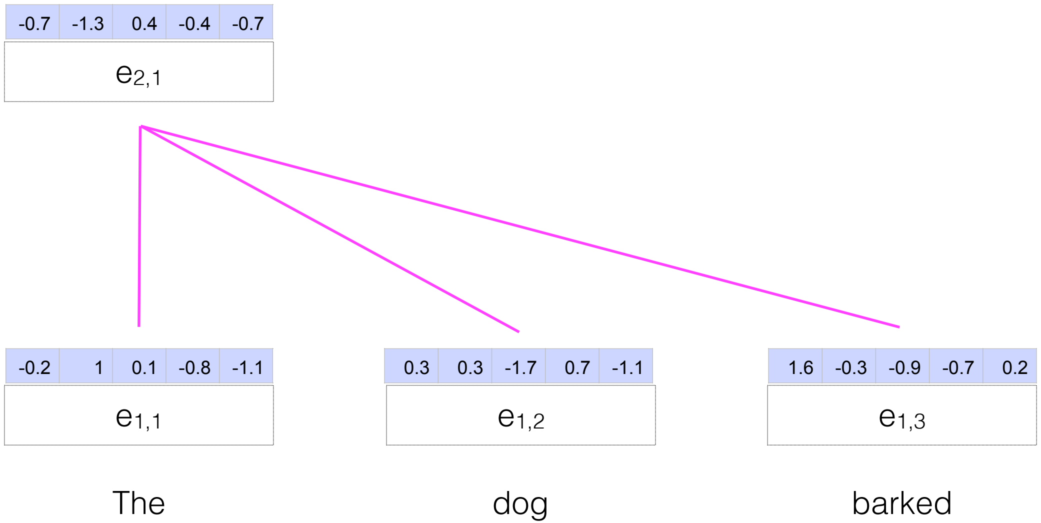
$$W^Q \in \mathbb{R}^{100\times37}$$

$$W^K \in \mathbb{R}^{100\times37}$$

$$W^V \in \mathbb{R}^{100\times100}$$

These are all parameters we *learn*. 100 is the original input dimension; 37 is a hyper-parameter we choose.

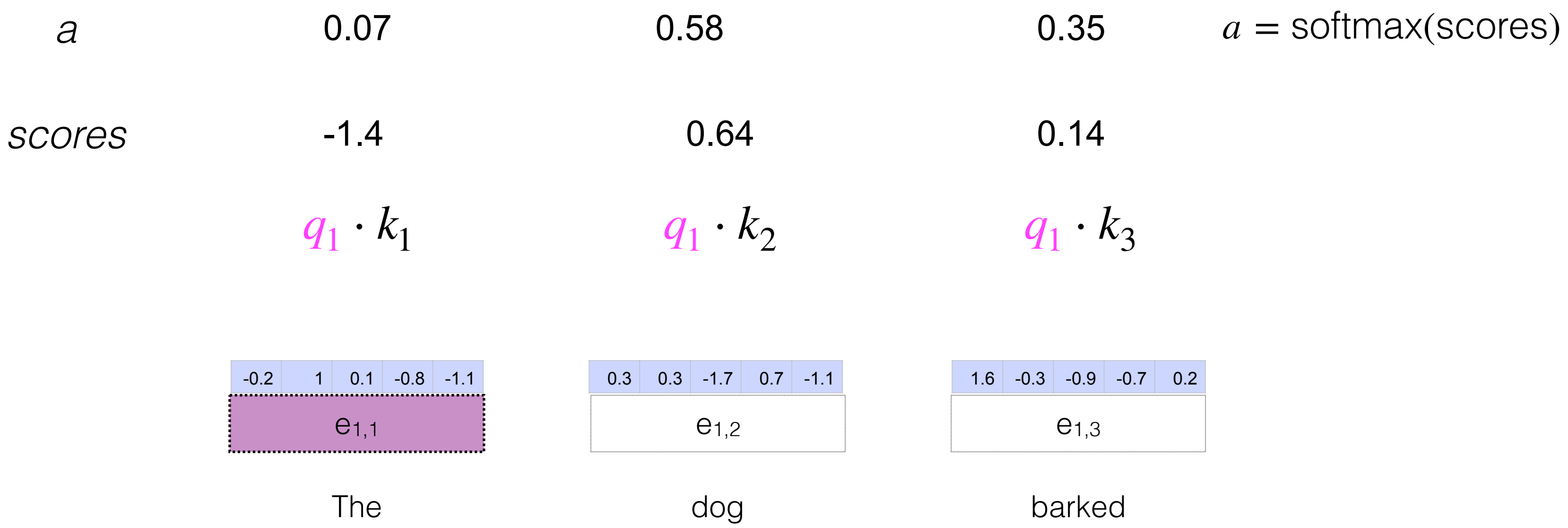
Self attention **from “The”** at position 1 to every token in the sentence



$e_{2,1}$	-0.7	-1.3	0.4	-0.4	-0.7
$e_{1,1}$	-0.2	1	0.1	-0.8	-1.1
$e_{1,2}$	0.3	0.3	-1.7	0.7	-1.1
$e_{1,3}$	1.6	-0.3	-0.9	-0.7	0.2

- The compatibility score between two words is the dot product between their respective query and key vectors.

$$score(e_i, e_j) = q_i \cdot k_j$$



a	0.07	0.58	0.35	a = softmax(scores)
scores	-1.4	0.64	0.14	
	$q_1 \cdot k_1$	$q_1 \cdot k_2$	$q_1 \cdot k_3$	

-0.2	1	0.1	-0.8	-1.1

$e_{1,1}$

The

0.3	0.3	-1.7	0.7	-1.1

$e_{1,2}$

dog

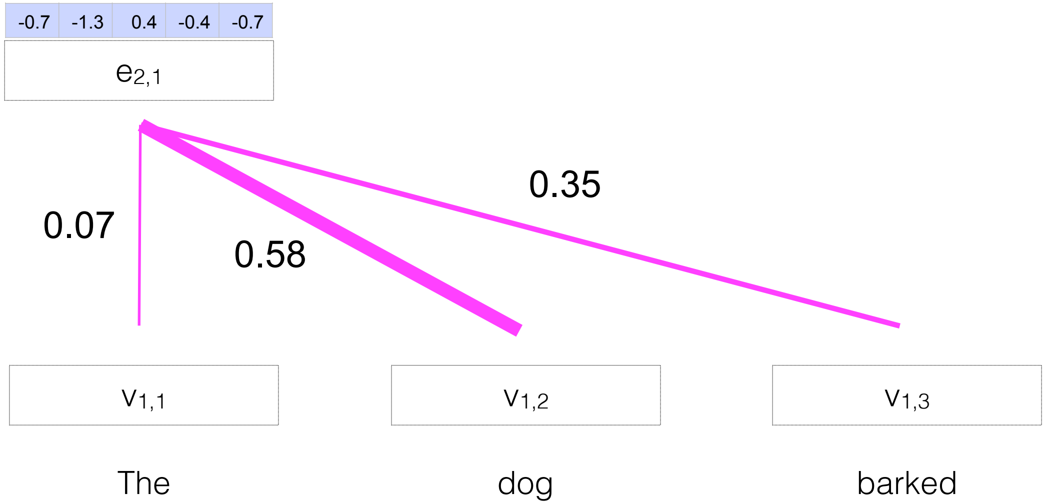
1.6	-0.3	-0.9	-0.7	0.2

$e_{1,3}$

barked

- The output of attention is a weighted sum over the values of the previous layer.

If the dimensionality of v is 100, how big is this vector?



Computing Self attention

- Attention in transformers is essentially a set of learned parameters (W^Q, W^K, W^V) and a mathematical expression for how an input is transformed into an output through operations involving those parameters.

$$Q = XW^Q; K = XW^K; V = XW^V$$

$$\text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Scaled by the dimensionality of the key vectors ($\sqrt{d_k}$)

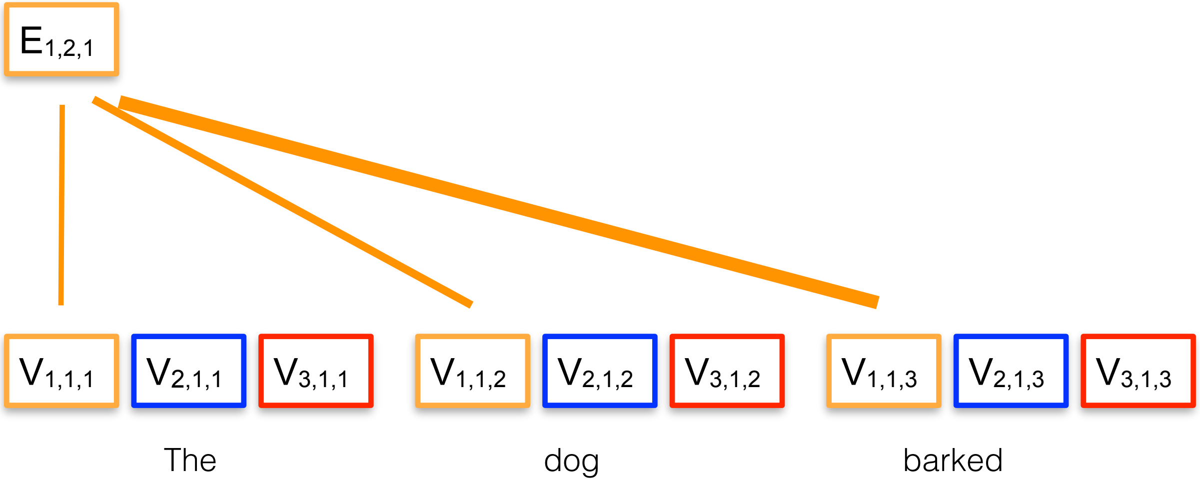
Multihead attention

- Attention provides one view on the data; just like we use multiple filters in CNNs to provide multiple perspectives (by learning separate parameters for each one), so too can we learn multiple perspectives in a transformer by learning multiple (W^Q, W^K, W^V) sets.

