Contextual Embeddings

Contextual Embeddings

Reminder: Static word embeddings (GLoVe or word2vec)

Meaning defined as a point in space based on distribution

- Each word = one fixed (static) vector

Similar words are "nearby in semantic space"

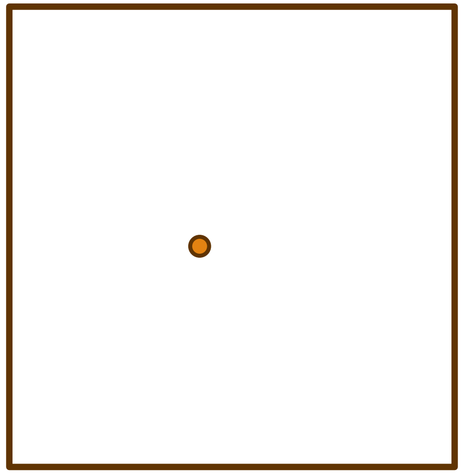
- Learned by seeing which words are **nearby in text**

Static embeddings

Each word is represented by a fixed vector (same in all contexts)

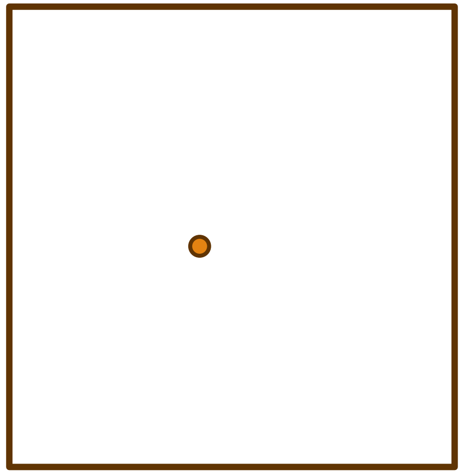
We had a picnic on the grassy river **bank**

Bank = [35, -1.8, 22, 0.006,...]



I went to the **bank** and withdrew some cash.

Bank = [35, -1.8, 22, 0.006,...]



In static embeddings, we get a fixed dictionary

So we get our embeddings from

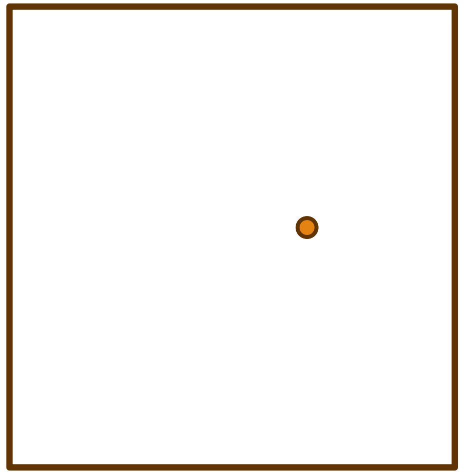
- a Dict
- that maps each string ("bank")
- to an np.array of length 50

Contextual embeddings

Each word **in context** is represented by a vector (different in every context)

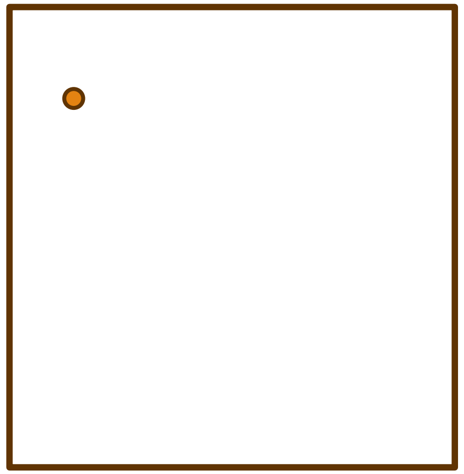
We had a picnic on the grassy river **bank**

Bank = [1.11, -1.7, -205, 0.006,...]



I went to the **bank** and withdrew some cash.

Bank = [-42.7, 9.8, -0.88, -2.559,...]



Contextual embeddings

Each word **in context** is represented by a vector

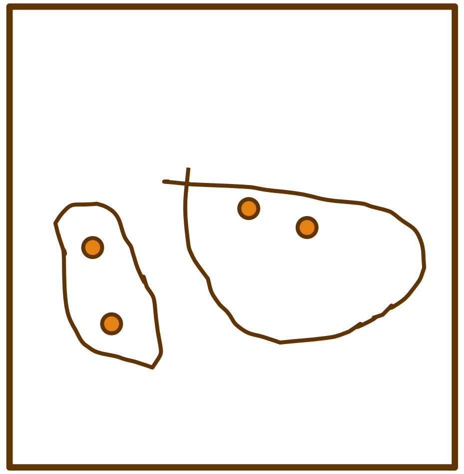
1 point for each sentence!

