AI VIETNAM
All-in-One Course

Deep Architectures for POS Tagging

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2024

Outline

- What We Have from Text Classification?
- Introduction to POS Tagging
- POS Tagging Using Classification
- Using Different Model Archectures
- PyTorch Implementation

NLP Applications

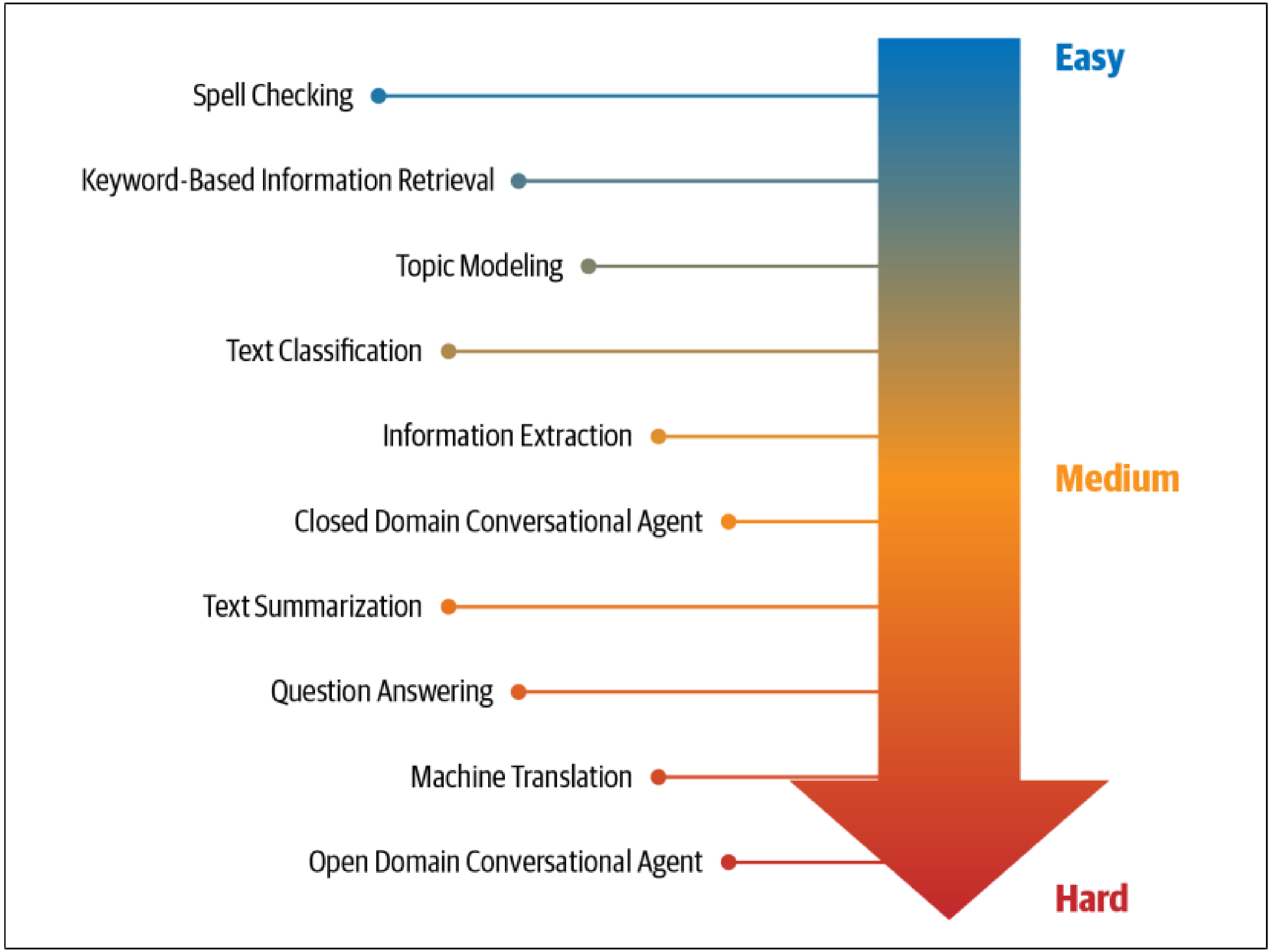
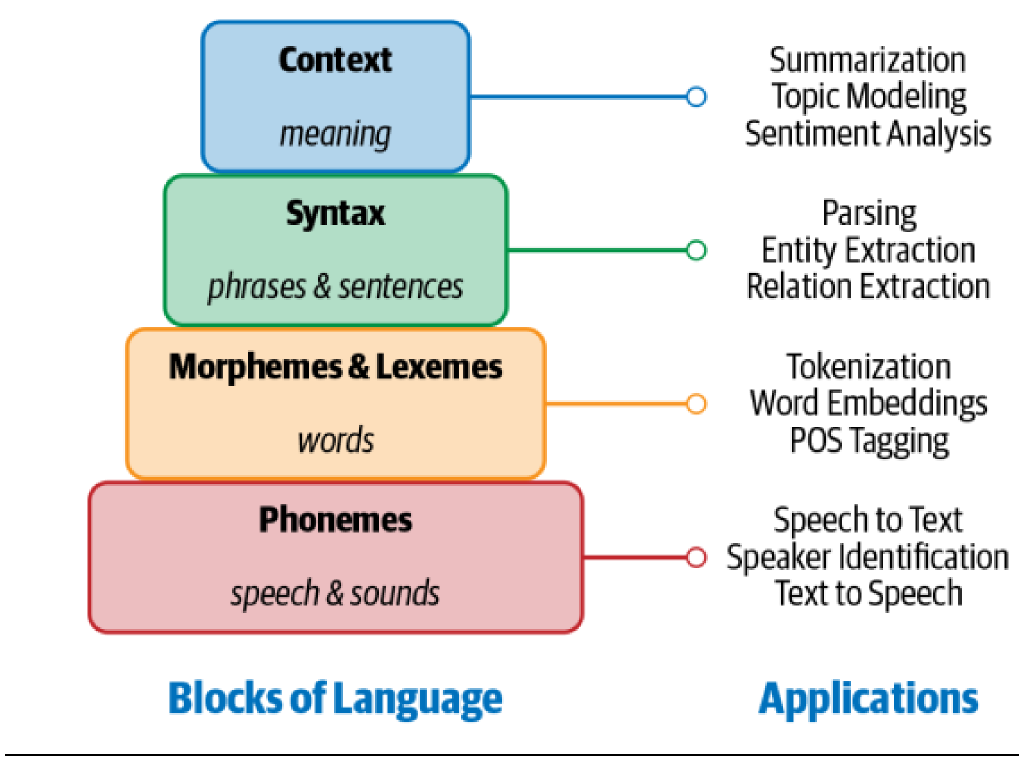