$$Q_i = XW_i^Q; K = XW_i^K; V = XW_i^V$$

$$\text{SelfAttention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_k}}\right) V_i$$

- With multihead attention, each attention head generates its own output vector i based on its own W_i^Q, W_i^K and W_i^V

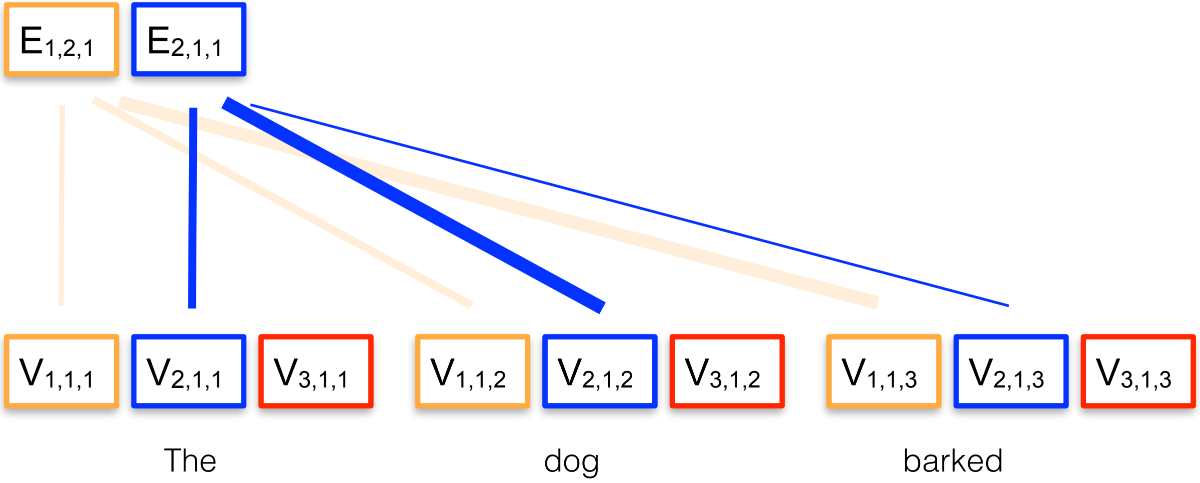


$E_{1,2,1}$

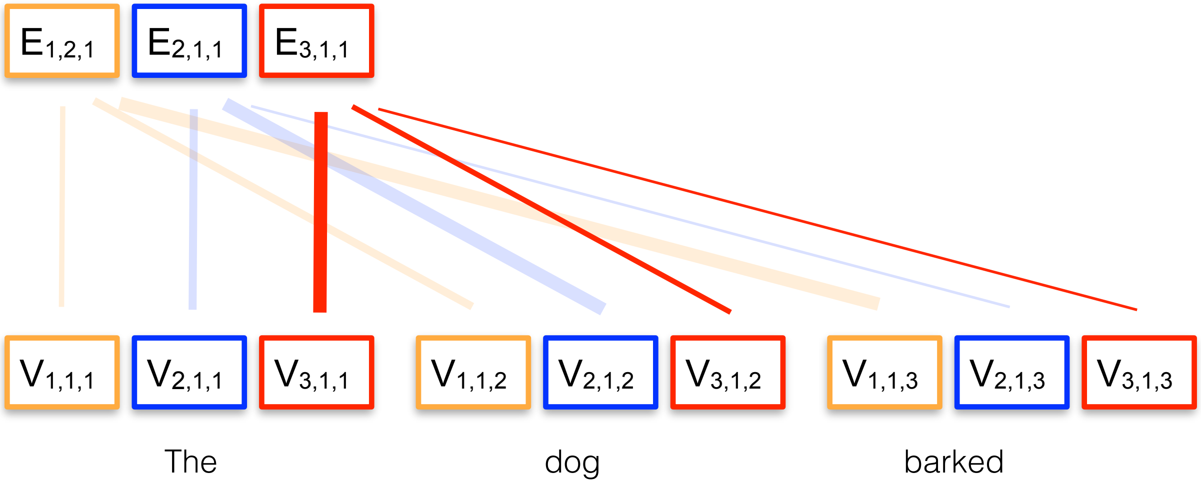
$V_{1,1,1}$ $V_{2,1,1}$ $V_{3,1,1}$ $V_{1,1,2}$ $V_{2,1,2}$ $V_{3,1,2}$ $V_{1,1,3}$ $V_{2,1,3}$ $V_{3,1,3}$

The dog barked

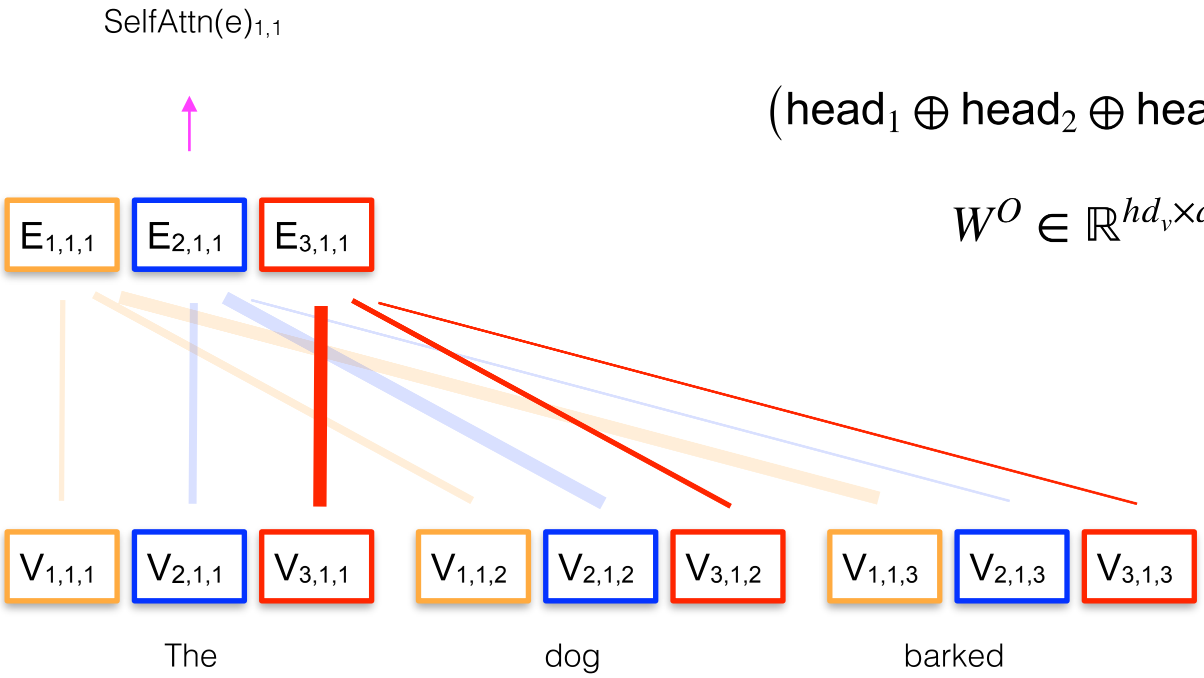
- With multihead attention, each attention head generates its own output vector i based on its own W_i^Q, W_i^K and W_i^V



- With multihead attention, each attention head generates its own output vector i based on its own W_i^Q, W_i^K and W_i^V