We had a picnic on the grassy river **bank**

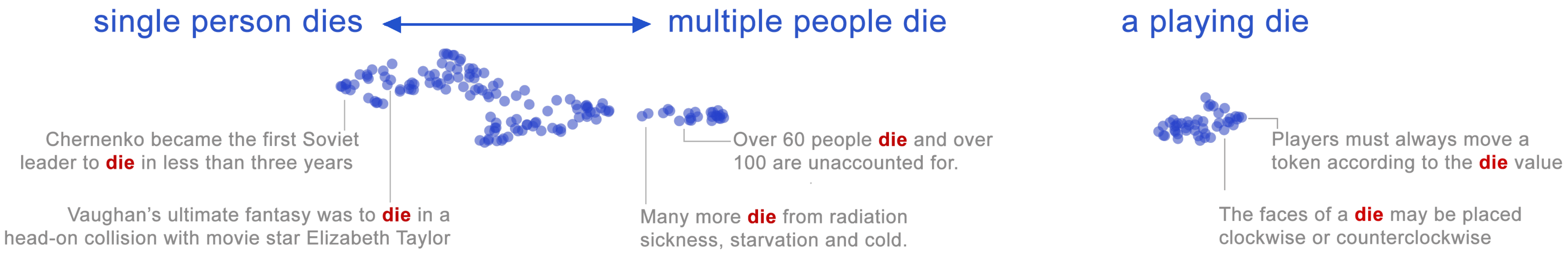
I went to the **bank** and withdrew some cash.

My friend works at the local branch of the bank

I sat on the damp bank and watched the river



Contextual Embedding for "die"



Most modern models are multilingual!
Contextual Embedding for "die" including German!

Word sense

Words are ambiguous

A **word sense** is a discrete representation of one aspect of meaning

mouse^1 : .... a *mouse* controlling a computer system in 1968.
mouse^2 : .... a quiet animal like a *mouse*
bank^1 : ...a *bank* can hold the investments in a custodial account ...
bank^2 : ...as agriculture burgeons on the east *bank*, the river ...

Contextual embeddings offer a continuous high-dimensional model of meaning that is more fine grained than discrete senses.

Summary: Static vs Contextual Embeddings

Static embeddings represent **word types** (dictionary entries)

- Look up a word in a Dict
- Returns a vector of shape (50,)

Contextual embeddings represent **word instances** (one for each time the word occurs in any context/sentence)

- Pass a sentence through a language model
- Get out one vector of shape (768,) for each word in the sentence

How do we compute contextual embeddings?

From internals of large language model!

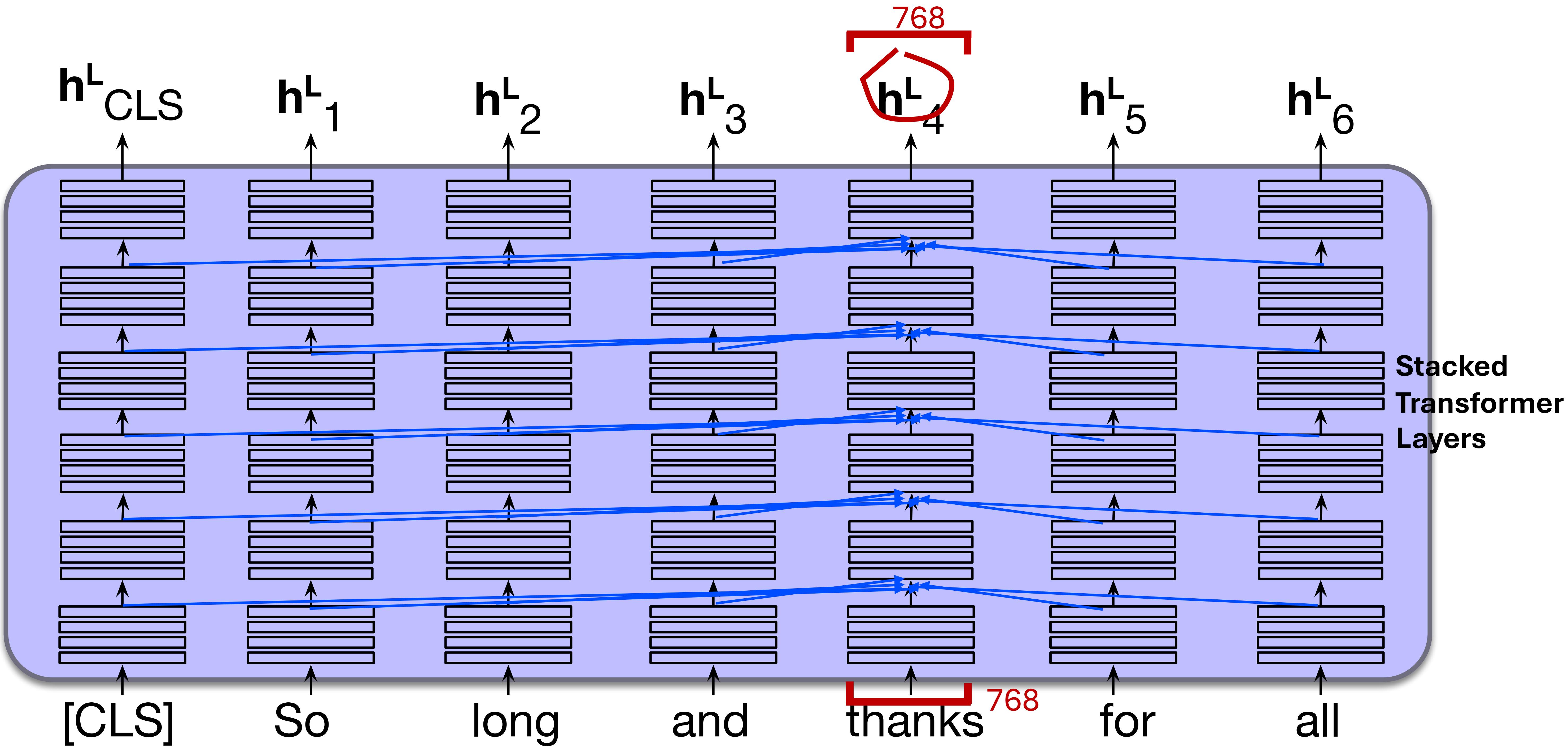
The transformer networks (next week's lecture) will have representations for each word at many different layers

