Lecture 12
Embedding: Representations of the meaning of words

Zhizheng Wu

Agenda

- Recap
- Embedding: dense vs sparse
- Static embedding: Word2vec
- Dynamic embedding: BERT

Word sense (concept)

- He wrote several **plays** but only one was produced on Broadway
- Insiders said the company's stock was in **play**
- The runner was out on a **play** by the shortstop

Recommended podcast on play (玩儿）: https://etw.fm/2036

Word representation

- Five words vocabulary: man, walk, wowan, swim, ask
  - 1-of-N encoding/one-hot encoding
    - [1, 0, 0, 0, 0]: man
    - [0, 1, 0, 0, 0]: walk
    - [0, 0, 1, 0, 0]: woman
    - [0, 0, 0, 1, 0]: swim
    - [0, 0, 0, 0, 1]: ask

Words as vectors: Document dimensions

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

similar words have similar vectors
because they tend to occur in similar documents

Words as vectors: Word dimensions

- word-word co-occurrence matrix

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

TF-IDF

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Inner/dot product

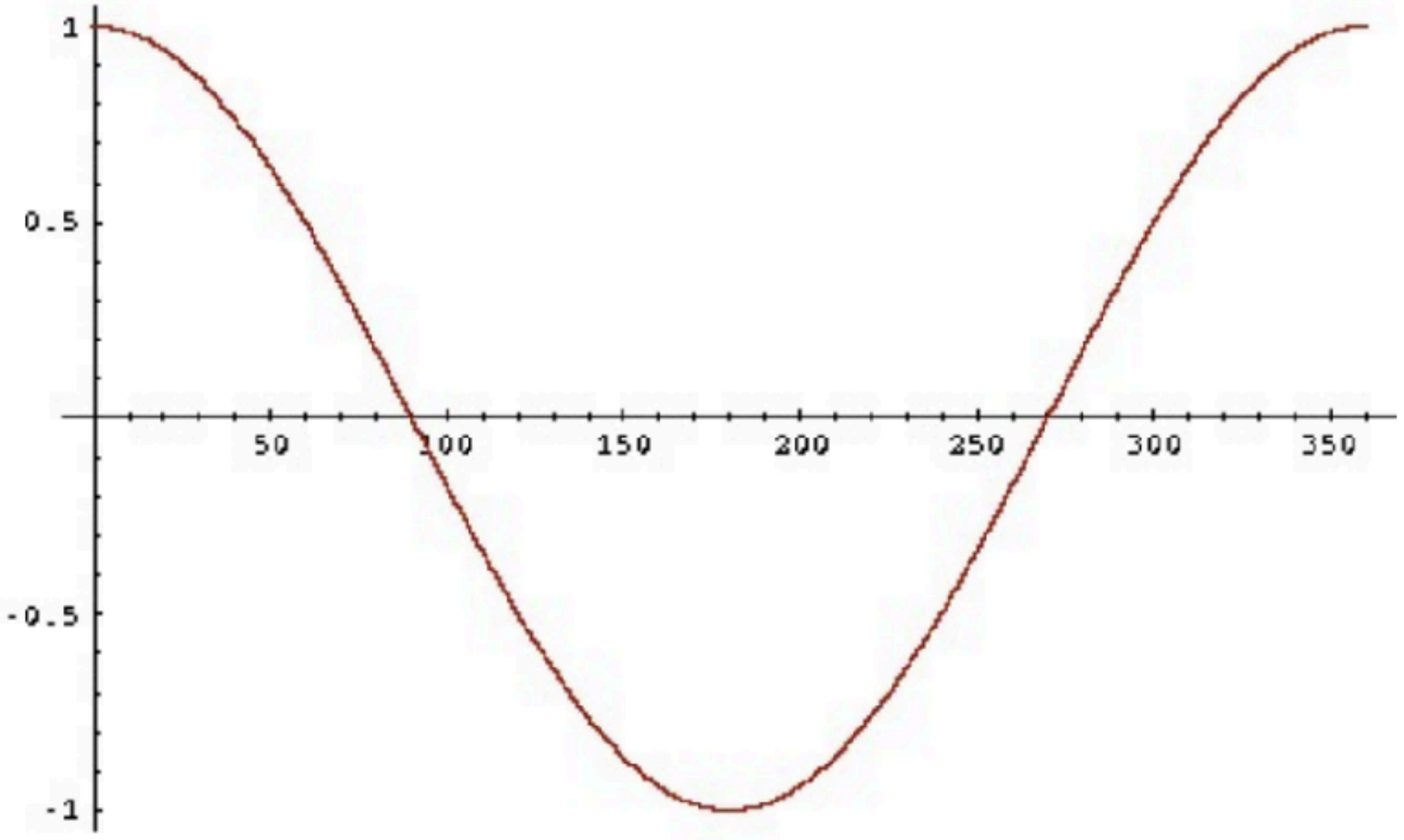
- The dot product between two vectors is a scalar

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

The dot product tends to be **high**
when the two vectors have **large** values in the **same** dimensions

Cosine similarity

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



Pointwise Mutual Information (PMI)

- Do events x and y co-occur more than if they were independent?

$$\mathrm{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI between two words
  - Do words x and y co-occur more than if they were independent?

$$\mathrm{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Section 6.6: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Zhizheng / Cai Xukun

```
cosine_similarity( [-0.4, 0.8, 0.5, -0.2, 0.3] , [-0.3, 0.2, 0.3, -0.4, 0.9] ) = 0.66
```

Zhizheng / Michelle Yeoh

```
cosine_similarity( [-0.4, 0.8, 0.5, -0.2, 0.3] , [-0.5, -0.4, -0.2, 0.7, -0.1] ) = -0.37
```

Embedding: short, dense vector

Sparse versus dense vectors

TF-IDF (or PMI) vectors are

- **long** (length |V|= 20,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors which are

- **short** (length 50-1000)
- **dense** (most elements are non-zero)

Sparse versus dense vectors

- Why dense vectors?
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts
  - Dense vectors may do better at capturing synonymy:
    - car and automobile are synonyms; but are distinct dimensions
    - a word with car as a neighbor and a word with automobile as a neighbor should be similar, but aren't
- In practice, they work better

Static embedding: one fixed embedding for each word in the vocabulary

Dynamic embedding: the vector for each word is different in different contexts

Word2vec

Popular embedding method

Very fast to train

Idea: **predict** rather than **count**

Word2vec provides various options. We'll do:

skip-gram with negative sampling (SGNS)

Skip-gram with negative samples

```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    w      c3      c4
```

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Word2vec

Instead of **counting** how often each word *w* occurs near "*apricot*"

- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

We don't actually care about this task

- But we'll take the learned classifier weights as the word embeddings

Big idea: **self-supervision**:

- A word c that occurs near apricot in the corpus cats as the gold "correct answer" for supervised learning
- No need for human labels
- Bengio et al. (2003); Collobert et al. (2011)

Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word *t* and a neighboring context word *c* as **positive examples**.
2. Randomly sample other words in the lexicon to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Skip-Gram Training Data

Assume a +/- 2 word window, given training sentence:

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 c3 c4

[target]

Skip-Gram Classifier

(assuming a +/- 2 word window)

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

Goal: train a classifier that is given a candidate (**w**ord, **c**ontext) pair

(apricot, jam)

(apricot, aardvark)

…

And assigns each pair a probability:

$P(+|w, c)$

$P(-|w, c) = 1 - P(+|w, c)$

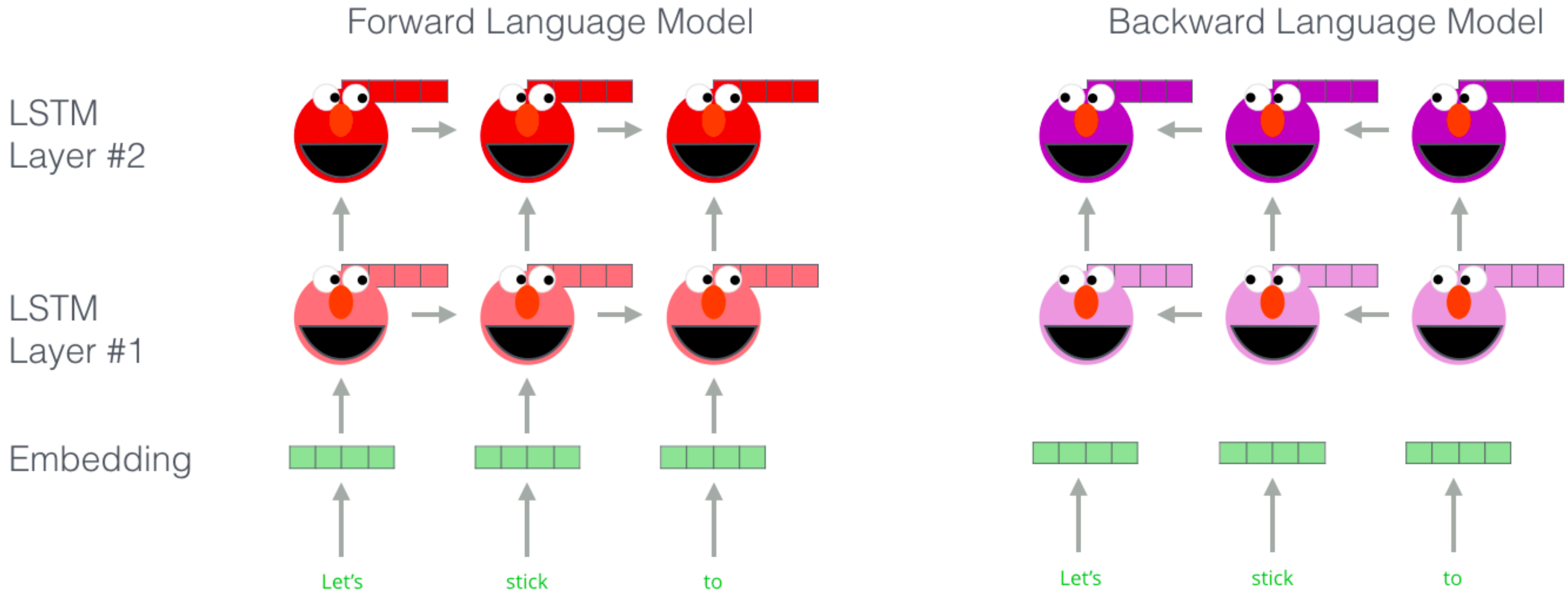
Context matters

Context matters

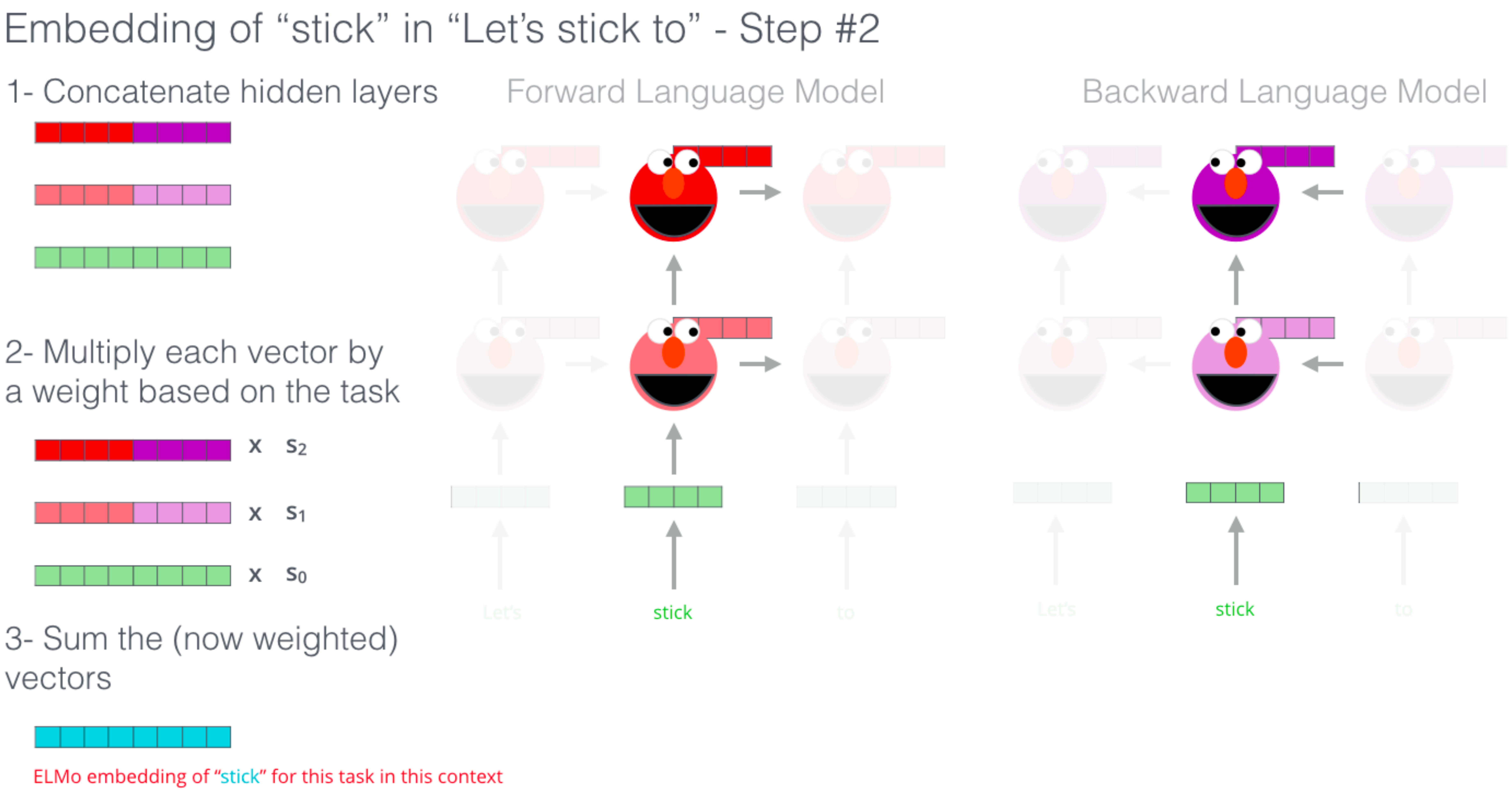
- Pass that book to me
- Book a flight for me

Embeddings from Language Model (ELMO)