Figure 1-2. NLP tasks organized according to their relative difficulty



Building blocks of language and their applications

Practical Natural Language Processing by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana

Applications of Text Analysis

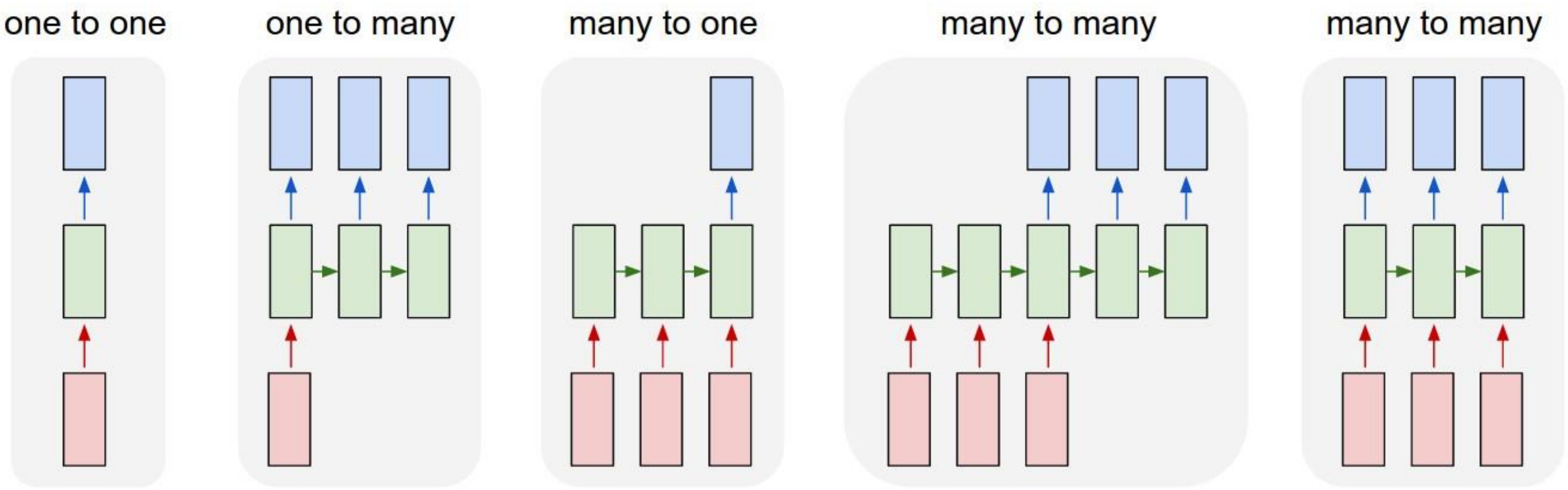


Image classification | Image Captioning | Text classification | Machine Translation Recognition | POS Tagging

Source: The Unreasonable Effectiveness of Recurrent Neural Networks

NLP Applications

❖ Information Extraction

Part-Of-Speech Tagging

Larry went to the office by bus.

Larry - PROPN	office - NOUN
went - VERB	by - PREP
to - PREP	bus - NOUN
the - ART	. - PUNCT

Named Entity Recognition

In the 19th century, there was something called the "cult of domesticity" for many American women. This meant that most married women were expected to stay in the home and raise children. As in other countries, American wives were very much under the control of their husband, and had almost no rights. Women who were not married had only a few jobs open to them, such as working in clothing factories and serving as maids. By the 19th century, women such as Lucretia Mott and Elizabeth Cady Stanton thought that women should have more rights. In 1848, many of these women met and agreed to fight for more rights for women, including voting. Many of the women involved in the movement for women's rights were also involved in the movement to end slavery.

WIKIPEDIA

Tag colors:

LOCATION PERSON TERM DATE CONDITION PROCESS PEOPLE

https://mobidev.biz/blog/natural-language-processing-nlp-use-cases-business

Text Classification

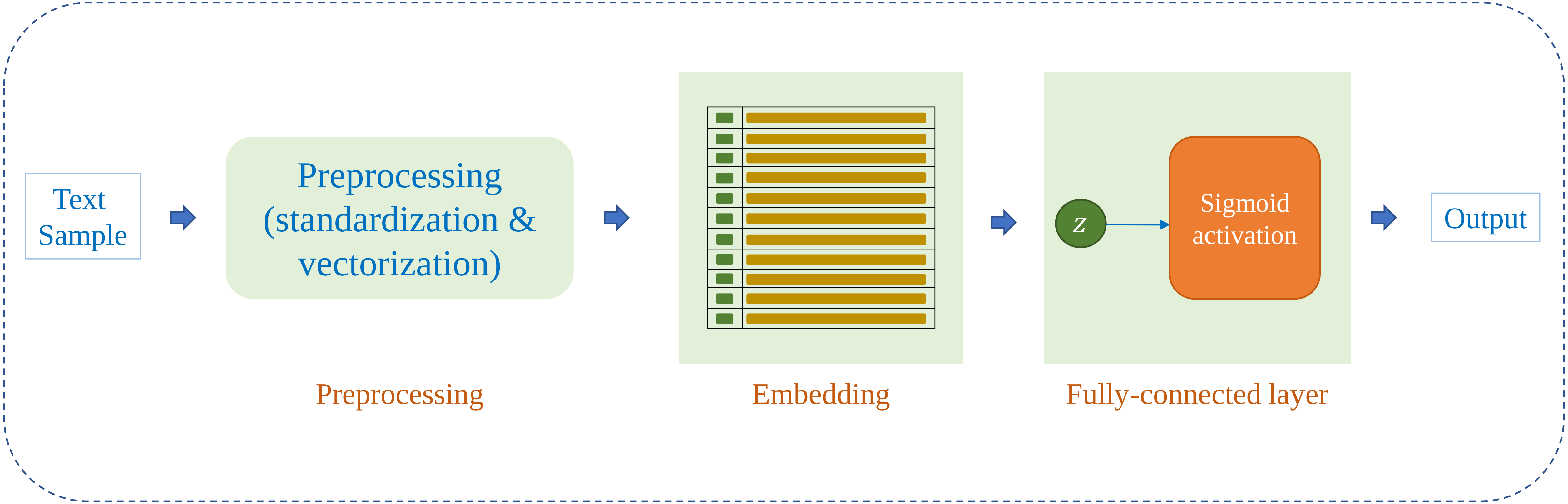
❖ IMDB dataset

- 50,000 movie review for sentiment analysis
- Consist of: + 25,000 movie review for training
  + 25,000 movie review for testing
- Label: positive – negative