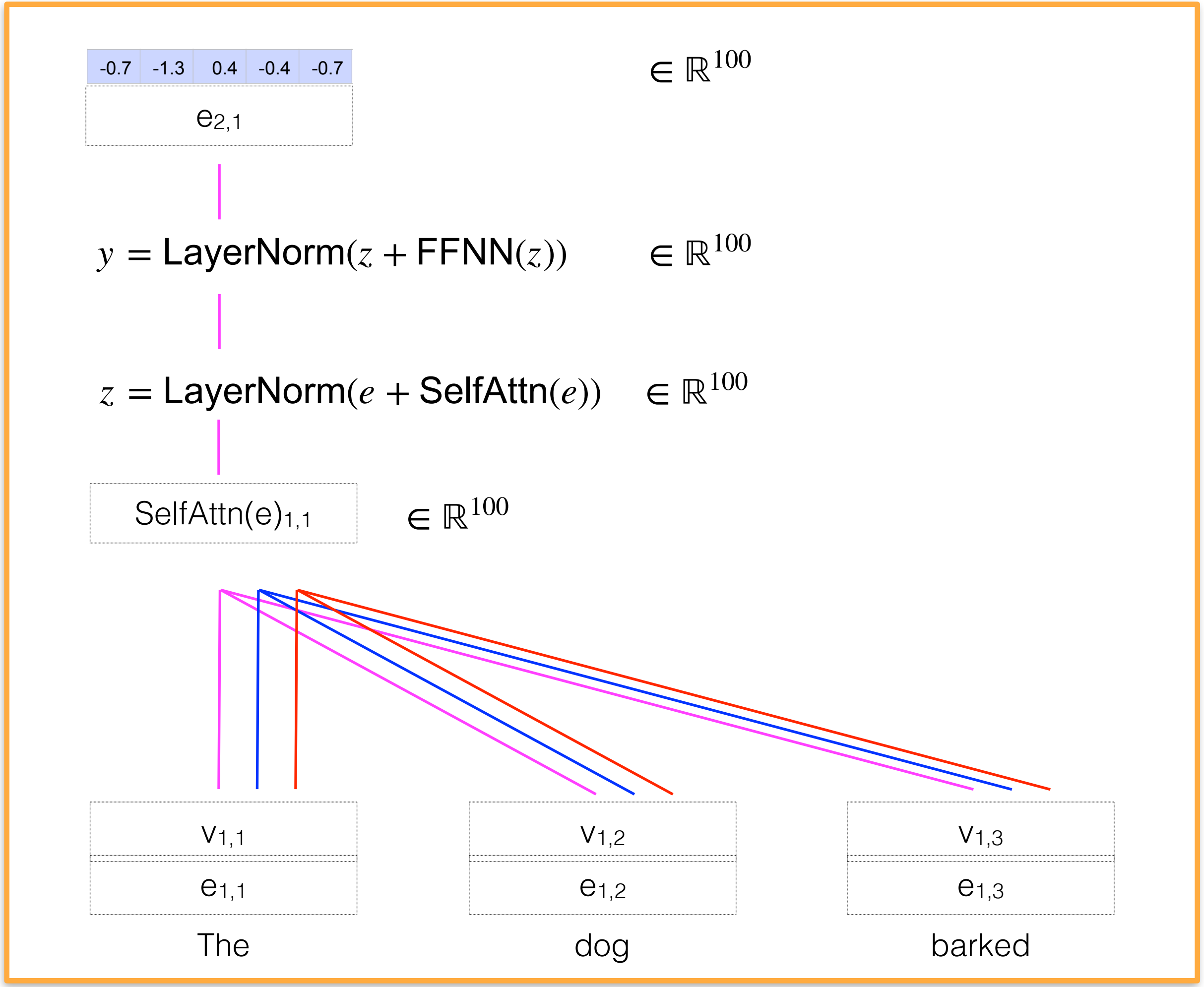


- These *h* separate output heads are then concatenated together and linearly transformed back to the original dimensionality d_v



$$\left(\mathbf{head}_1 \oplus \mathbf{head}_2 \oplus \mathbf{head}_3\right) W^O$$

$$W^O \in \mathbb{R}^{hd_v \times d_v}$$



-0.7	-1.3	0.4	-0.4	-0.7

$e_{2,1}$ $\in \mathbb{R}^{100}$

$$y = \text{LayerNorm}(z + \text{FFNN}(z)) \quad \in \mathbb{R}^{100}$$

$$z = \text{LayerNorm}(e + \text{SelfAttn}(e)) \quad \in \mathbb{R}^{100}$$

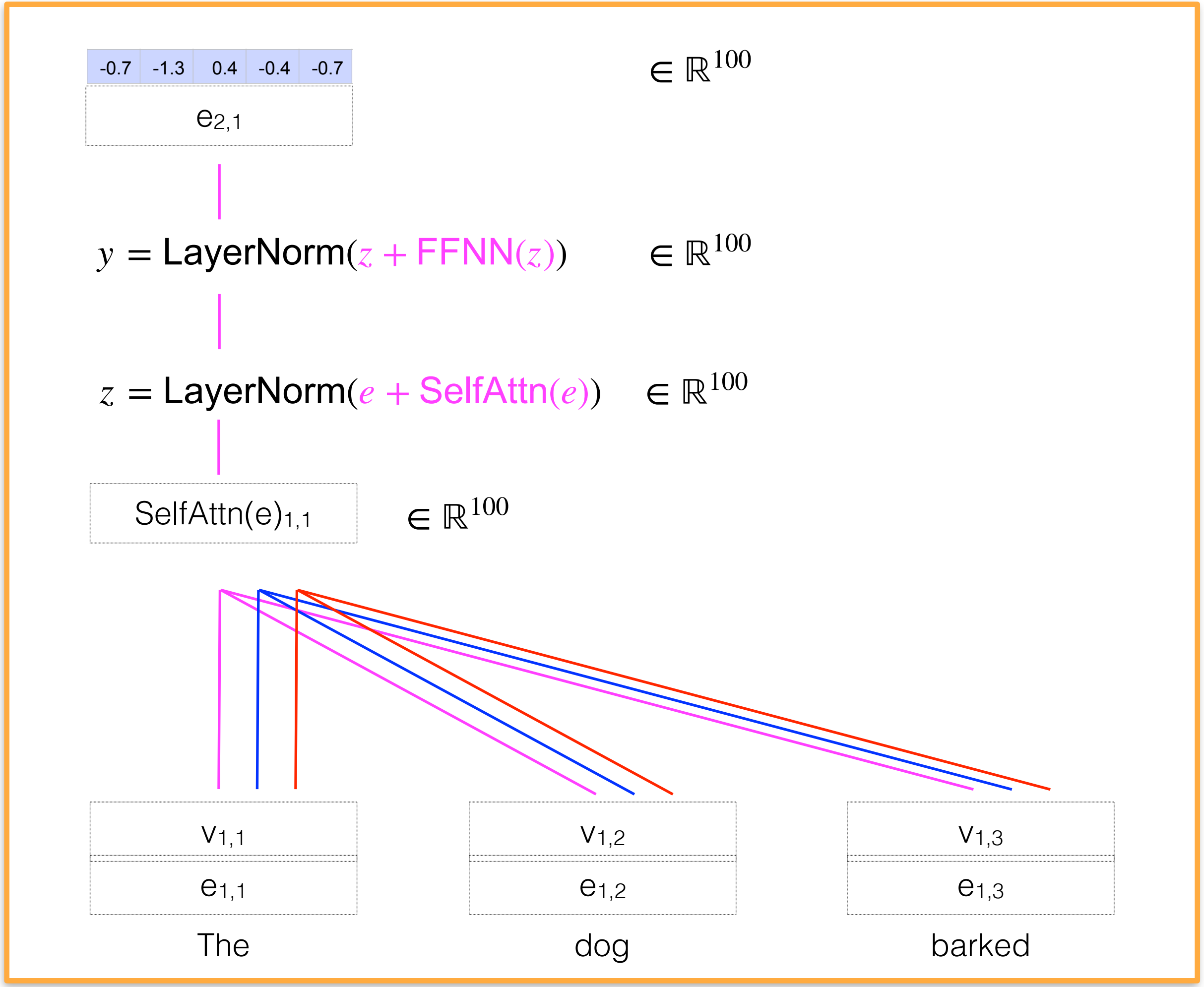
$\text{SelfAttn}(e)_{1,1}$ $\in \mathbb{R}^{100}$

$v_{1,1}$ $v_{1,2}$ $v_{1,3}$

$e_{1,1}$ $e_{1,2}$ $e_{1,3}$

The dog barked

- Residual layers add a layer's input to its output, giving later layers access to unmediated information.



-0.7	-1.3	0.4	-0.4	-0.7

$e_{2,1}$ $\in \mathbb{R}^{100}$

$$y = \text{LayerNorm}(z + \text{FFNN}(z)) \quad \in \mathbb{R}^{100}$$

$$z = \text{LayerNorm}(e + \text{SelfAttn}(e)) \quad \in \mathbb{R}^{100}$$

$\text{SelfAttn}(e)_{1,1}$ $\in \mathbb{R}^{100}$

$v_{1,1}$ $e_{1,1}$ The

$v_{1,2}$ $e_{1,2}$ dog

$v_{1,3}$ $e_{1,3}$ barked

Layer Normalization

- Transform each output from a layer d_h into its z-score, with two learnable parameters ɣ and β.

$$\mu = \frac{1}{d_h}\sum_{i=1}^{d_h} x_i \qquad \sigma = \sqrt{\frac{1}{d_h}\sum_{i=1}^{d_h}(x_i - \mu)^2}$$