Embedding of “stick” in “Let’s stick to” - Step #1



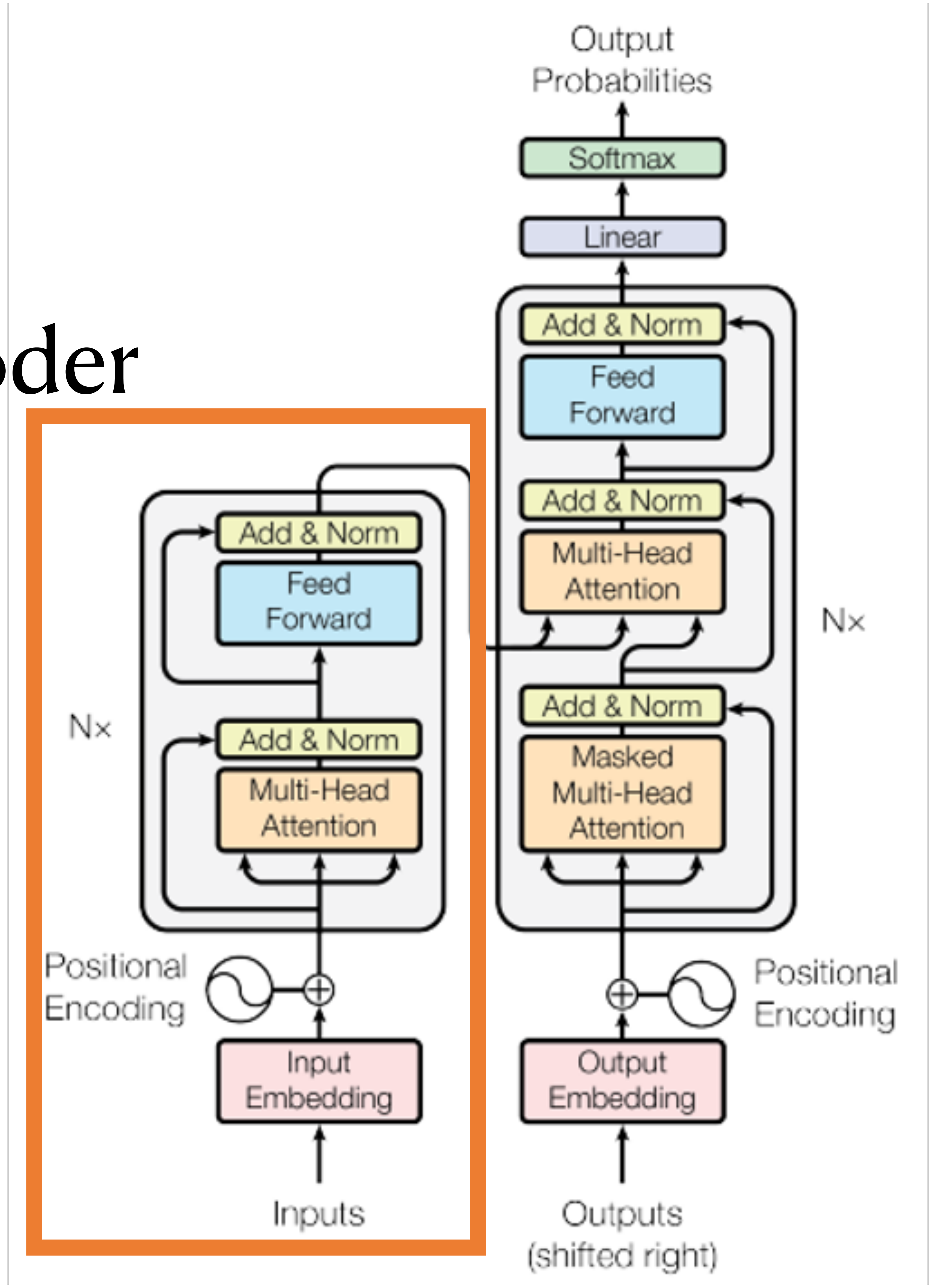
Embeddings from Language Model (ELMO)



Bidirectional Encoder Representations from Transformers (BERT)

BERT = Encoder of Transformer

Encoder

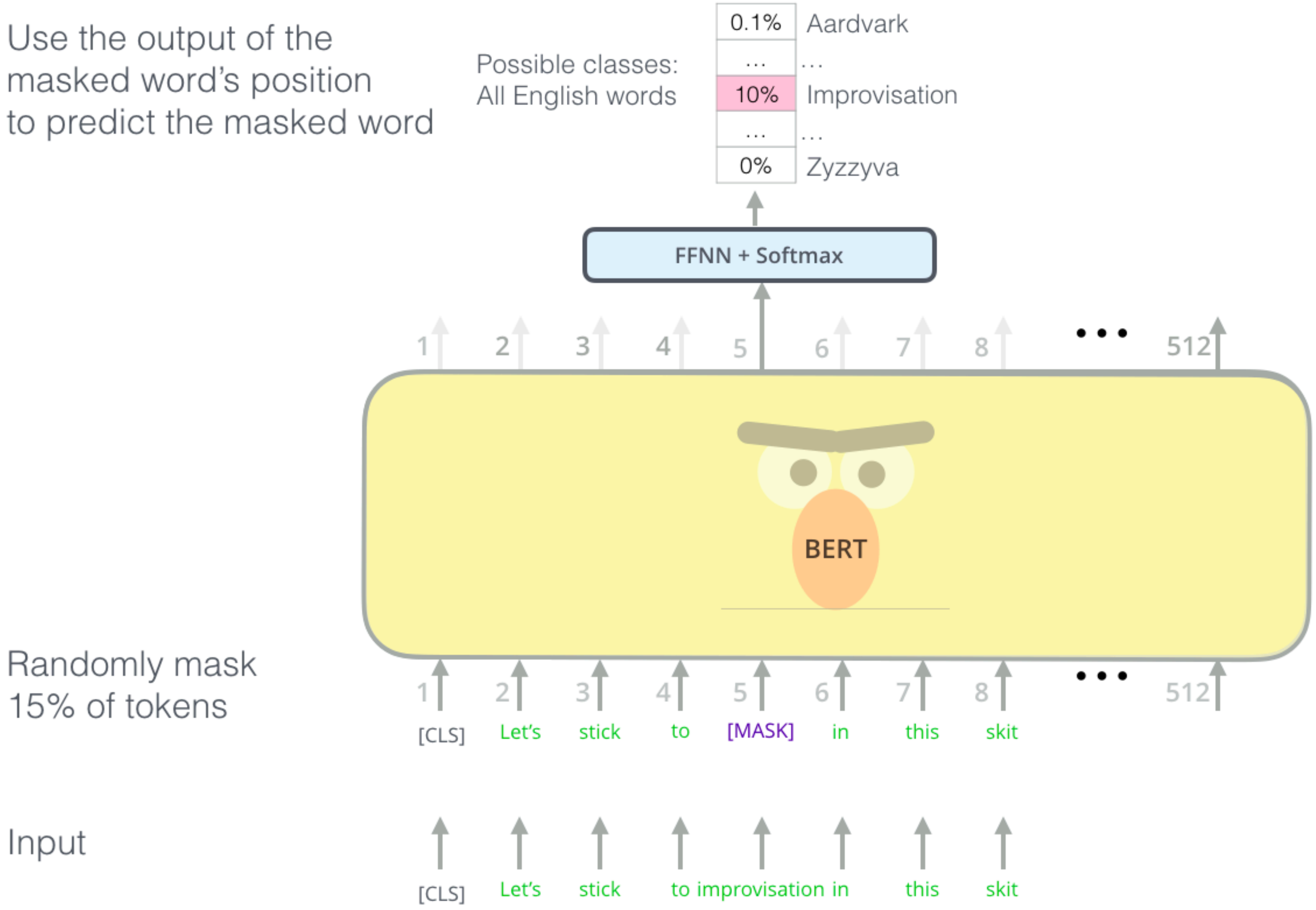


“We’ll use transformer encoders”, said BERT.