We'll be using BERT, which is a Masked Language Model based on the transformer architecture

More of those details next week!

BERT contextual embeddings to represent words

768-dimensional embedding for "thanks" in "So long and thanks for all"

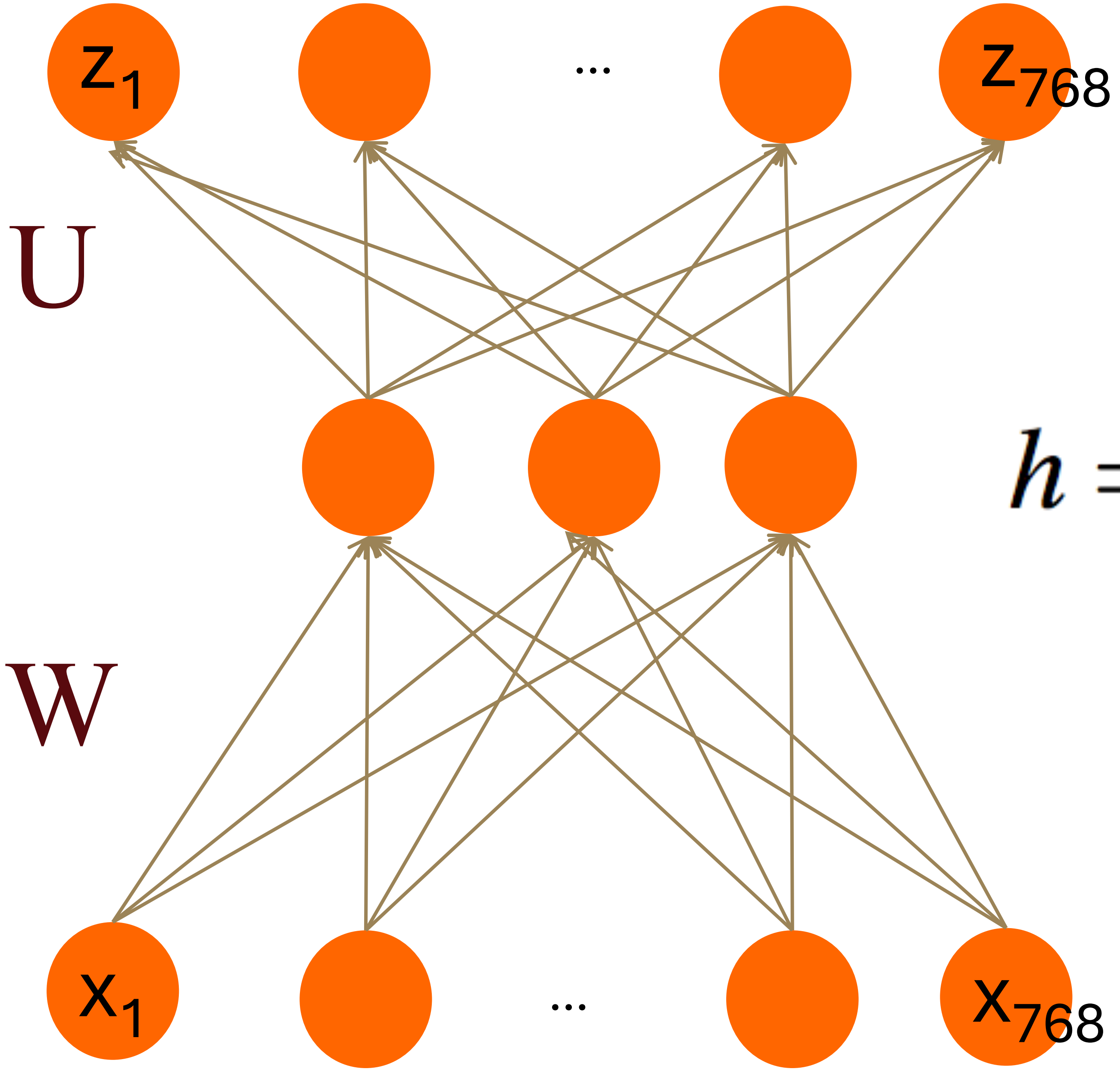


The stream of information in a feedforward neural network

Output layer
(vector d=768)

hidden units

Input layer
(vector d=768)