“A wonderful little production. The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece…..”	positive
“This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. By 1990, the show was not really funny anymore, and it's continued its decline further to the complete waste of time it is today….”	negative
“I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy. The plot is simplistic, but the dialogue is witty and the characters are likable (even the well bread suspected serial killer)….”	positive
“BTW Carver gets a very annoying sidekick who makes you wanna shoot him the first three minutes he's on screen.”	negative

Text Classification

❖ Simple approach



Embedding

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

- Example corpus
  - sample1: ‘We are learning AI’
  - sample2: ‘AI is a CS topic’

(1) Build vocabulary from corpus

We are learning AI	AI is a CS topic

Standardize

we	are	learning	ai

ai	is	a	cs	topic

```python
from torchtext.data.utils import get_tokenizer

sample1 = 'We are learning AI'
sample2 = 'AI is a CS topic'

# Define tokenizer function
tokenizer = get_tokenizer('basic_english')
sample1_tokens = tokenizer(sample1)
sample2_tokens = tokenizer(sample2)

print(sample1_tokens)
print(sample2_tokens)
```

```
['we', 'are', 'learning', 'ai']
['ai', 'is', 'a', 'cs', 'topic']
```

Embedding

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

- Example corpus
  sample1: 'We are learning AI'
  sample2: 'AI is a CS topic'

(1) Build vocabulary from corpus

#different words are enormous

How to represent 'text' effectively?

➡ Use a limited number of words

➡ Get data sample-by-sample

```python
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator

sample1 = 'We are learning AI'
sample2 = 'AI is a CS topic'
data = [sample1, sample2]

# Create a function to yield list of tokens
tokenizer = get_tokenizer('basic_english')
def yield_tokens(examples):
    for text in examples:
        yield tokenizer(text)

# Create vocabulary
vocab_size = 8
vocab = build_vocab_from_iterator(yield_tokens(data),
                                  max_tokens=vocab_size,
                                  specials=["<unk>",
                                            "<pad>"])

vocab.set_default_index(vocab["<unk>"])
```

```python
vocab.get_stoi()
```

```
{'<unk>': 0,
 '<pad>': 1,
 'ai': 2,
 'a': 3,
 'is': 6,
 'are': 4,
 'learning': 7,
 'cs': 5}
```

Embedding

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

- Example corpus
  sample1: 'We are learning AI'
  sample2: 'AI is a CS topic'

(1) Build vocabulary from corpus

(2) Transform text into features

'We' 'are' 'learning' 'AI'

```
tokens = tokenizer(sample1)
print(tokens)

sample1_tokens = [vocab[token] for token in tokens]
print(sample1_tokens)
```

```
['we', 'are', 'learning', 'ai']
[0, 4, 7, 2]
```

```
tokens = tokenizer(sample2)
print(tokens)

sample2_tokens = [vocab[token] for token in tokens]
print(sample2_tokens)
```

```
['ai', 'is', 'a', 'cs', 'topic']
[2, 6, 3, 5, 0]
```

We are learning AI						AI is a CS topic				

Standardize

we	are	learning	ai		ai	is	a	cs	topic

Vectorization

0	4	7	2	1		2	6	3	5	0

Embedding

- Example corpus

  sample1: ‘We are learning AI’

  sample2: ‘AI is a CS topic’

(1) Build vocabulary from corpus

(2) Transform text into features

We are learning AI					AI is a CS topic				

Standardize

we	are	learning	ai		ai	is	a	cs	topic

Vectorization

0	4	7	2	1	2	6	3	5	0

index	0	1	2	3	4	5	6	7
word	[UNK]	pad	ai	a	are	cs	is	learning

```python
def vectorize(text, vocab, seq_len):
    tokens = tokenizer(text)
    tokens = [vocab[token] for token in tokens]

    num_pads = sequence_length - len(tokens)
    tokens = tokens[:sequence_length]
           + [vocab["<pad>"]]*num_pads

    return torch.tensor(tokens, dtype=torch.long)

# Vectorize the samples
sequence_length = 5
vectorized_sample1 = vectorize(sample1, vocab,
                               sequence_length)
vectorized_sample2 = vectorize(sample2, vocab,
                               sequence_length)


print("Vectorized Sample 1:", vectorized_sample1)
print("Vectorized Sample 2:", vectorized_sample2)
```

```
Vectorized Sample 1: tensor([0, 4, 7, 2, 1])
Vectorized Sample 2: tensor([2, 6, 3, 5, 0])
```

```python
sample3 = 'AI topic in CS is difficult'
vectorized_sample3 = vectorize(sample3, vocab,
                               sequence_length)
print(vectorized_sample3)
```

```
tensor([2, 0, 0, 5, 6])
```