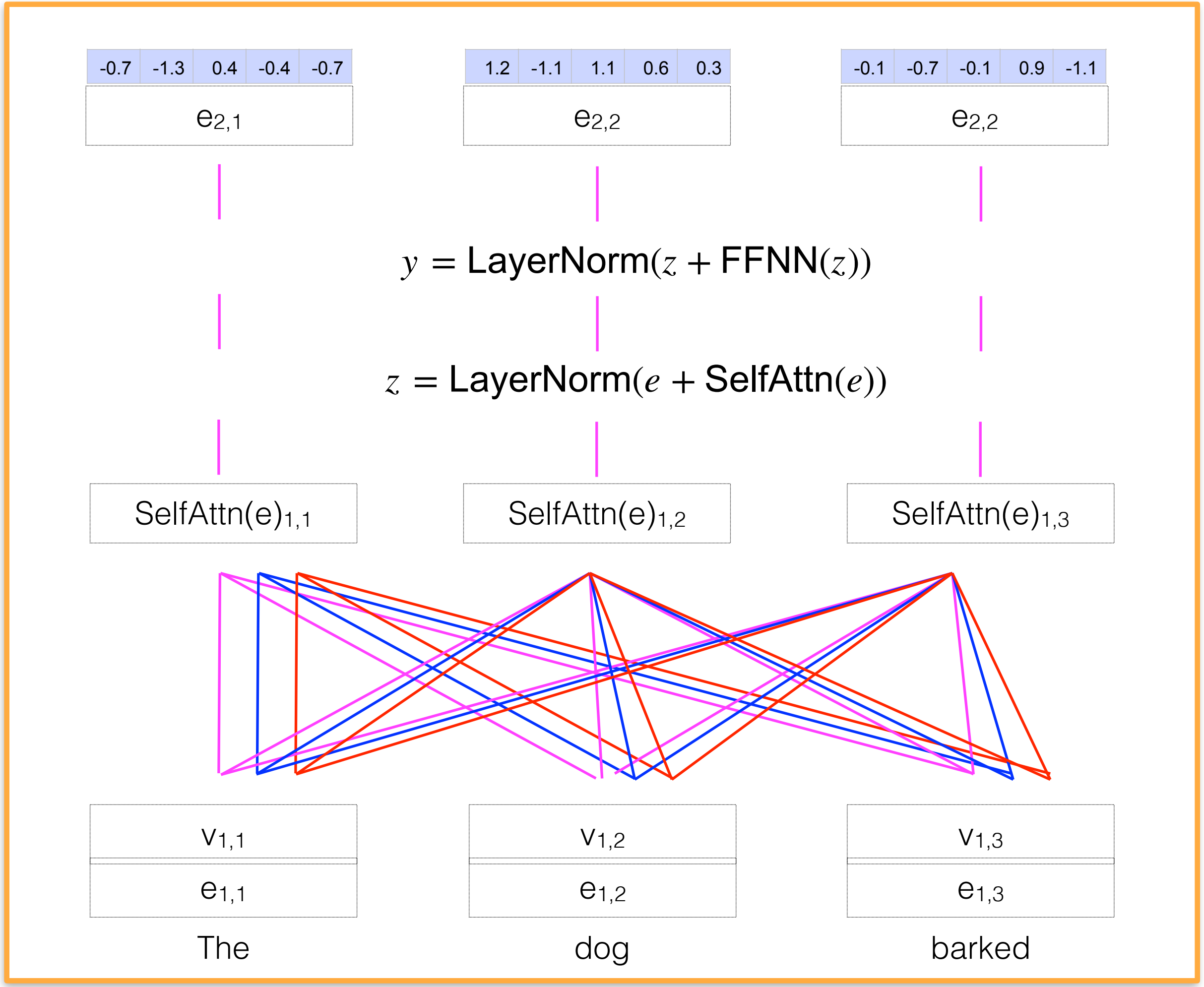
$$\hat{x} = \frac{x - \mu}{\sigma}$$

$$\text{LayerNorm} = \gamma\hat{x} + \beta$$

Output

This whole process defines one attention block. The input is a sequence of (e.g. 100-dimensional) vectors; the output of each block is a sequence of (100-dimensional) vectors.

Input

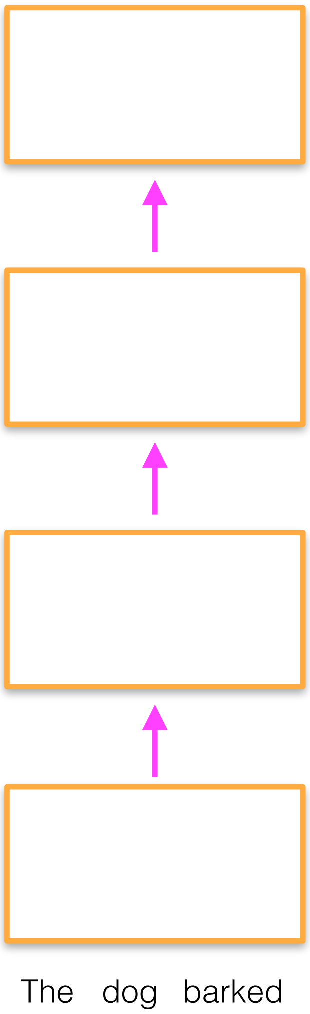


$$y = \text{LayerNorm}(z + \text{FFNN}(z))$$

$$z = \text{LayerNorm}(e + \text{SelfAttn}(e))$$

This whole process defines one attention block. The input is a sequence of (e.g. 100-dimensional) vectors; the output of each block is a sequence of (100-dimensional) vectors.

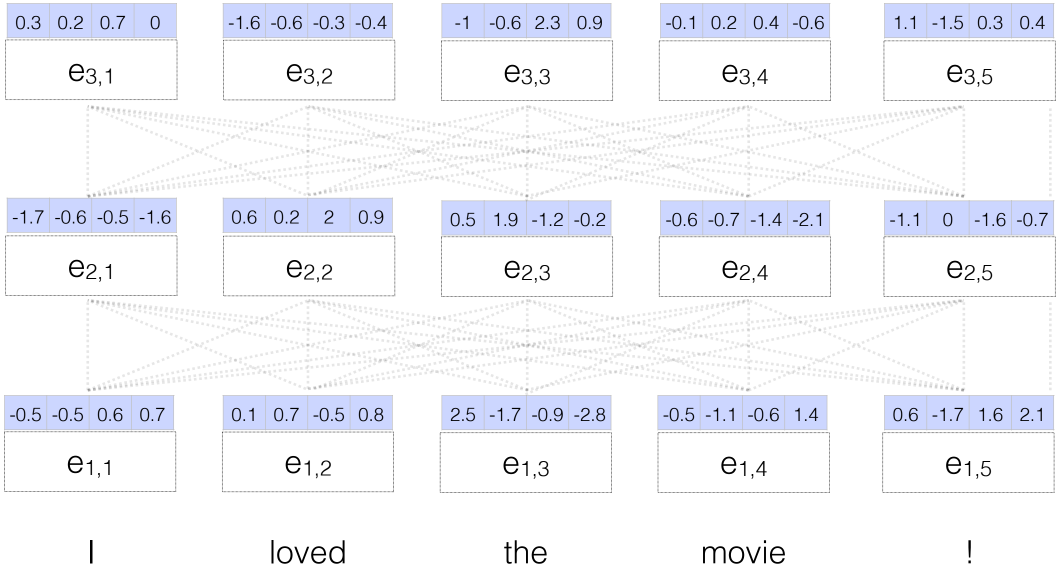
Transformers can stack many such blocks; where the output from block b is the input to block b+1.