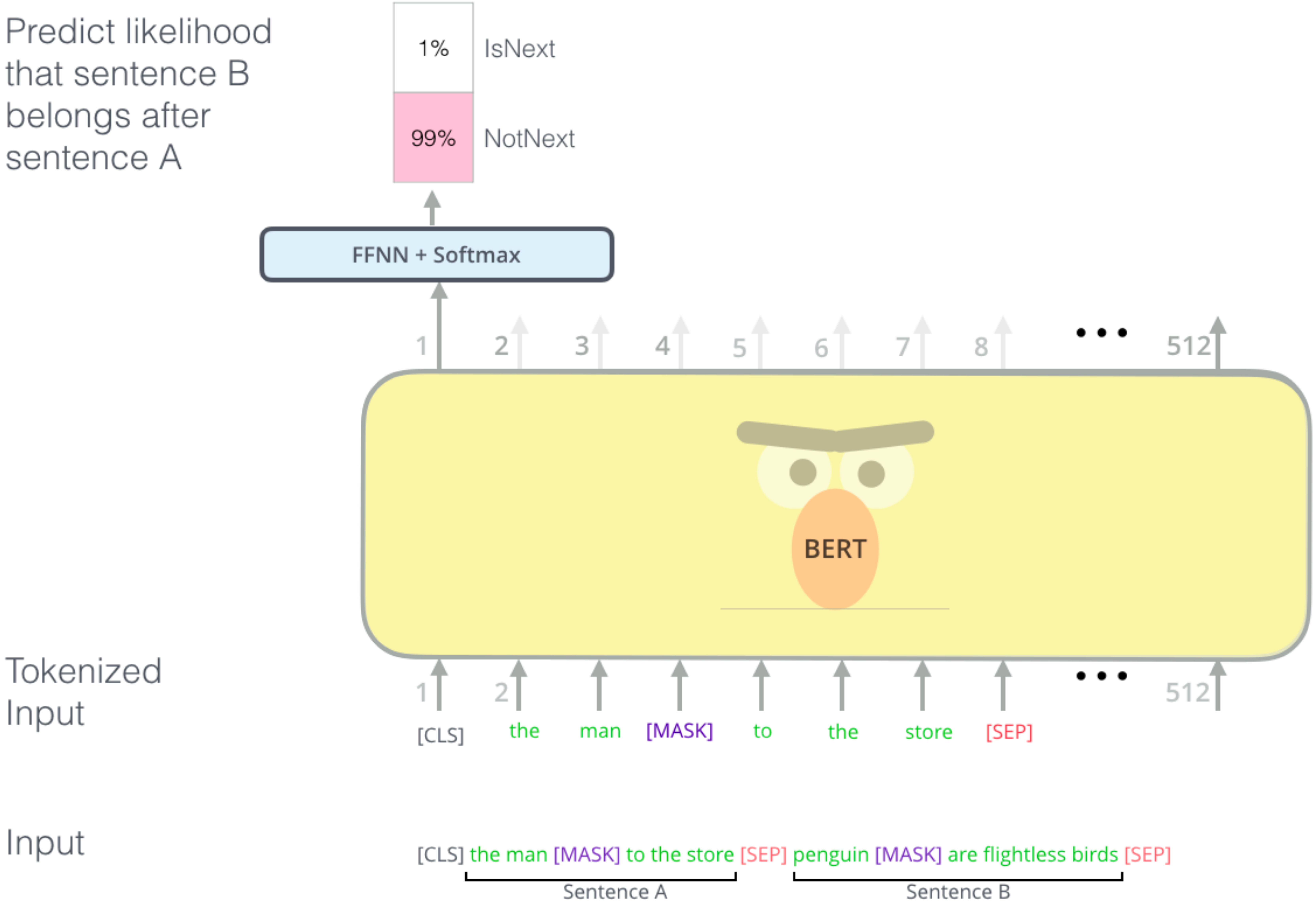
“This is madness”, replied Ernie, “Everybody knows bidirectional conditioning would allow each word to indirectly see itself in a multi-layered context.”

“We’ll use masks”, said BERT confidently.