Embedding Layer

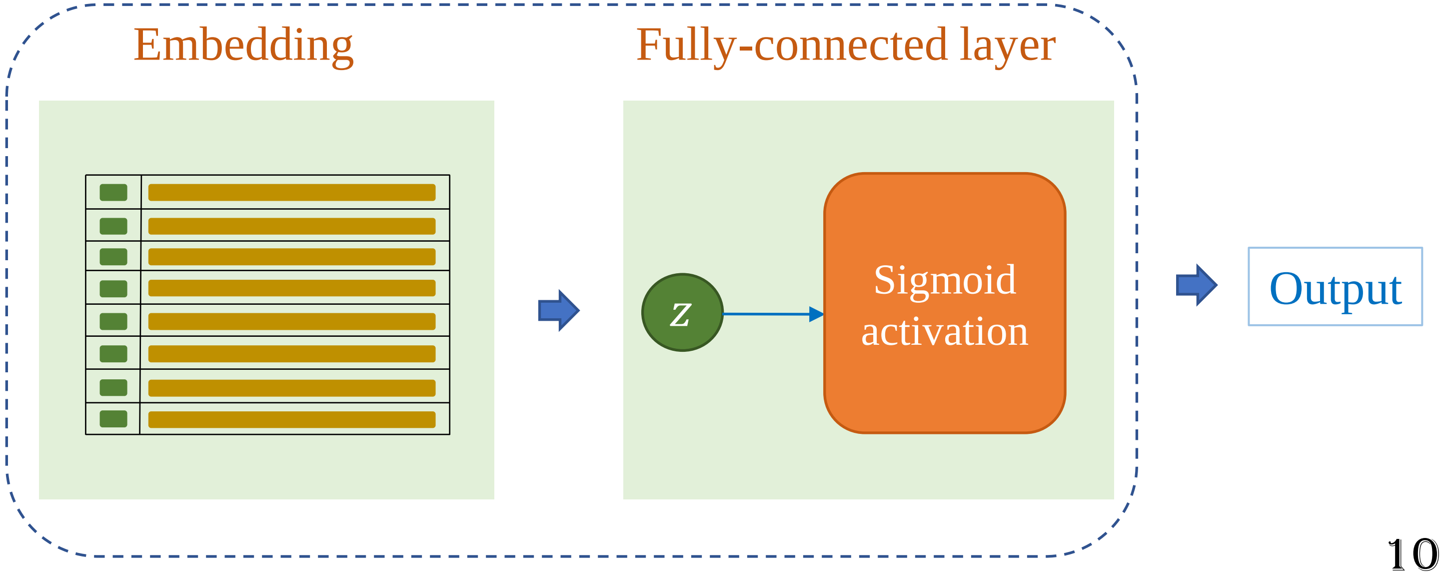
```python
vocab_size = 8
embed_dim = 4
embedding = nn.Embedding(vocab_size,
                         embed_dim)
```

```
Parameter containing:
tensor([[-0.1882,  0.5530,  1.6267,  0.7013],
        [ 1.7840, -0.8278, -0.2701,  1.3586],
        [ 1.0281, -1.9094,  0.3182,  0.4211],
        [-1.3083, -0.0987,  0.7647, -0.3680],
        [ 0.2293,  1.3255,  0.1318,  2.0501],
        [ 0.4058, -0.6624, -0.8745,  0.7203],
        [ 0.5582,  0.0786, -0.6817,  0.6902],
        [ 0.4309, -1.3067, -0.8823,  1.5977]],
```

index	word
0	[UNK]
1	[pad]
2	ai
3	a
4	are
5	cs
6	is
7	learning

We are learning AI

0
4
7
2
1



Revisit input x

Convert from text to numbers

index	word
0	[UNK]
1	[pad]
2	ai
3	a
4	are
5	cs
6	is
7	learning

A sample X

We are learning AI → 0, 4, 7, 2, 1

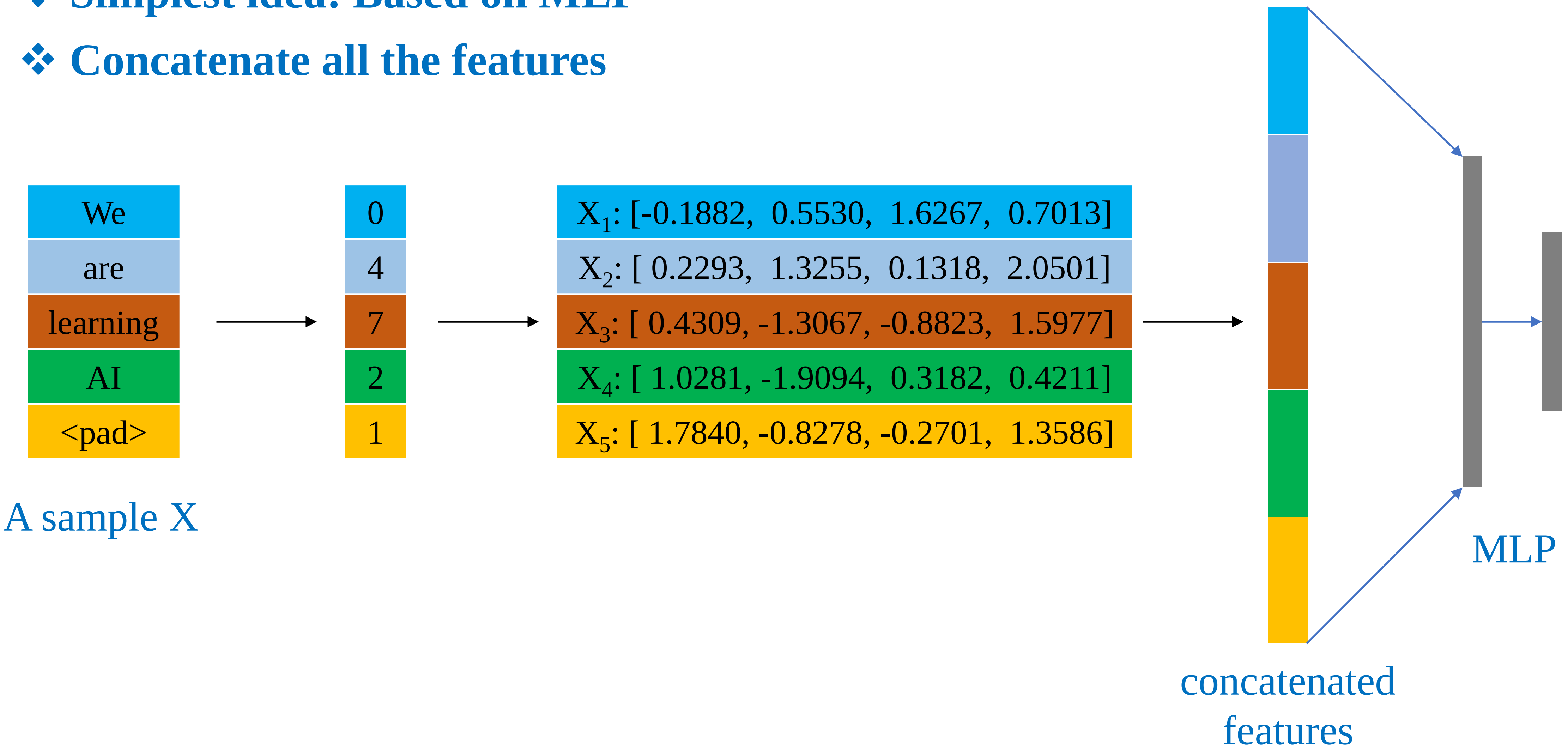
```
Parameter containing:
tensor([[-0.1882,  0.5530,  1.6267,  0.7013],
        [ 1.7840, -0.8278, -0.2701,  1.3586],
        [ 1.0281, -1.9094,  0.3182,  0.4211],
        [-1.3083, -0.0987,  0.7647, -0.3680],
        [ 0.2293,  1.3255,  0.1318,  2.0501],
        [ 0.4058, -0.6624, -0.8745,  0.7203],
        [ 0.5582,  0.0786, -0.6817,  0.6902],
        [ 0.4309, -1.3067, -0.8823,  1.5977]],
```