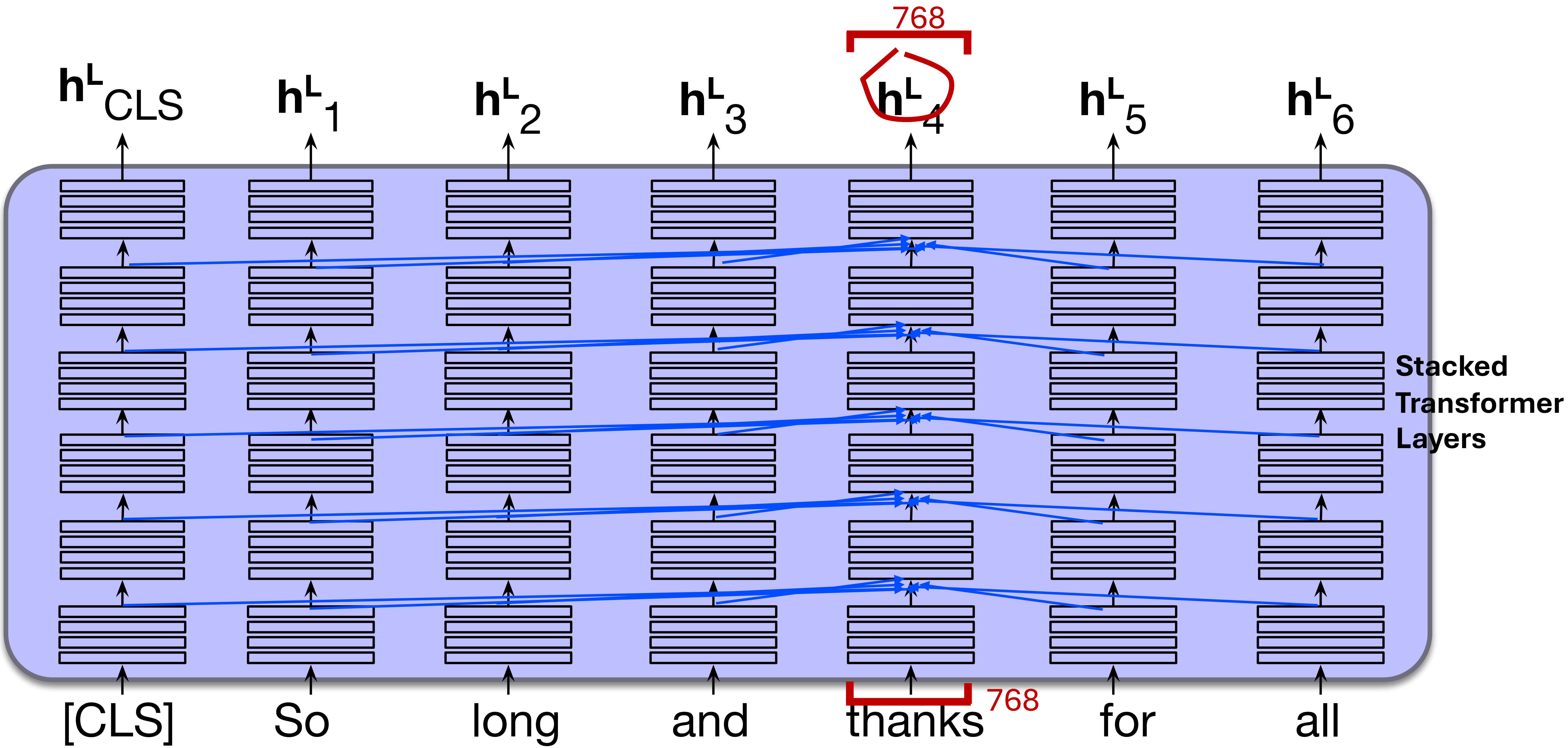
$$z = \sigma\,(Uh)$$

$$h = \boldsymbol{\sigma}(Wx + b)$$

σ Could be ReLU
Or tanh

BERT contextual embeddings to represent words

768-dimensional embedding for "thanks" in "So long and thanks for all"



Contextual Embeddings

Contextual Embeddings

Embeddings

Computing word and sentence similarity with contextual embeddings

Computing word similarity: Reminder about dot product and cosine

Dot product between two vectors is a scalar:

$$\begin{bmatrix} 92 \\ 0 \\ 40 \end{bmatrix} \cdot \begin{bmatrix} 35 \\ 2 \\ 85 \end{bmatrix} = 92*35 + 0*2 + 40+85 = 3345$$

Dot product is high when both vectors have large values in same dimensions

Reminder: cosine fixes problem with raw dot-product

Dot product is higher if a vector is longer, higher magnitude (has higher values in many dimension)