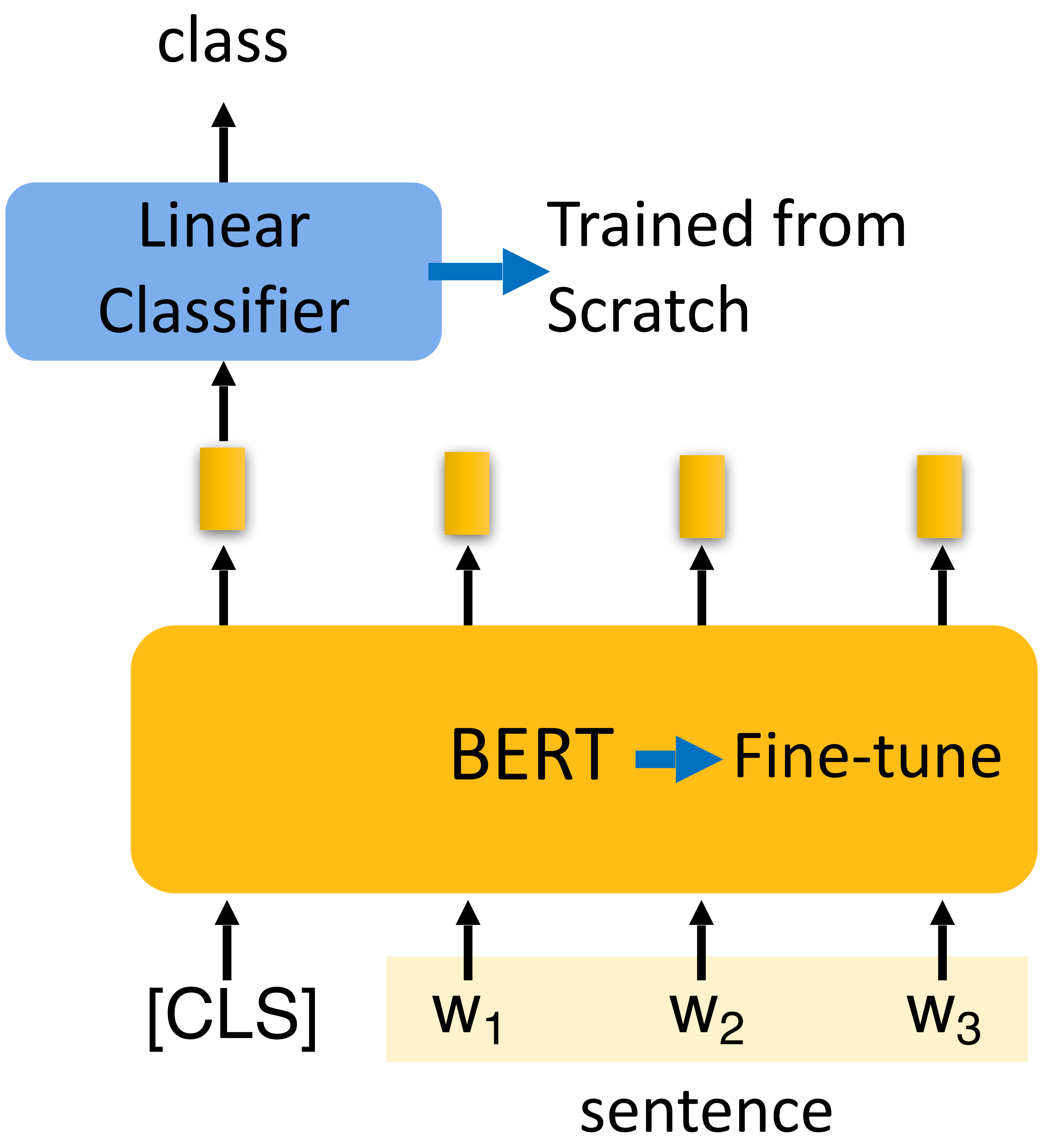
Masked language model



Next sentence prediction



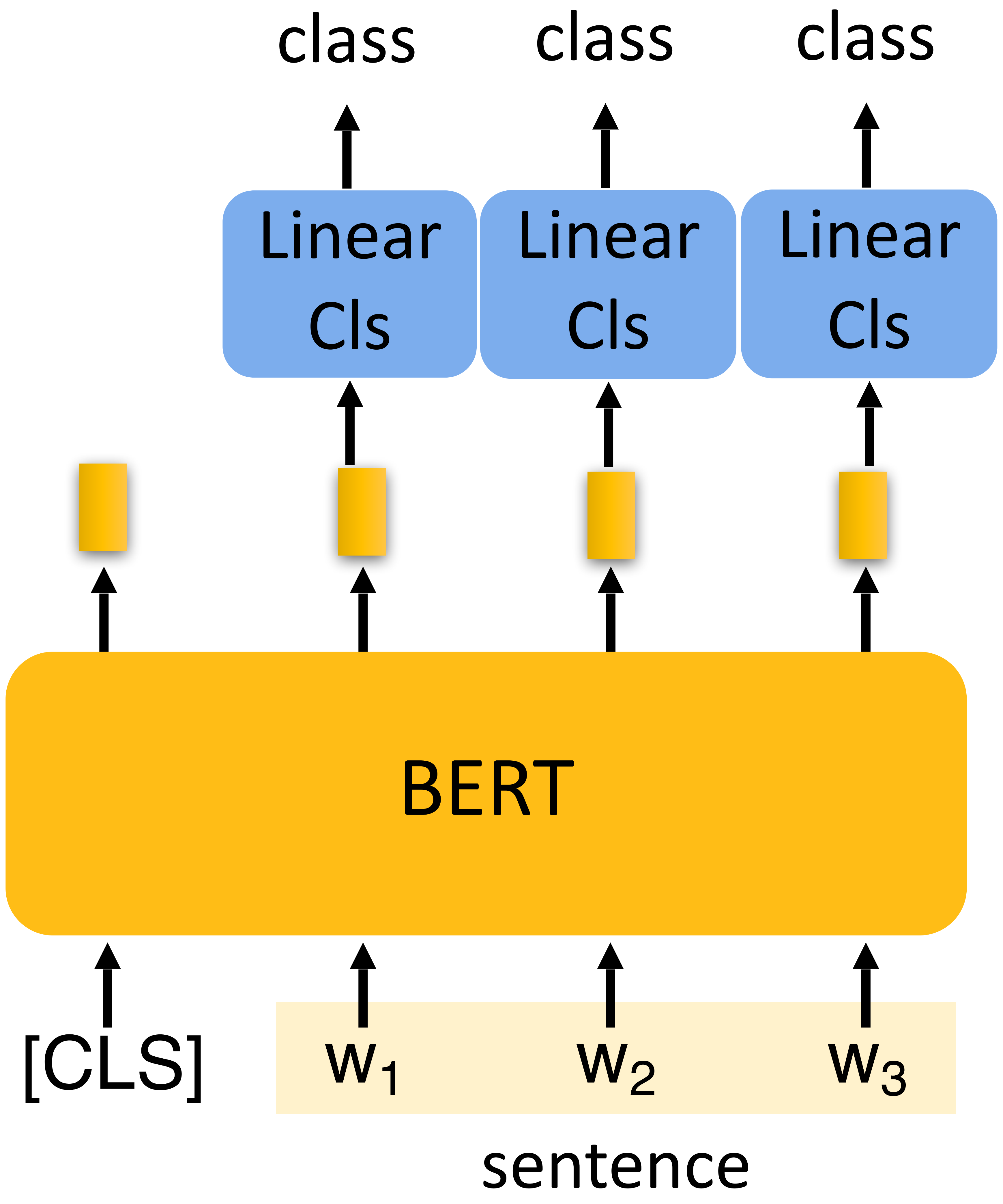
How to use BERT – Case 1



Input: single sentence,
output: class

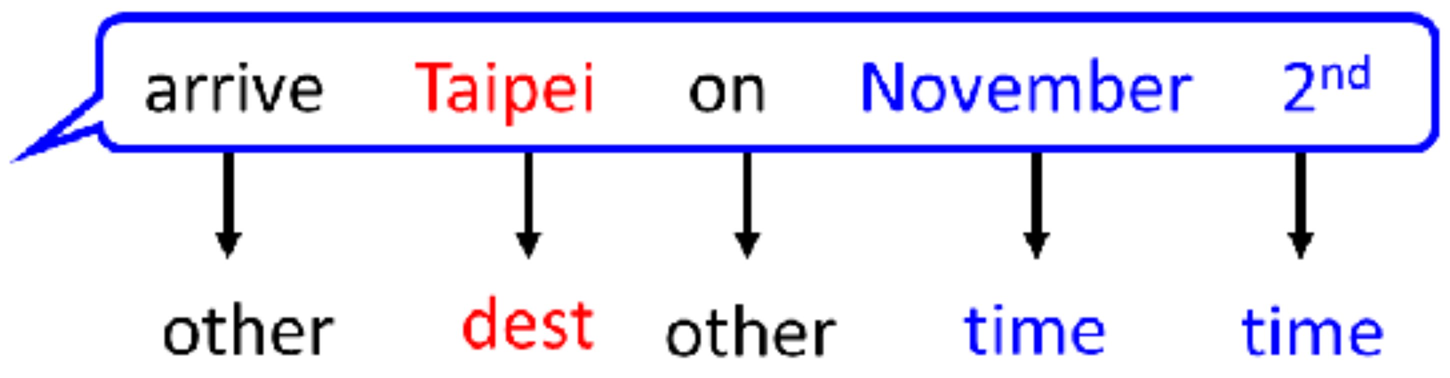
Use cases:
Sentiment analysis
Document Classification