A sample X

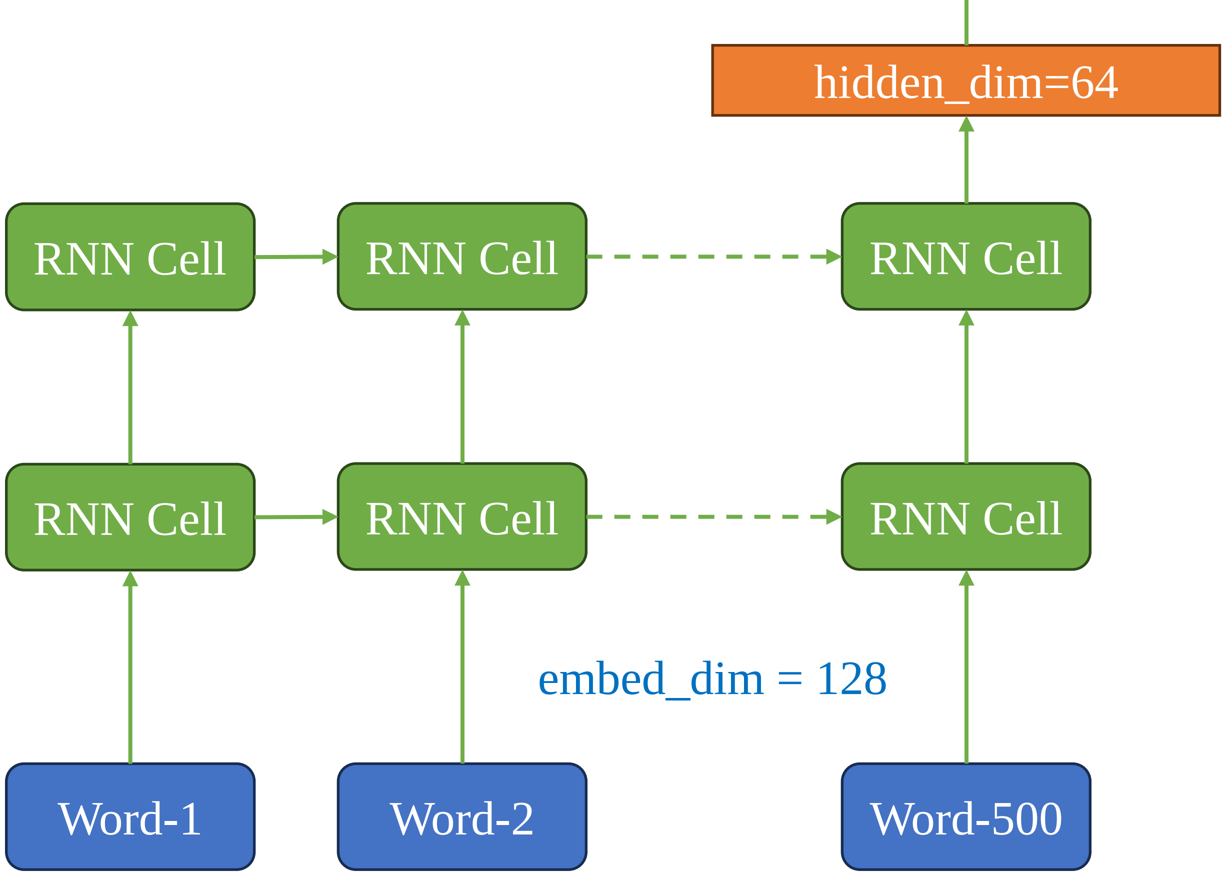
We	0	X_1: [-0.1882, 0.5530, 1.6267, 0.7013]
are	4	X_2: [0.2293, 1.3255, 0.1318, 2.0501]
learning	7	X_3: [0.4309, -1.3067, -0.8823, 1.5977]
AI	2	X_4: [1.0281, -1.9094, 0.3182, 0.4211]
<pad>	1	X_5: [1.7840, -0.8278, -0.2701, 1.3586]

How to deal with this input?