Vector length:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Cosine normalizes for vector length:

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Reminder: Computing similarity between two words

1. Look up their static embedding vectors
2. Take the cosine between them

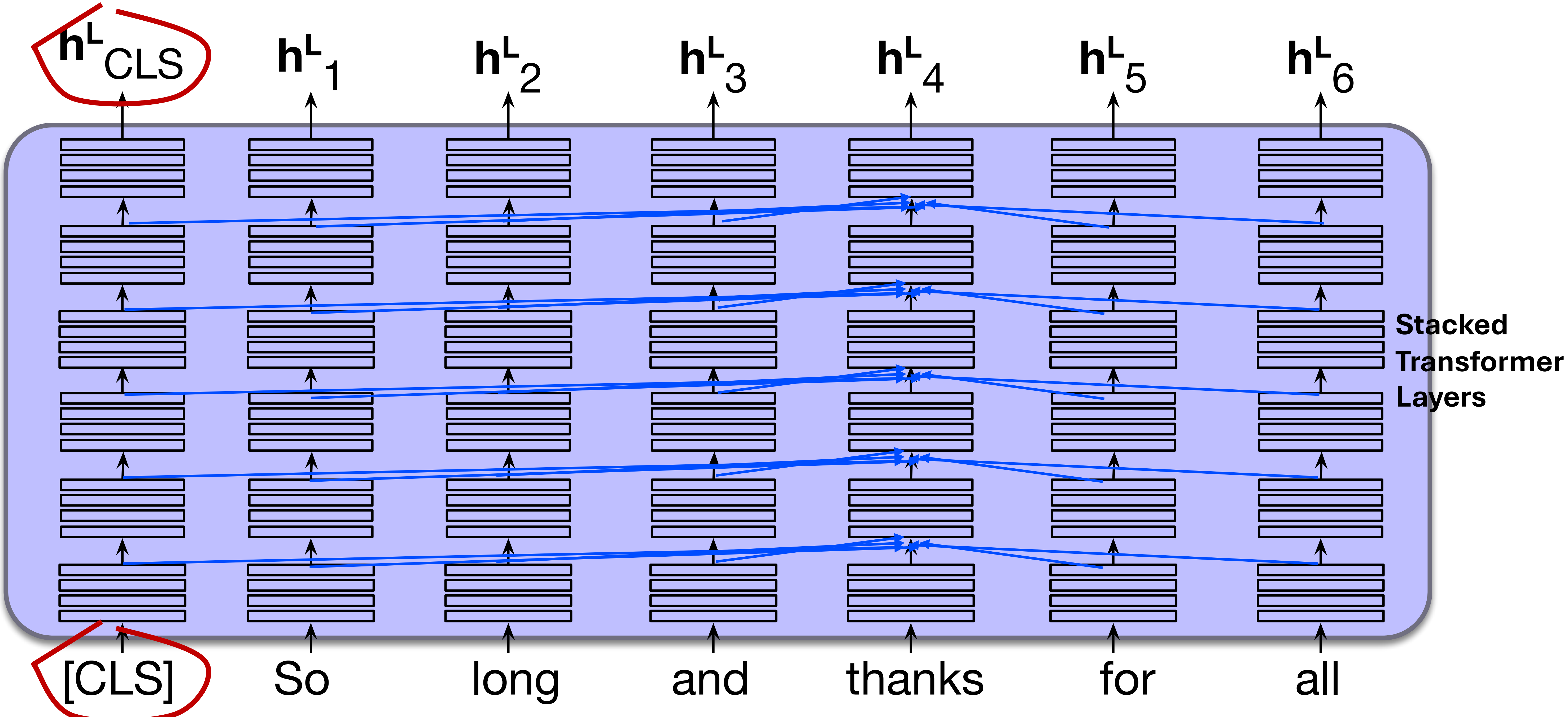
But how to take similarity of two sentences?

Two methods:

1. **CLS token**: a special token that represents the whole sentence
2. **Mean-pooling**: Average the embeddings of all the words!