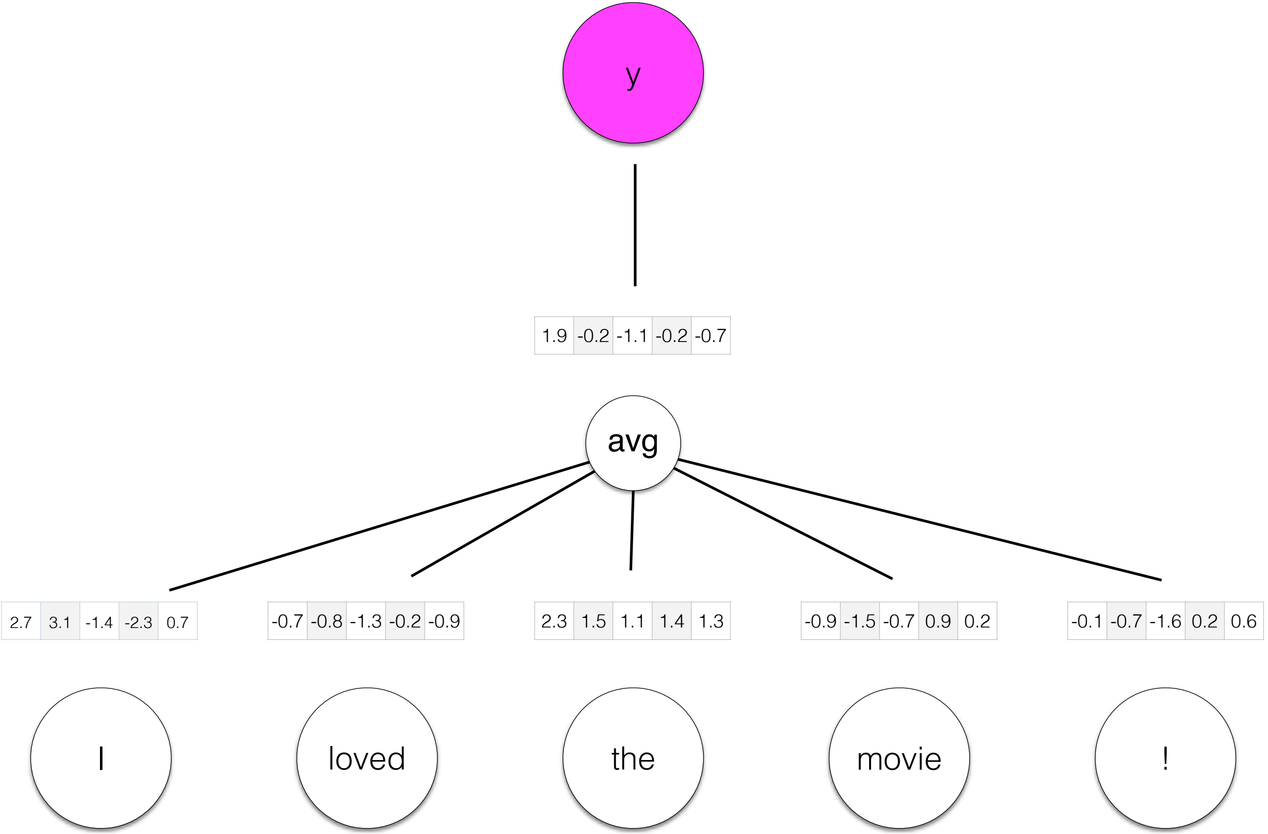


BERT

- *Bidirectional Encoder Representations from Transformers* (Delvin et al 2019)
- Deep layers (12 for BERT base, 24 for BERT large)
- Large representation sizes (768 per layer)
- Pretrained on English Wikipedia (2.5B words) and BooksCorpus (800M words) — we'll cover this on 2/12 when we study contextual language models.

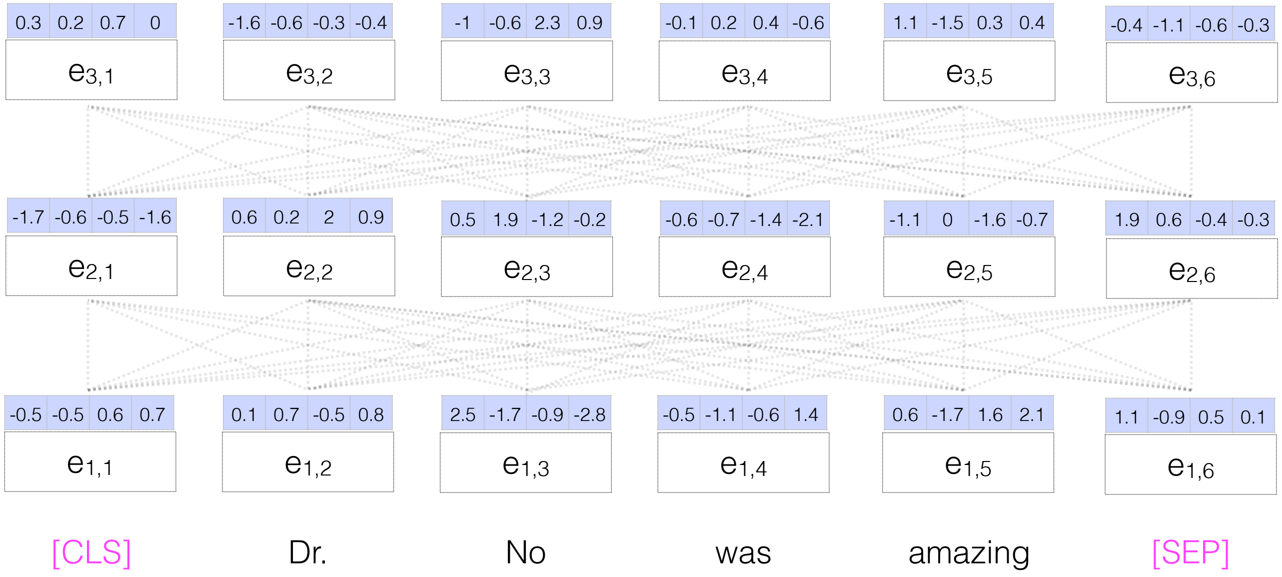
- At this point, we have one representation for each token for each layer in our transformer. How do we use this for document classification?



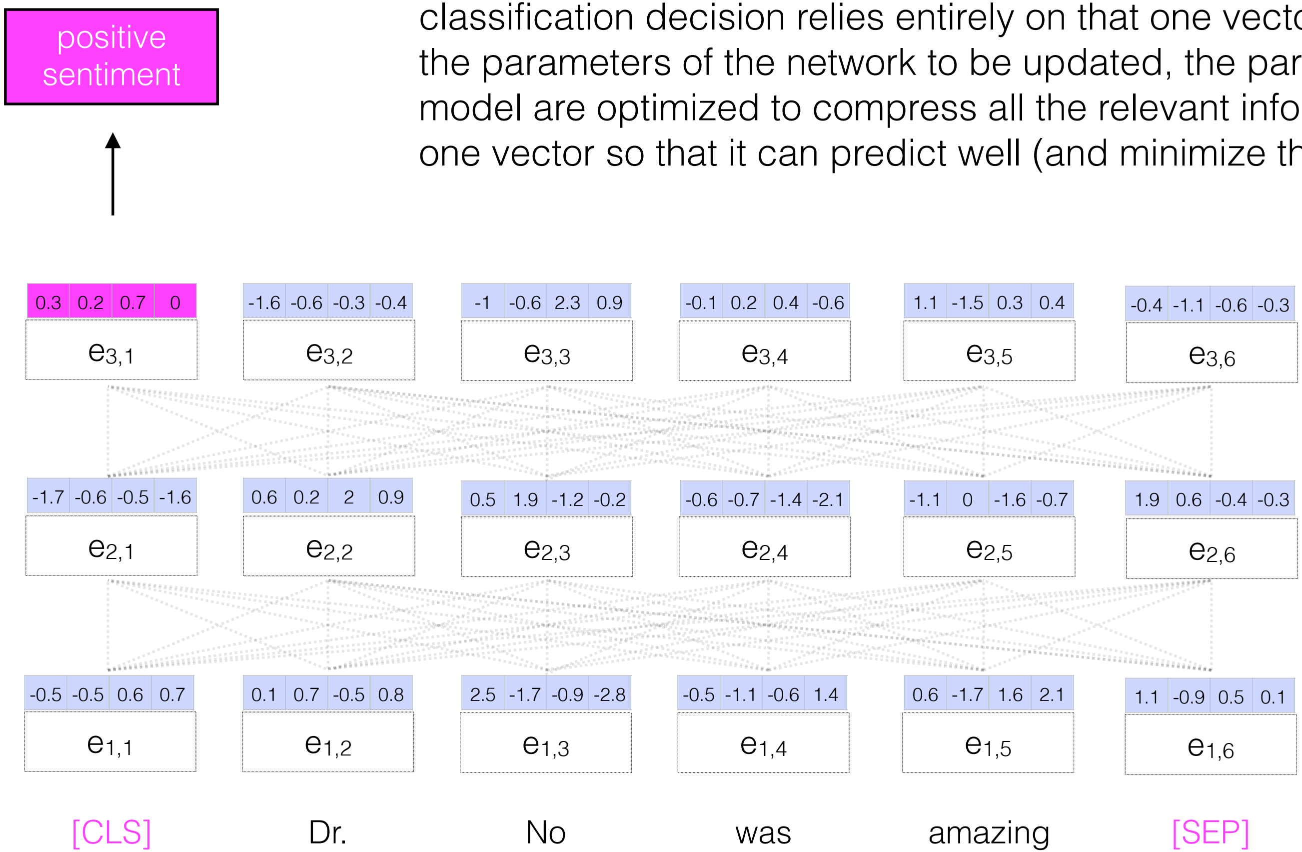


Iyyer et al. (2015), “Deep Unordered Composition Rivals Syntactic Methods for Text Classification” (ACL)