- ❖ **Simplest idea: Based on MLP**
- ❖ **Concatenate all the features**

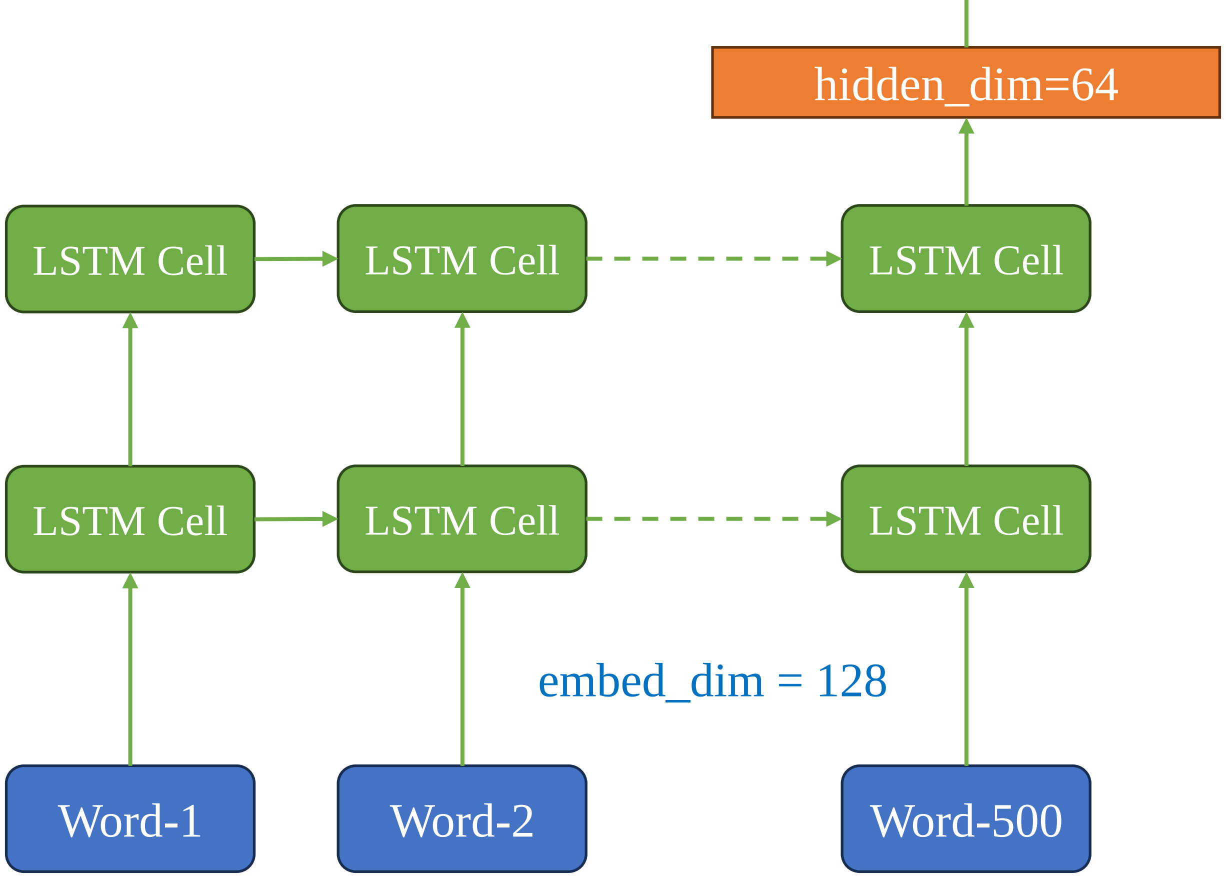


Using RNN

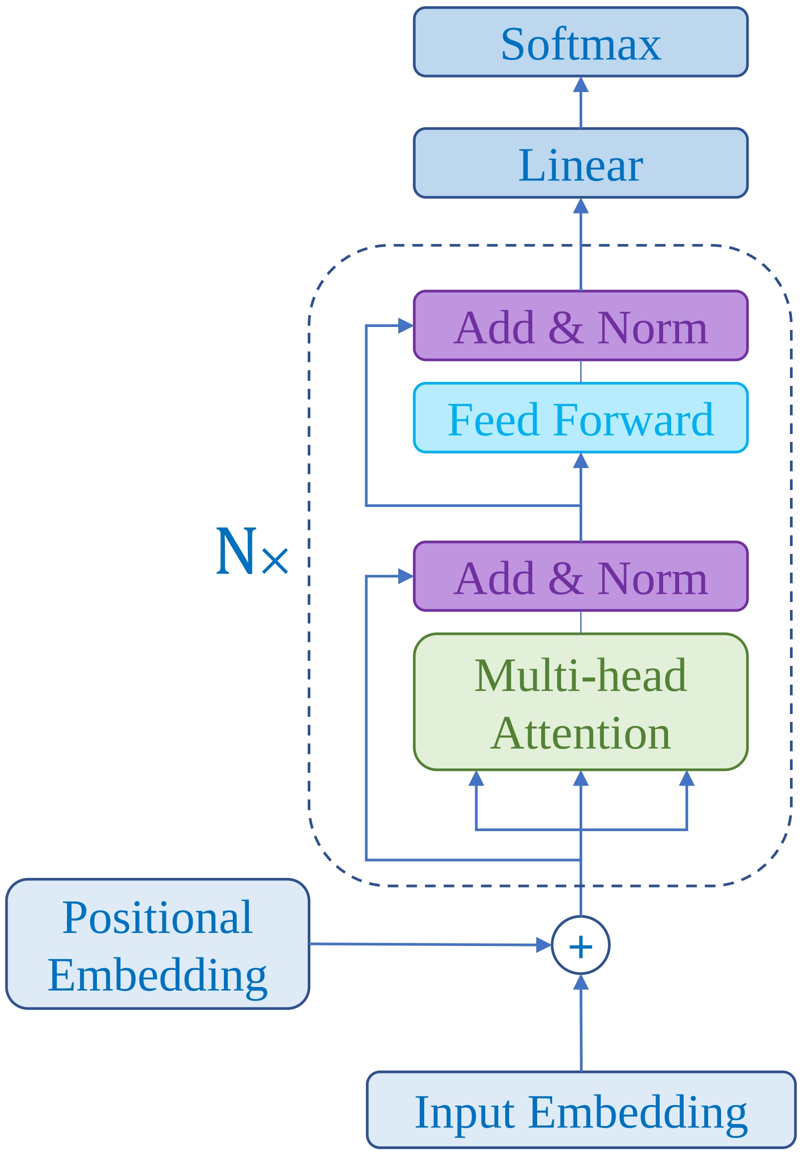



```
==================================================================
Layer (type:depth-idx)                    Output Shape
==================================================================
├─Embedding: 1-1                          [-1, 500, 128]
├─RNN: 1-2                                [-1, 500, 64]
├─Linear: 1-3                             [-1, 2]
==================================================================
```