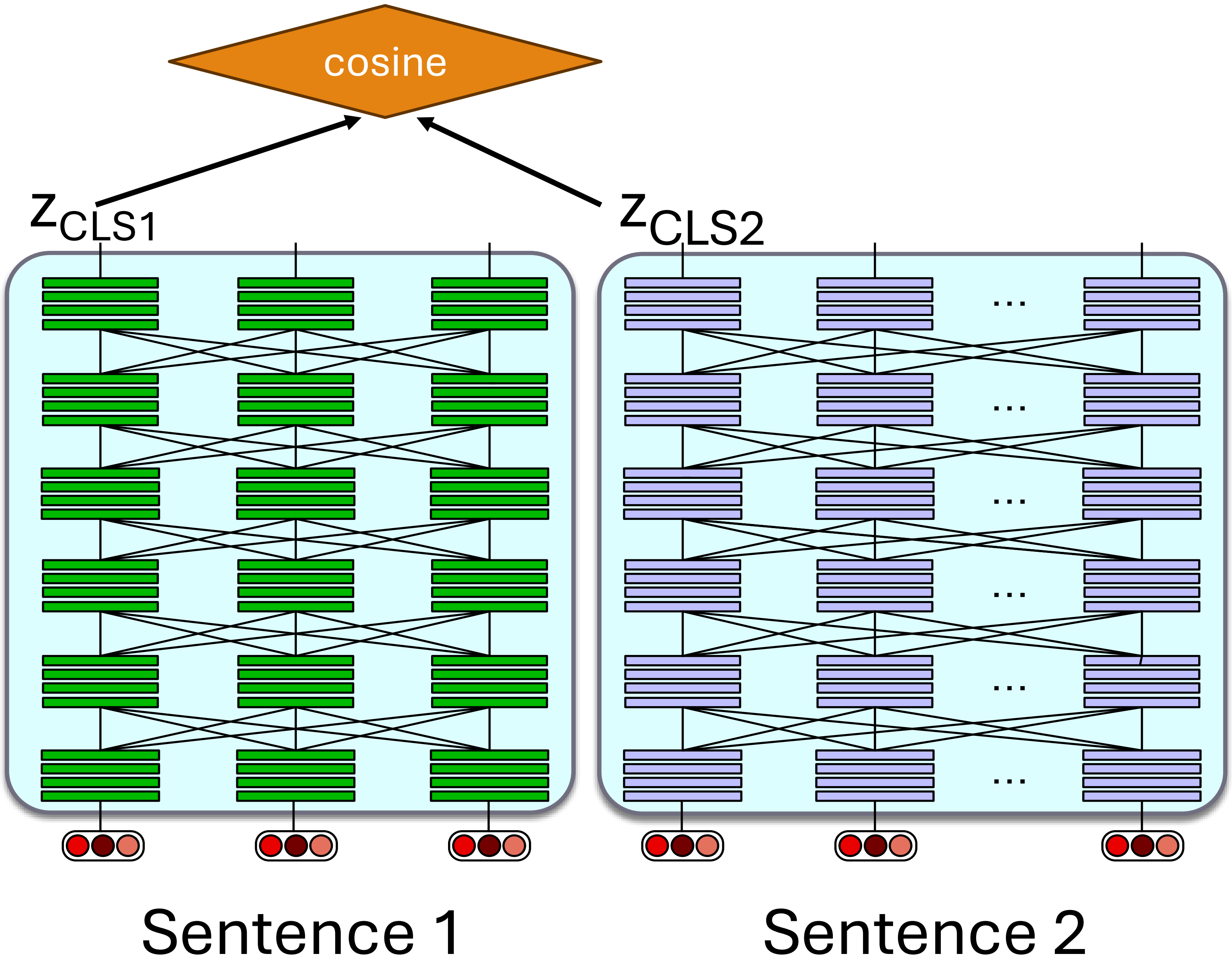
BERT contextual embeddings to represent words

CLS is a special token that represents the meaning of the sentence



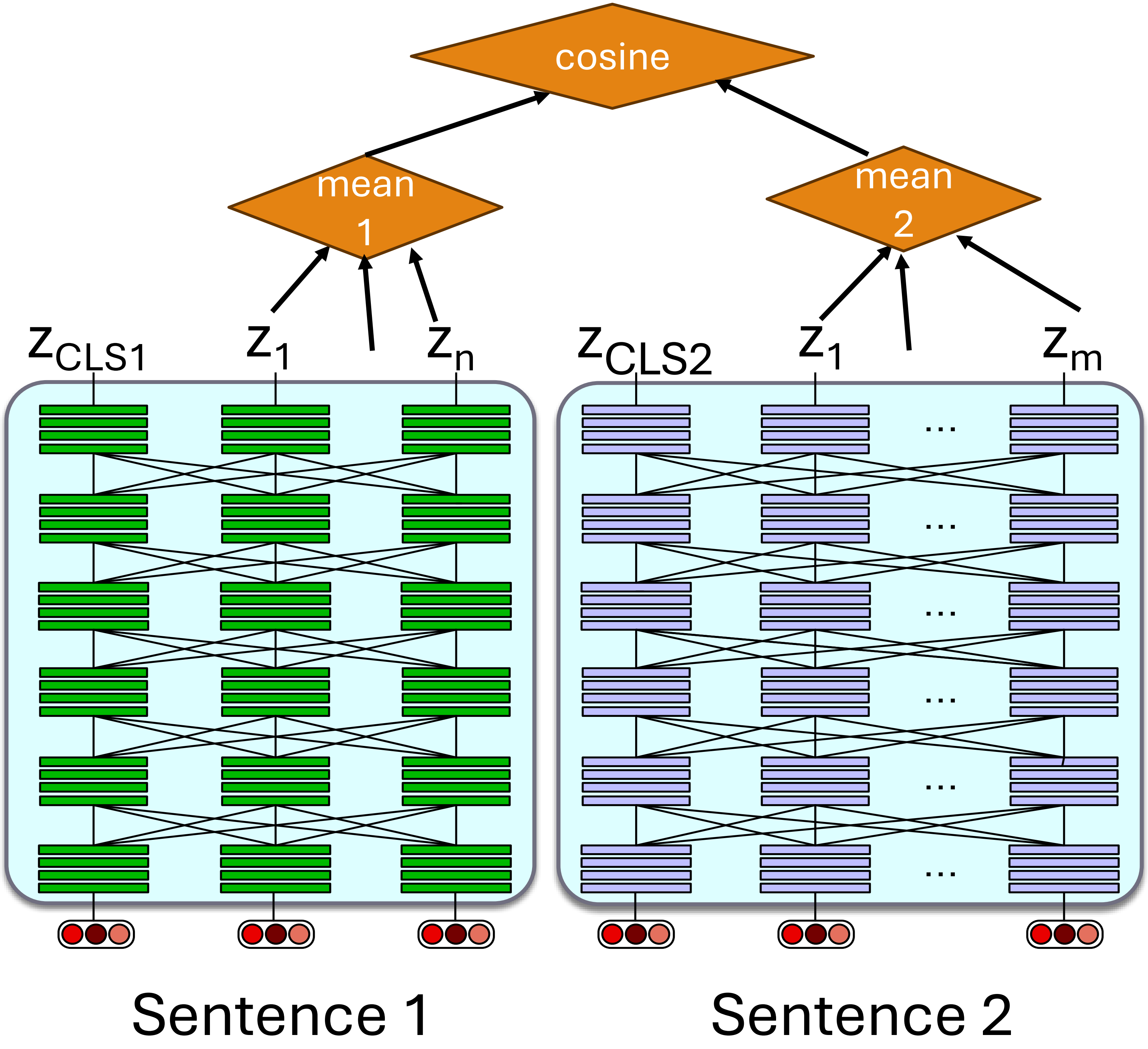
Take the cosine between the CLS tokens!