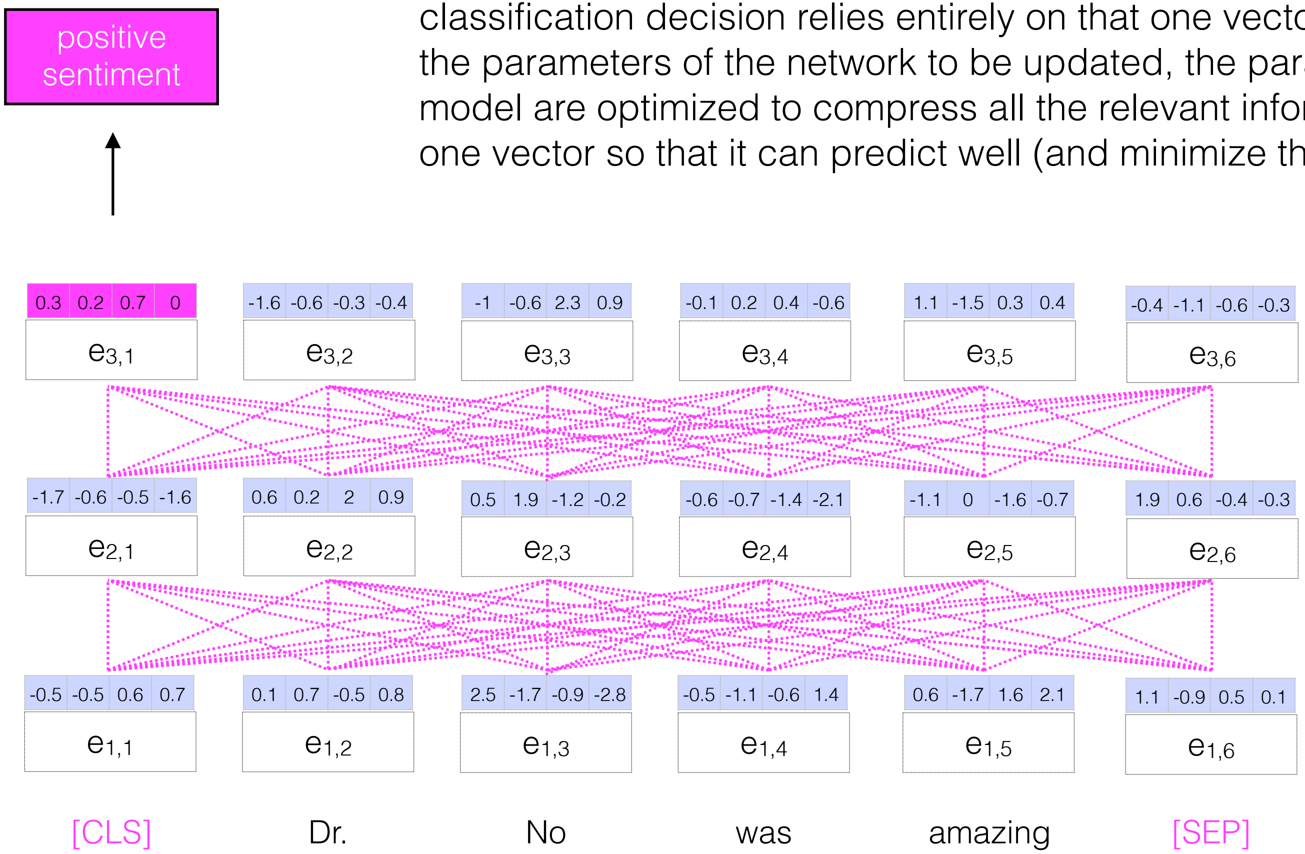
- BERT also encodes each sentence by appending a special token to the beginning ([CLS]) and end ([SEP]) of each sequence.
- This helps provides a single token that can be optimized to represent the entire sequence (e.g., for document classification)



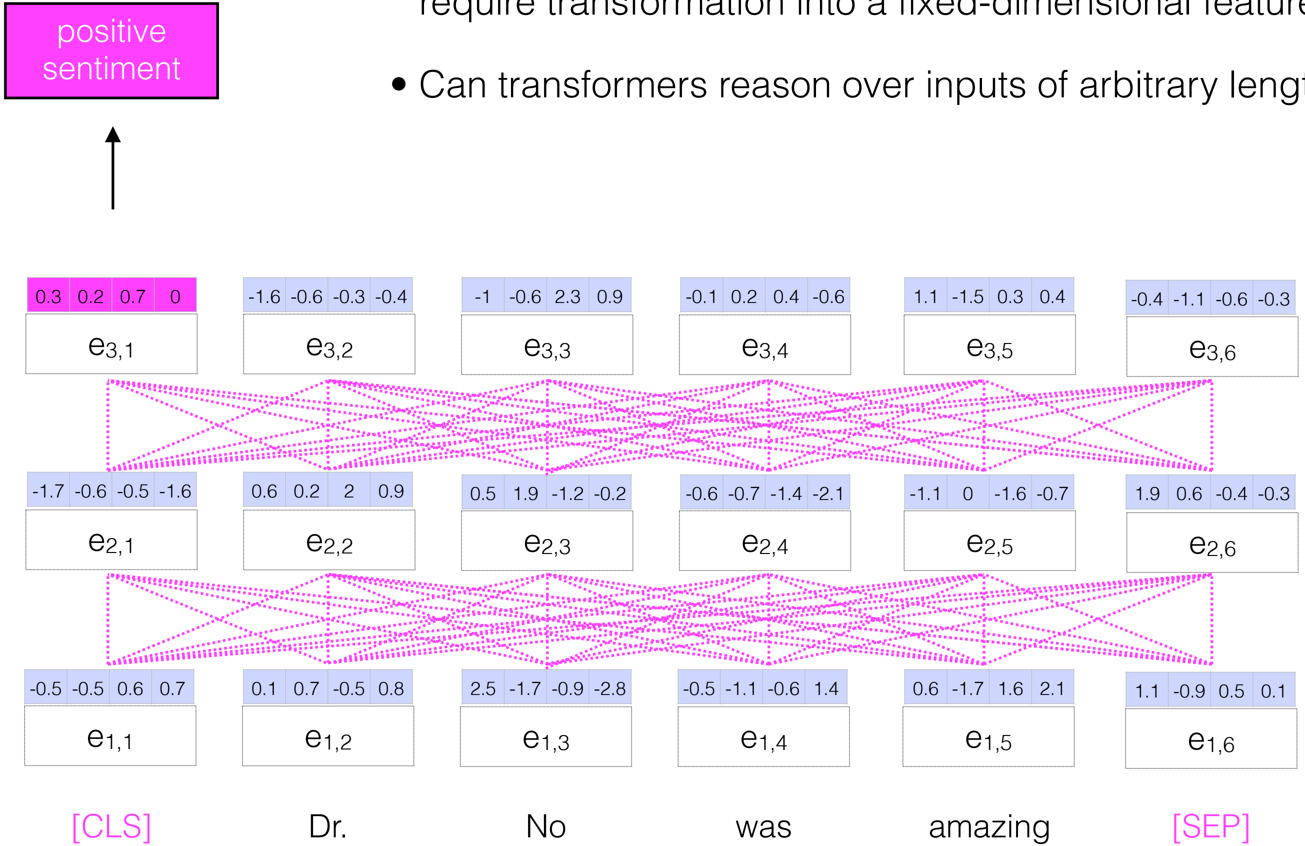
- We can represent the entire document with this *one* [CLS] vector
- Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).



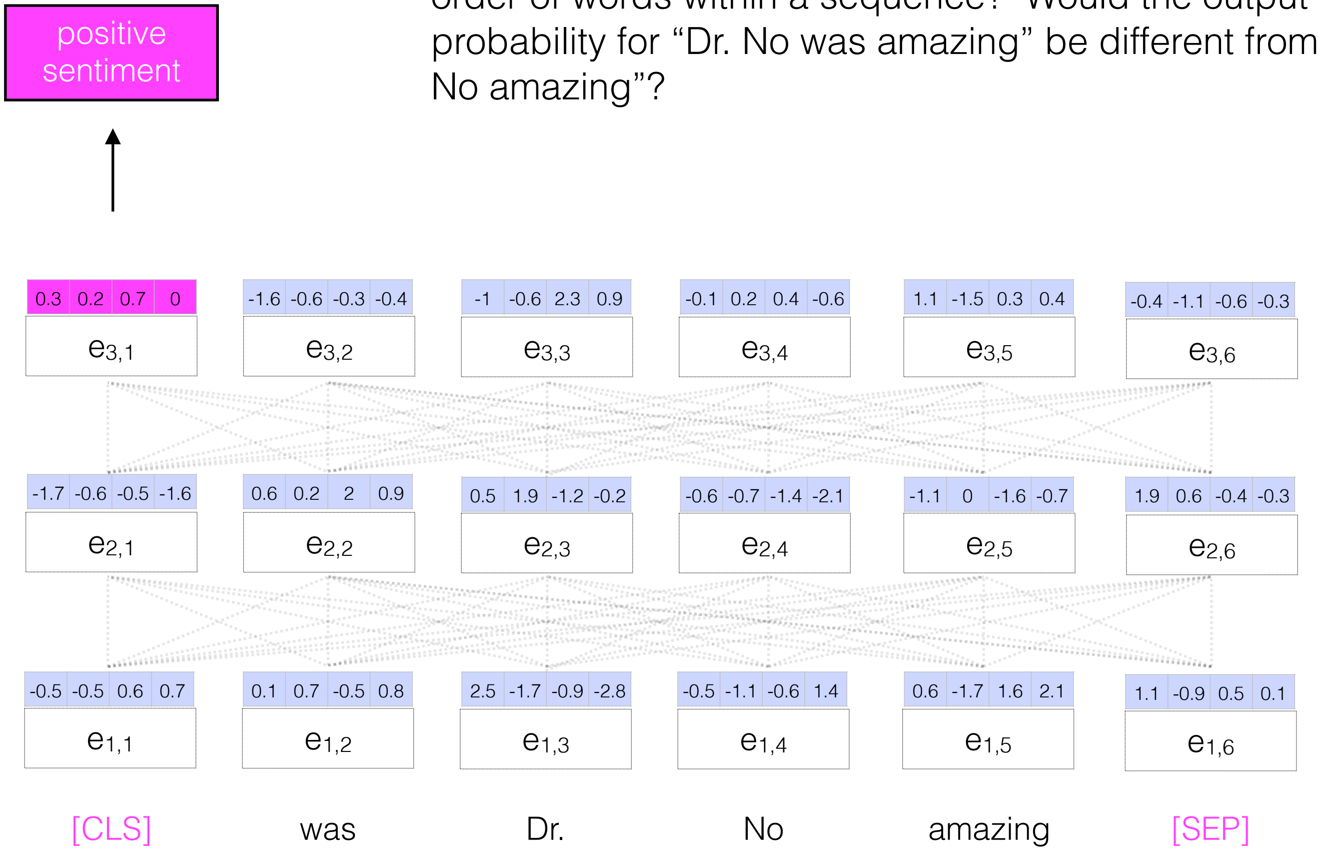
- We can represent the entire document with this *one* [CLS] vector
- Why does this work? When we design our network so that a classification decision relies entirely on that one vector and allow all the parameters of the network to be updated, the parameters of the model are optimized to compress all the relevant information into that one vector so that it can predict well (and minimize the loss).