Using LSTM

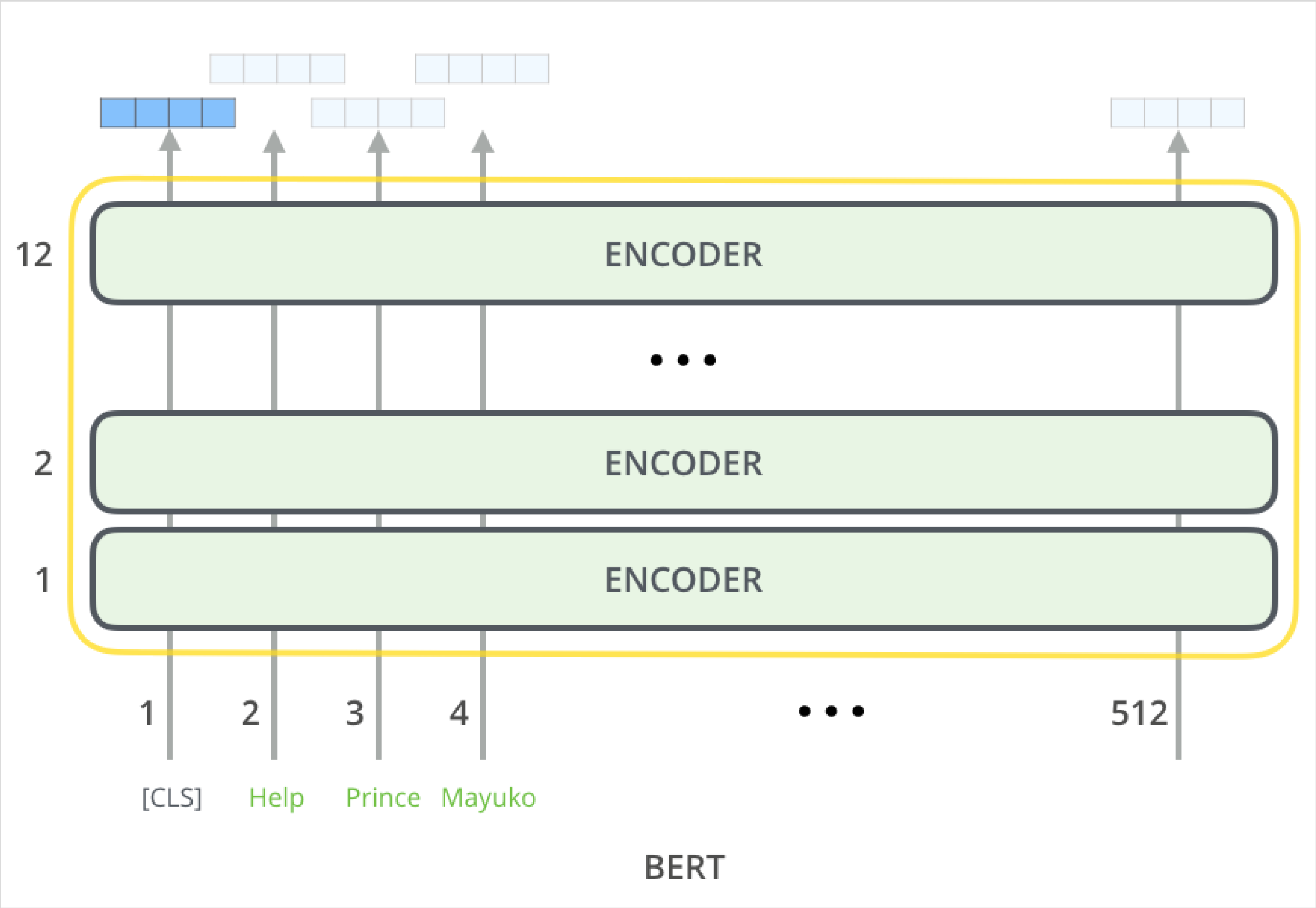


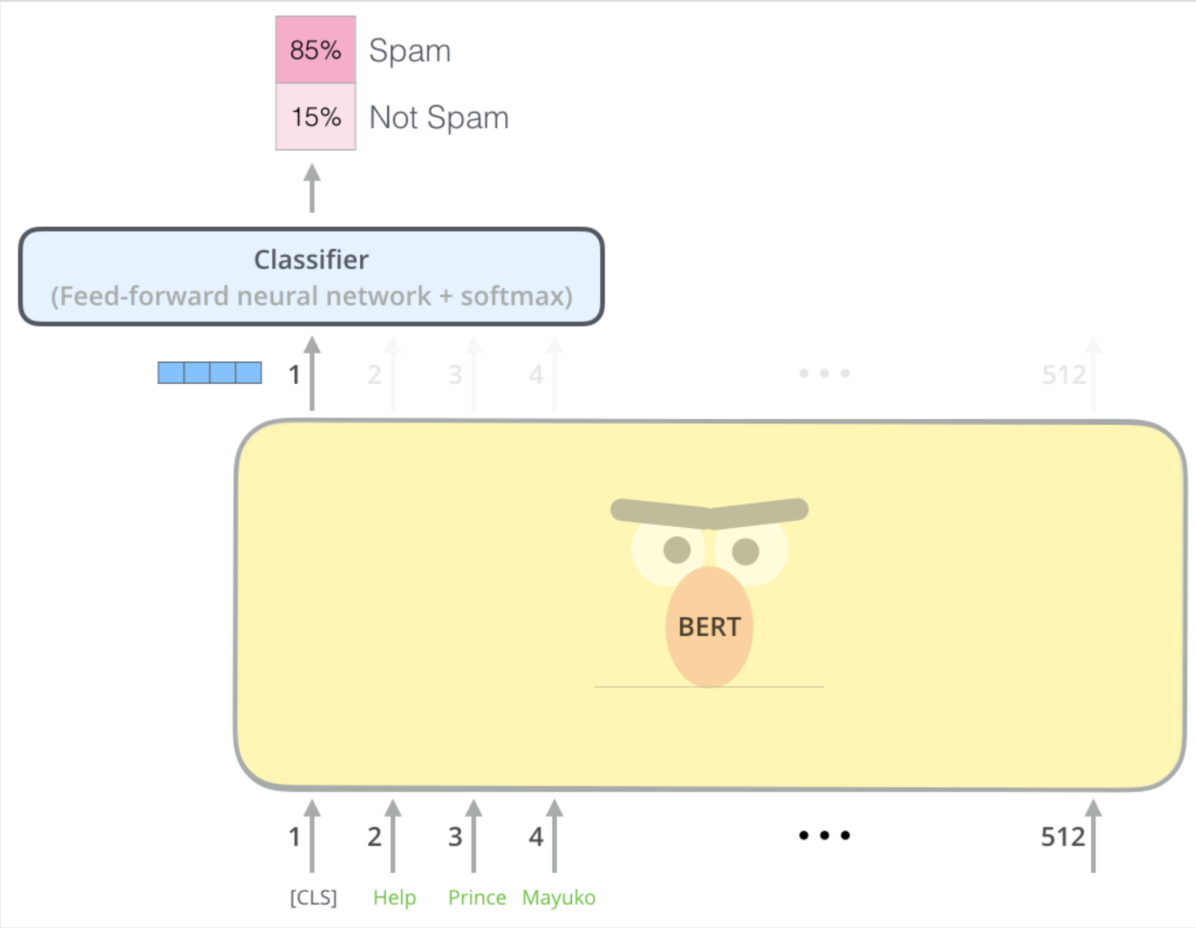
Transformer Models for Text Classification




```python
class TransformerTextCls(nn.Module):
    def __init__(self, vocab_size,
                 max_length, embed_dim,
                 num_heads, ff_dim,
                 dropout, device):
        super().__init__()
        self.embd_layer = TokenAndPositionEmbedding(vocab_size,
                                                    embed_dim,
                                                    max_length)
        self.transformer_layer = TransformerBlock(embed_dim,
                                                  num_heads,
                                                  ff_dim)
        self.pooling = nn.AvgPool1d(kernel_size=max_length)
        self.fc = nn.Linear(in_features=embed_dim,
                            out_features=2)
        self.relu = nn.ReLU()

    def forward(self, x):
        output = self.embd_layer(x)
        output = self.transformer_layer(output, output, output)
        output = self.pooling(output.permute(0,2,1)).squeeze()
        output = self.fc(output)

        return output
```