How to use BERT – Case 2

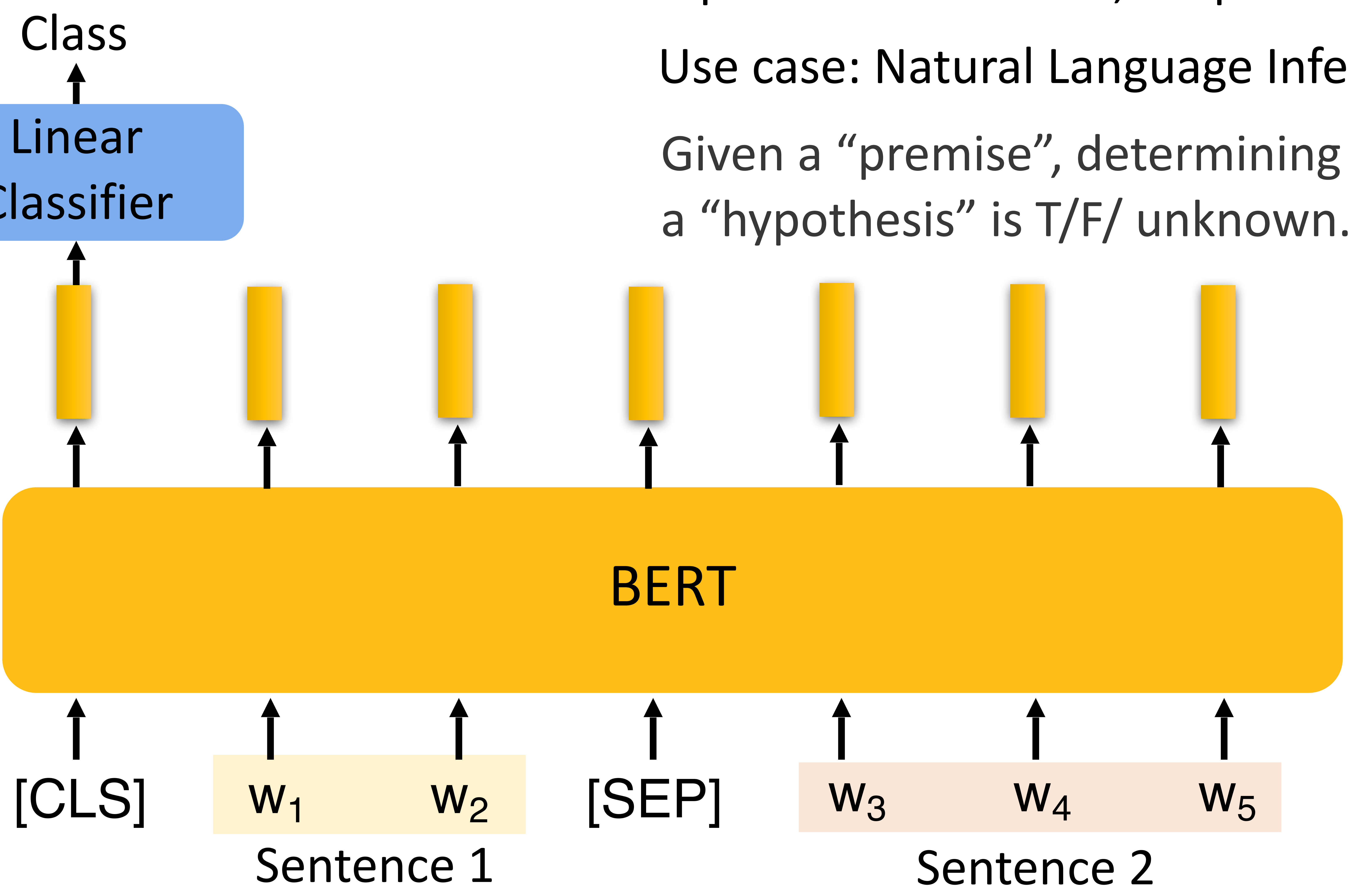


Input: single sentence,
output: class of each word

Use case: Slot filling



How to use BERT – Case 3



Input: two sentences, output: class

Use case: Natural Language Inference

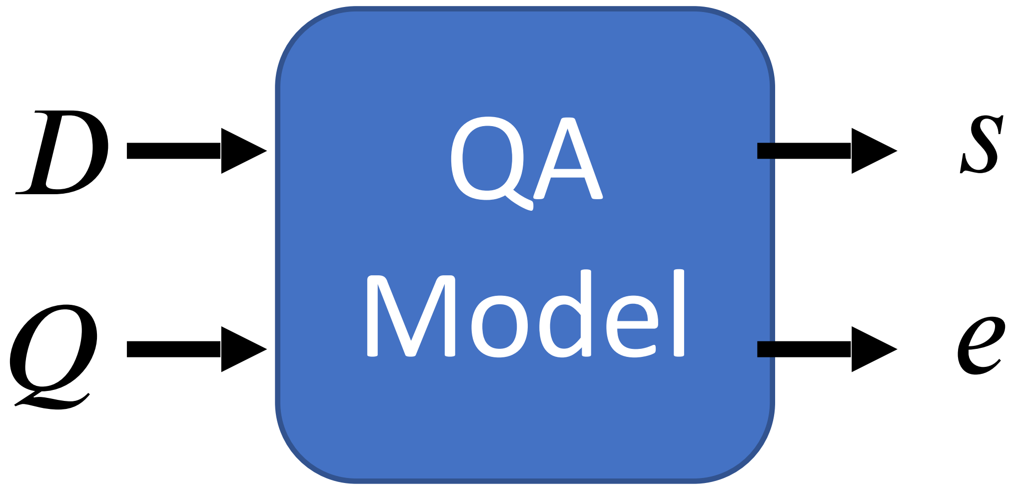
Given a “premise”, determining whether a “hypothesis” is T/F/ unknown.

How to use BERT – Case 4

- Extraction-based Question Answering (QA) (E.g. SQuAD)

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$



output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

What causes precipitation to fall?
gravity $s = 17, e = 17$

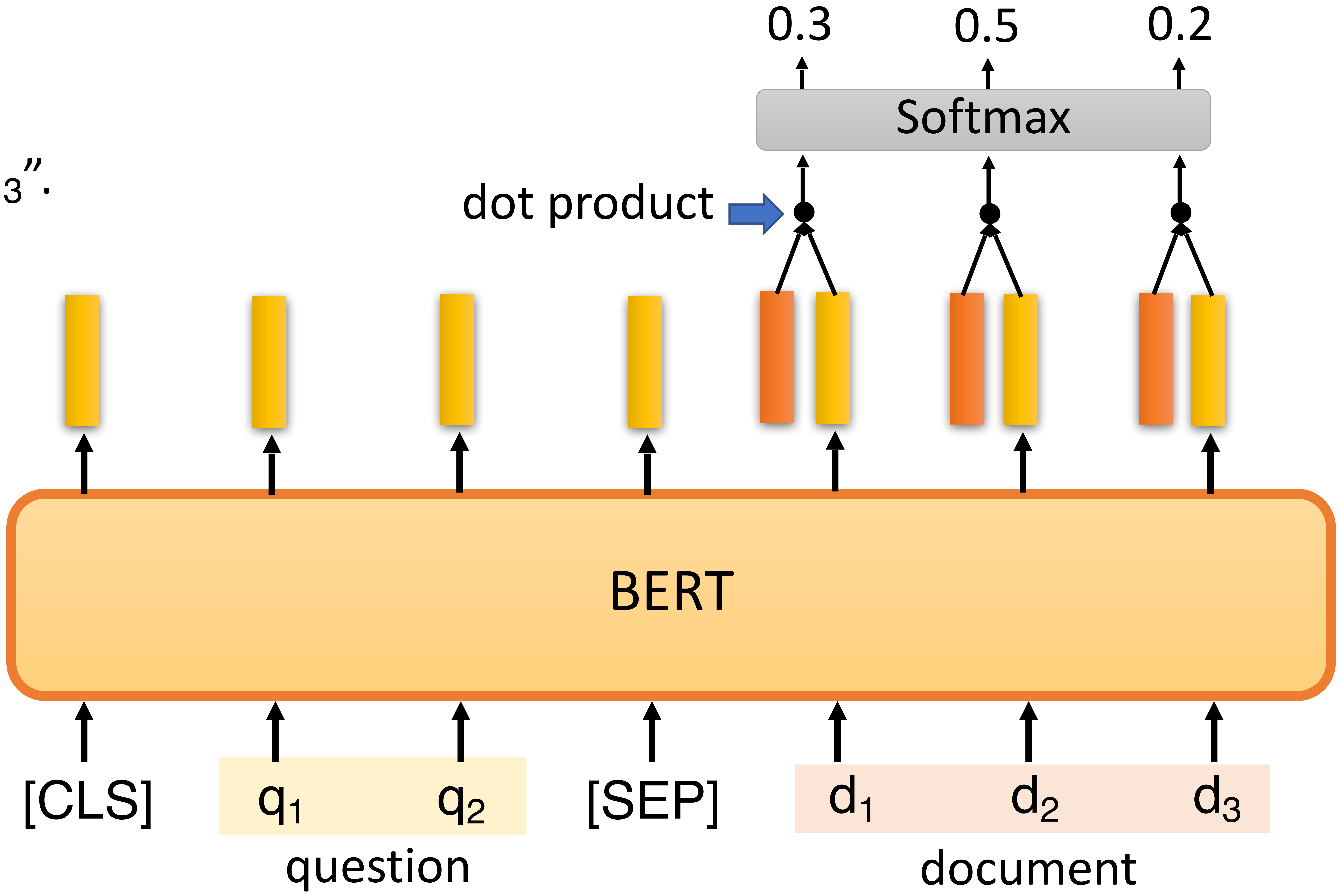
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud $s = 77, e = 79$