1. Run each sentence through BERT
2. Take each CLS token vector [768,]
3. Compute their cosine



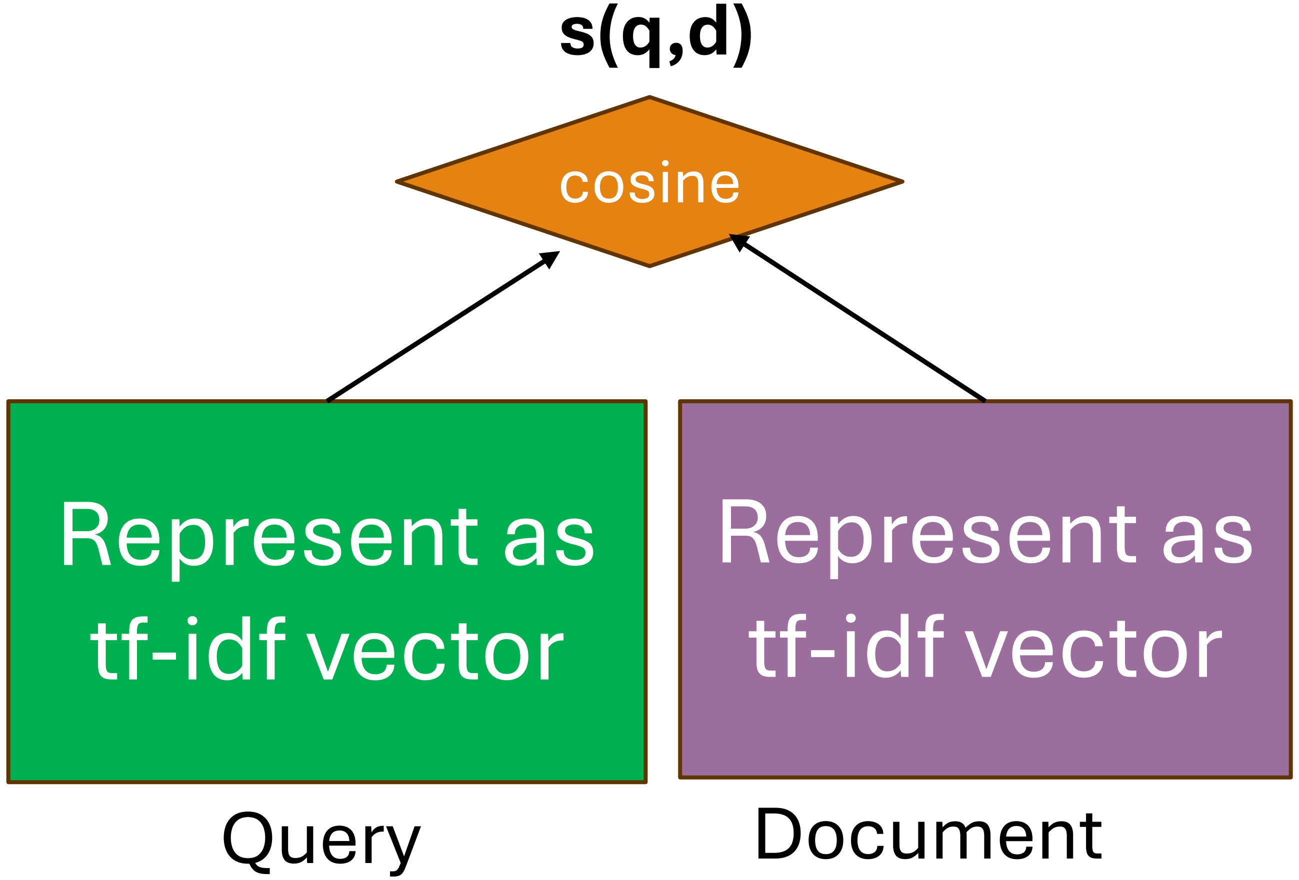
Mean pooling

1. Run each sentence through BERT
2. Average the vectors for each sentence