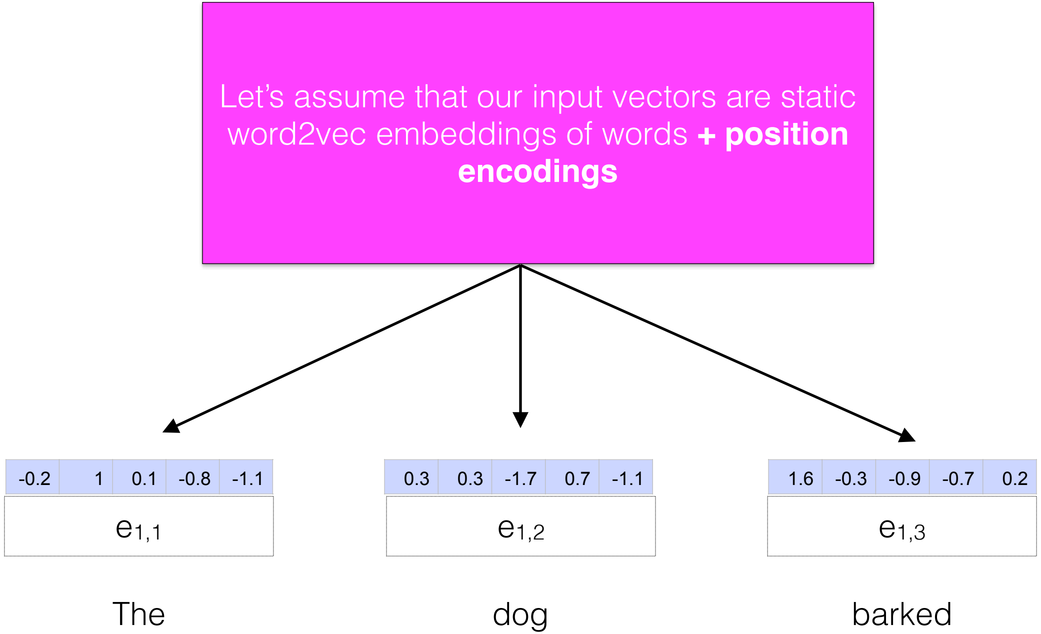
- CNNs can reason over text inputs of arbitrary length; FFNN require transformation into a fixed-dimensional feature vector.
- Can transformers reason over inputs of arbitrary length?



- Does a transformer encode any intrinsic information about the order of words within a sequence? Would the output probability for “Dr. No was amazing” be different from “was Dr. No amazing”?



Position encoding



Let's assume that our input vectors are static word2vec embeddings of words **+ position encodings**

-0.2	1	0.1	-0.8	-1.1

$e_{1,1}$

The

0.3	0.3	-1.7	0.7	-1.1

$e_{1,2}$

dog

1.6	-0.3	-0.9	-0.7	0.2

$e_{1,3}$

barked

Position embeddings

One option is to add learnable position embeddings pe[i] to each word embedding *e* at position *i* (or concatenate them)

We can add two vectors if they're the same dimensionality

$$e_i = e_i + \text{pe}[i]$$

Or concatenate them if not

$$e_i = e_i \oplus \text{pe}[i]$$

0	2	-0.5	1.1	0.3	0.4	-0.5
1	-1.4	0.4	-0.2	-0.9	0.5	0.9
2	-1.1	-0.2	-0.5	0.2	-0.8	0
3	0.7	-0.3	1.5	-0.3	-0.4	0.1
4	-0.8	1.2	1	-0.7	-1	-0.4
5	0	0.3	-0.3	-0.9	0.2	1.4
6	0.8	0.8	-0.4	-1.4	1.2	-0.9
7	1.6	0.4	-1.1	0.7	0.1	1.6
…	…	…	…	…	…	…

position embeddings (pe)

Position encodings

Vaswani et al. 2017 use sinusoidal functions to deterministically create a vector of position encodings the same dimensionality as the input (d_{model}); the embedding dimensions progress from wavelength of 2π to 10000* 2π.

i = embedding dimension

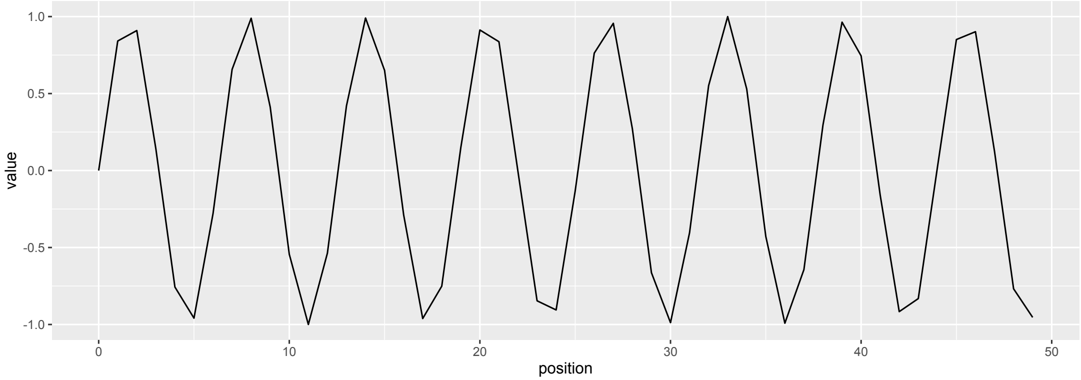
pos = position in sequence

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

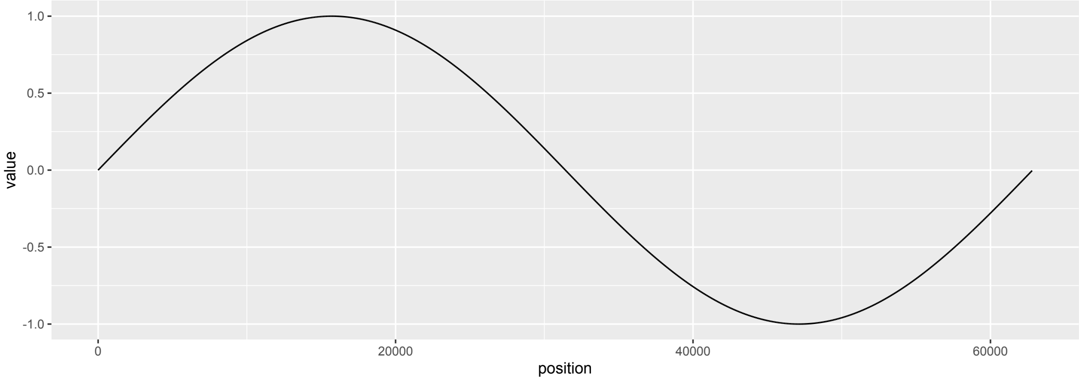
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Position encodings

Sine(x) values for integers x from positions 0 to 50, embedding index = 0 (d_{model} = 100)



Sine(x) values for integers x from positions 0 to 62,800, embedding index = 49 (d_{model} = 100)