How to use BERT – Case 4

s = 2, e = 3

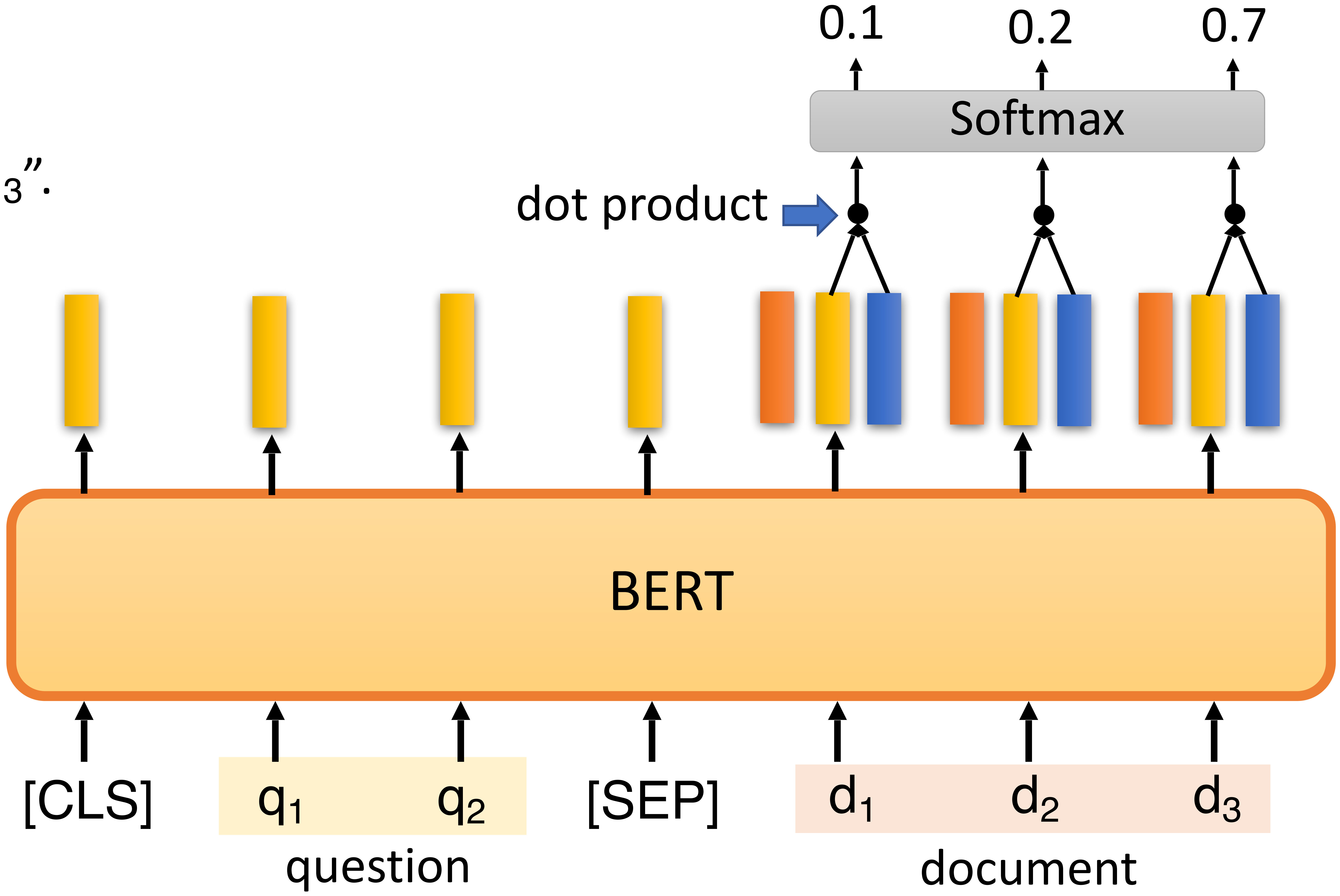
The answer is “d_2 d_3”.