3. Compute the cosine

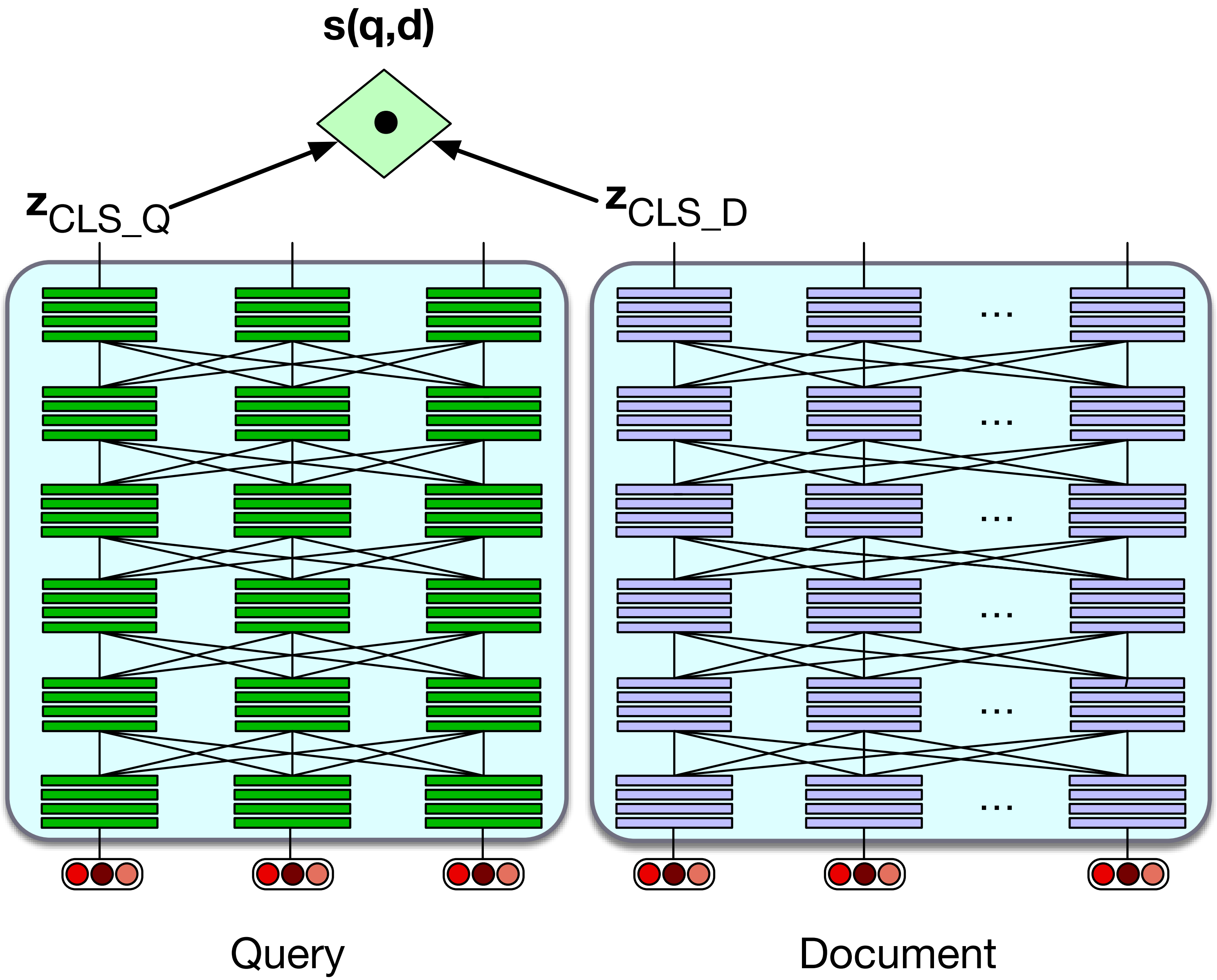


IR: classic tf-idf vs dense retrieval

Classic tf-idf (PA4)



Dense retrieval (PA5)



Embeddings

Computing word and sentence similarity with contextual embeddings