Position encodings

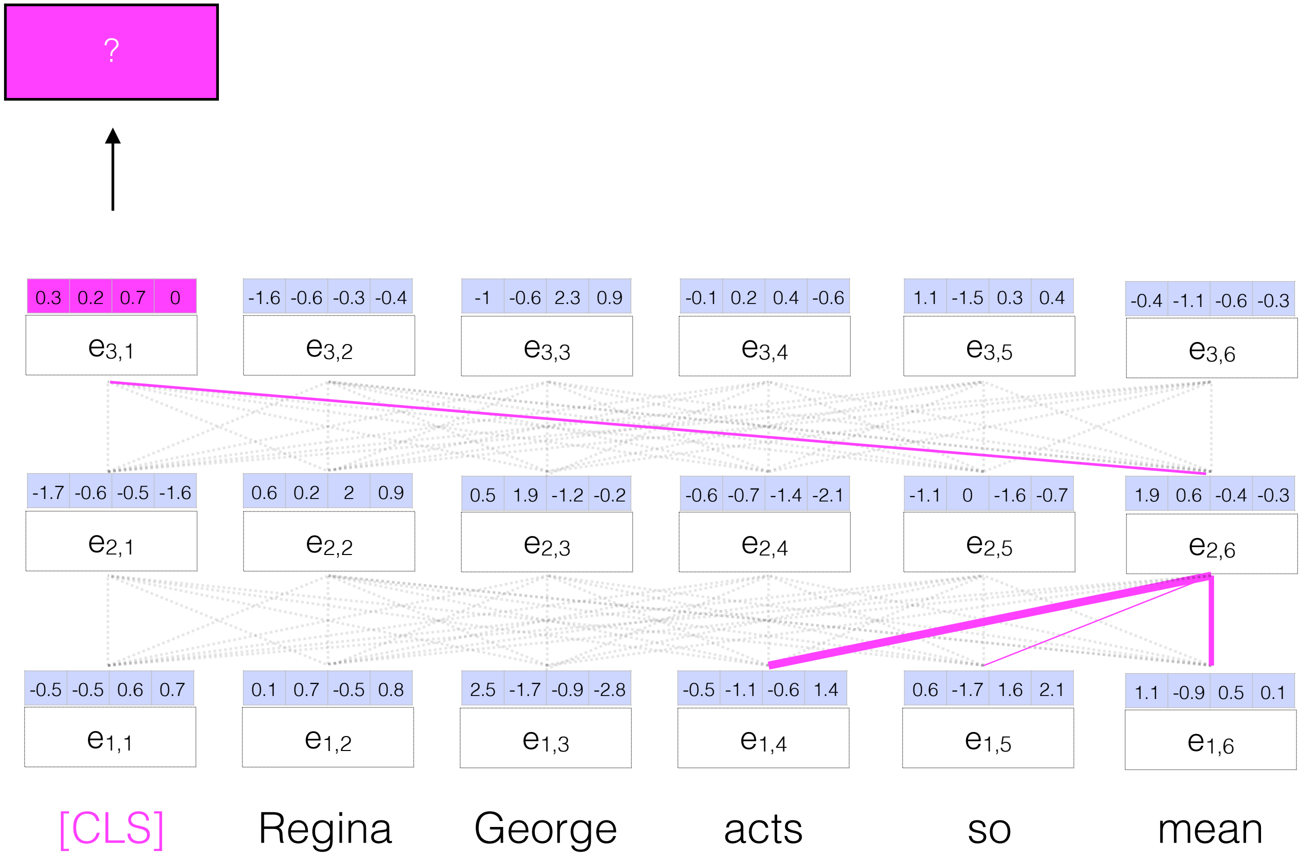
We can see that the position embedding for word at position 20 is much more similar to the embedding for word at position 21 (closer) than that at position 98 (much further away).

position in sequence

dimension

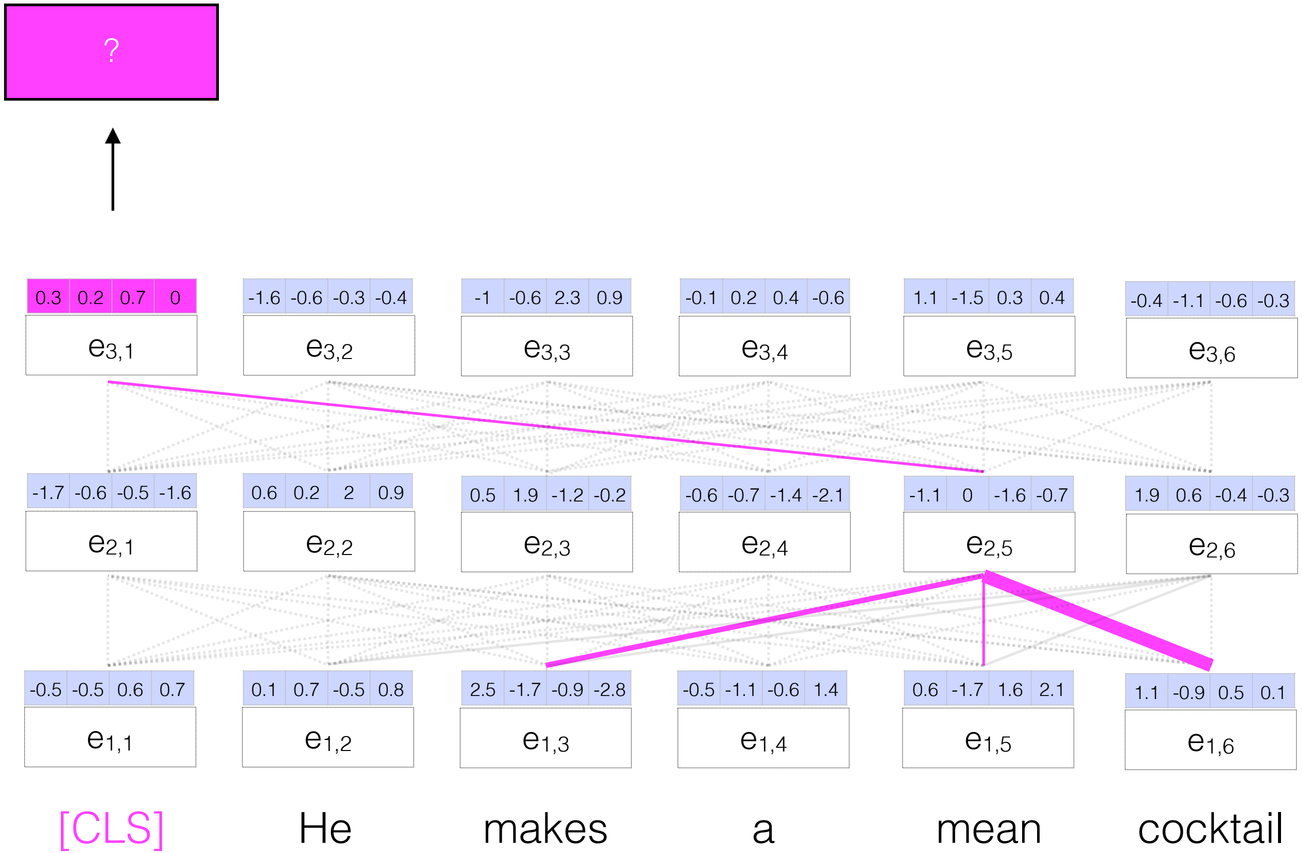
	20	21	98
0	0.91	0.84	-0.57
10	-0.11	-0.49	0.25
20	-0.03	-0.19	0.18
30	0.30	0.24	1.00
40	0.48	0.50	0.63
50	0.98	0.98	0.56
60	0.08	0.08	0.38
70	1.00	1.00	0.99
80	0.01	0.01	0.06
90	1.00	1.00	1.00

What does “mean” mean?



?

0.3 0.2 0.7 0	-1.6 -0.6 -0.3 -0.4	-1 -0.6 2.3 0.9	-0.1 0.2 0.4 -0.6	1.1 -1.5 0.3 0.4	-0.4 -1.1 -0.6 -0.3
$e_{3,1}$	$e_{3,2}$	$e_{3,3}$	$e_{3,4}$	$e_{3,5}$	$e_{3,6}$
-1.7 -0.6 -0.5 -1.6	0.6 0.2 2 0.9	0.5 1.9 -1.2 -0.2	-0.6 -0.7 -1.4 -2.1	-1.1 0 -1.6 -0.7	1.9 0.6 -0.4 -0.3
$e_{2,1}$	$e_{2,2}$	$e_{2,3}$	$e_{2,4}$	$e_{2,5}$	$e_{2,6}$
-0.5 -0.5 0.6 0.7	0.1 0.7 -0.5 0.8	2.5 -1.7 -0.9 -2.8	-0.5 -1.1 -0.6 1.4	0.6 -1.7 1.6 2.1	1.1 -0.9 0.5 0.1
$e_{1,1}$	$e_{1,2}$	$e_{1,3}$	$e_{1,4}$	$e_{1,5}$	$e_{1,6}$
[CLS]	Regina	George	acts	so	mean



?

0.3 0.2 0.7 0
$e_{3,1}$

-1.6 -0.6 -0.3 -0.4
$e_{3,2}$

-1 -0.6 2.3 0.9
$e_{3,3}$

-0.1 0.2 0.4 -0.6
$e_{3,4}$

1.1 -1.5 0.3 0.4
$e_{3,5}$

-0.4 -1.1 -0.6 -0.3
$e_{3,6}$

-1.7 -0.6 -0.5 -1.6
$e_{2,1}$

0.6 0.2 2 0.9
$e_{2,2}$

0.5 1.9 -1.2 -0.2
$e_{2,3}$

-0.6 -0.7 -1.4 -2.1
$e_{2,4}$

-1.1 0 -1.6 -0.7
$e_{2,5}$

1.9 0.6 -0.4 -0.3
$e_{2,6}$

-0.5 -0.5 0.6 0.7
$e_{1,1}$

0.1 0.7 -0.5 0.8
$e_{1,2}$

2.5 -1.7 -0.9 -2.8
$e_{1,3}$

-0.5 -1.1 -0.6 1.4
$e_{1,4}$

0.6 -1.7 1.6 2.1
$e_{1,5}$

1.1 -0.9 0.5 0.1
$e_{1,6}$

[CLS] He makes a mean cocktail

Transformers

- Transformers have been extremely influential in NLP (Vaswani et al. 2017 has 35K citations!)

- We'll see them much more in this class in the context of specific applications:

    - Contextual language models, including causal self-attention (GPT), and bidirectional attention (BERT).

    - Machine translation

    - Text generation

Logistics

- Quiz 2 will be out this Friday (due next Monday Feb 5).
- Homework 1 is out & due next Tuesday, Feb 6 (11:59 pm)
    - Homework 2 will be out early next week.

- Next time:
    - Annotation