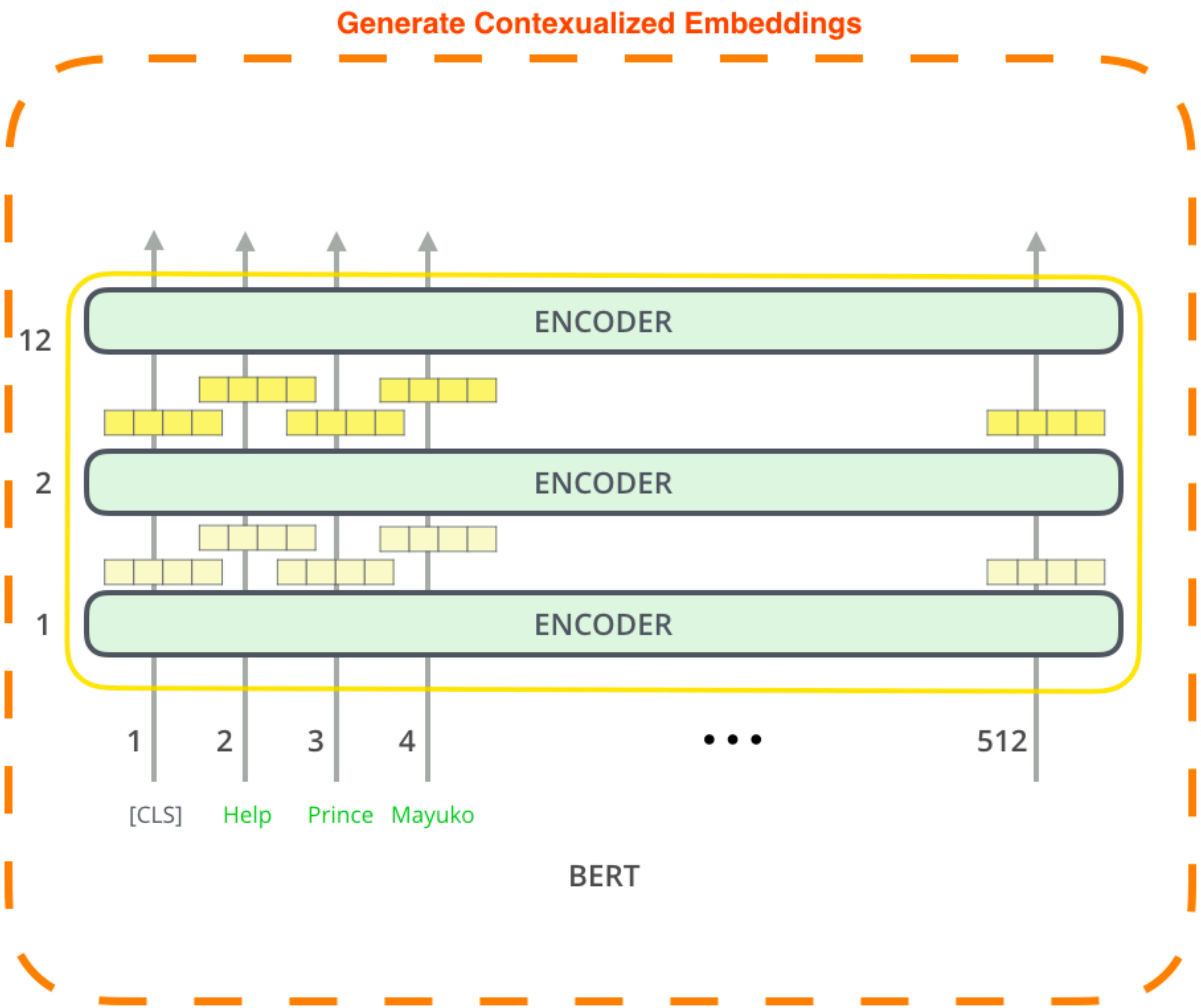
How to use BERT – Case 4

s = 2, e = 3

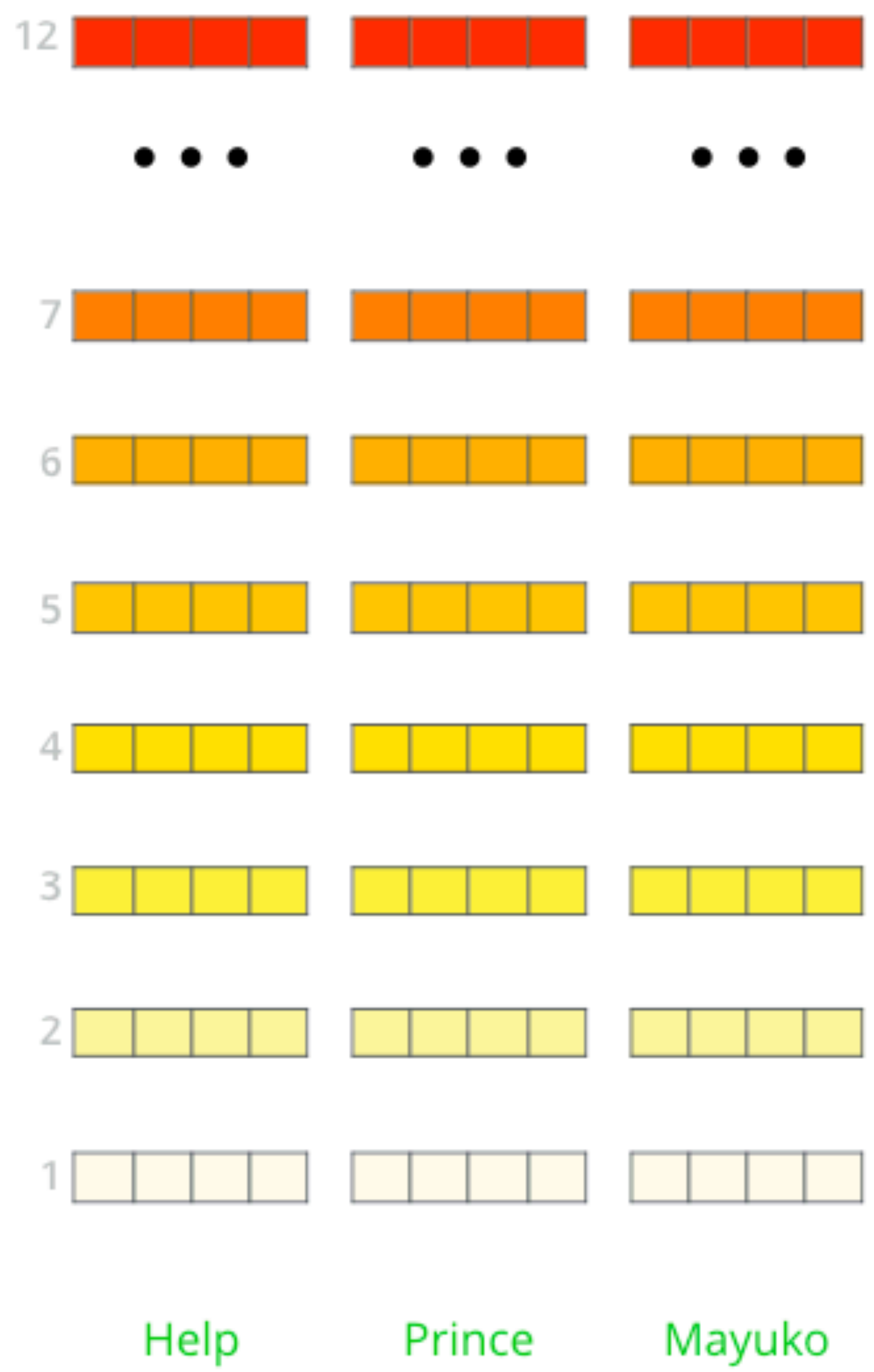
The answer is “d_2 d_3”.



BERT for feature extraction



The output of each encoder layer along each token's path can be used as a feature representing that token.

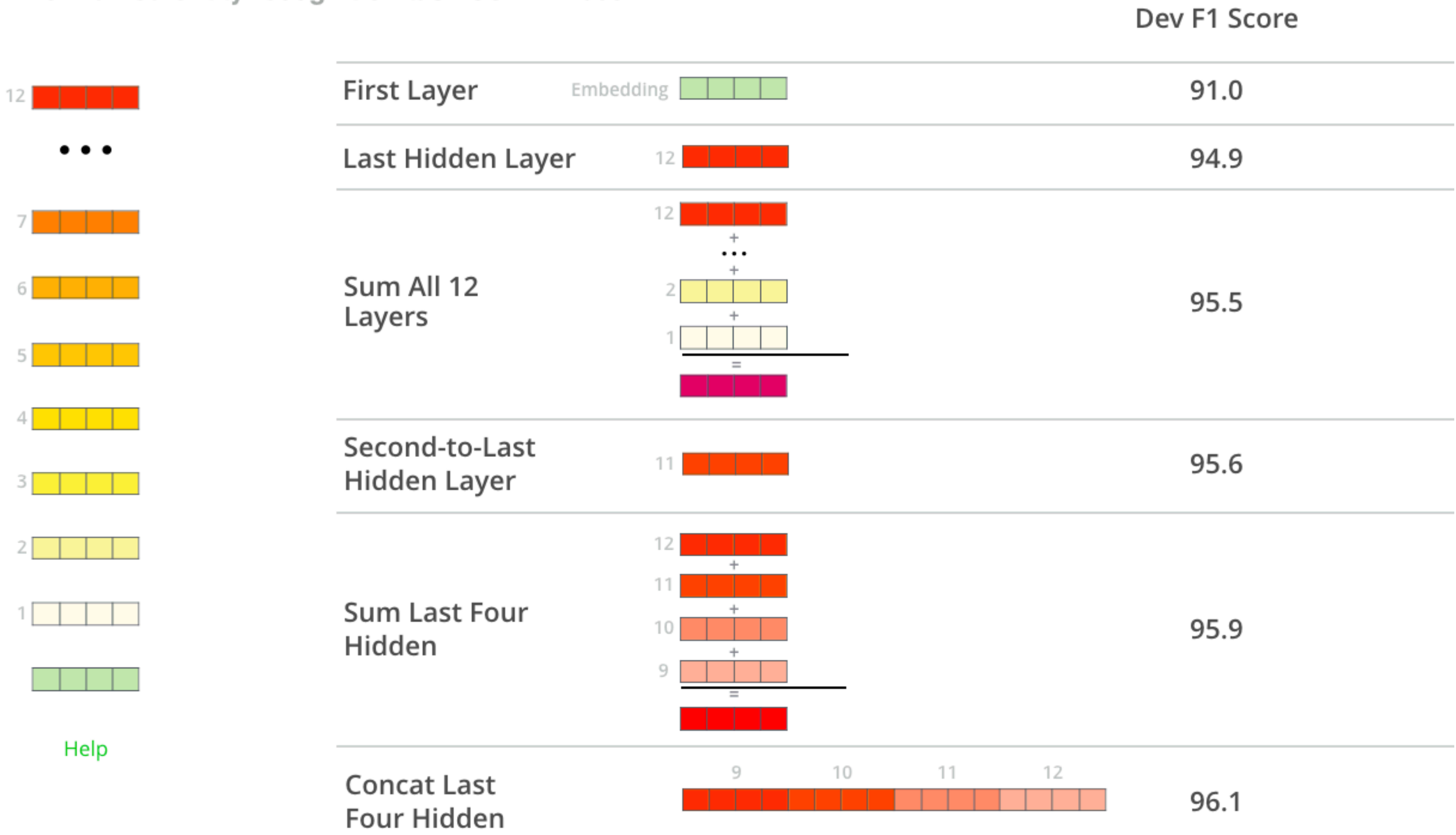


But which one should we use?

Contextual embedding

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER



Reading materials

- Chapter 6: Vector Semantics and Embeddings
  - https://web.stanford.edu/~jurafsky/slp3/6.pdf
- Finding the Words to Say: Hidden State Visualizations for Language Models
  - http://jalammar.github.io/hidden-states/
- The Illustrated Word2vec
  - https://jalammar.github.io/illustrated-word2vec/