Bidirectional Encoder Representations from Transformers





Figures from the BERT paper

Outline

- What We Have from Text Classification?
- Introduction to POS Tagging
- POS Tagging Using Classification
- Using Different Model Archectures
- PyTorch Implementation

Conll2003 Dataset for Part-of-Speed Tagging

Num_classes = 47

Train	Val	Test
14041	3250	3453

0	“	Quotation mask
1		space
2	#	Hash
3	$	Dolla
4	(	Opening parenthesis
5	)	Closing parenthesis
6	,	Comma
7	.	Dot
8	:	Colon
9	``	Apostrophe

10	CC	Coordinating conjunction
11	CD	Cardinal number
12	DT	Determiner
13	EX	Existential *there*
14	FW	Foreign word
15	IN	Preposition or subordinating conjunction
16	JJ	Adjective
17	JJR	Adjective, comparative
18	JJS	Adjective, superlative
19	LS	List item marker

20	MD	Modal
21	NN	Noun, singular or mass
22	NNP	Proper noun, singular
23	NNPS	Proper noun, plural
24	NNS	Noun, plural
25	NN|SYM	Noun or Symbol
26	PDT	Predeterminer
27	POS	Possessive ending
28	PRP	Personal pronoun
29	PRP$	Possessive pronoun

Conll2003 Dataset for Part-of-Speed Tagging

Num_classes = 47

30	RB	Adverb
31	RBR	Adverb, comparative
32	RBS	Adverb, superlative
33	RP	Particle
34	SYM	Symbol
35	TO	to
36	UH	Interjection
37	VB	Verb, base form
38	VBD	Verb, past tense
39	VBG	Verb, gerund or present participle
40	VBN	Verb, past participle
41	VBP	Verb, non-3rd person singular present
42	VBZ	Verb, 3rd person singular present
43	WDT	Wh-determiner
44	WP	Wh-pronoun
45	WP$	Possessive wh-pronoun
46	WRB	Wh-adverb

Example

Input tokens

["Cup", "qualifying", "round", ",", "second", "leg", "soccer", "matches", "on", "Thursday"]

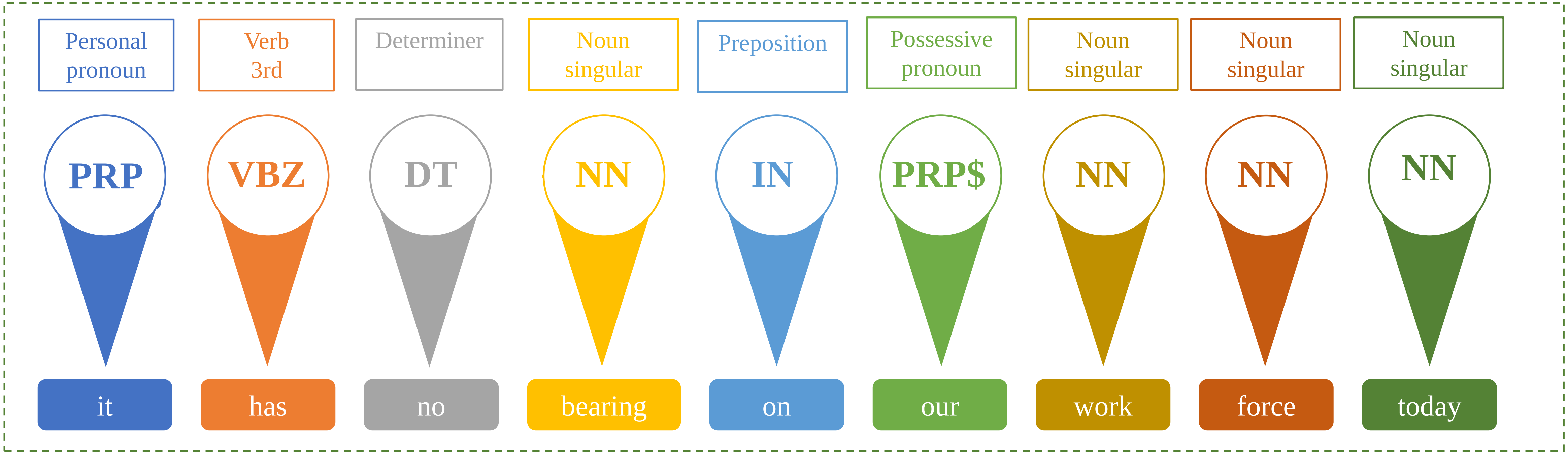
Label

[”NNP", ”VBG", ”RB", ",", ”JJ", ”NN", ”NN", ”NNS", ”IN", ”NNP"]

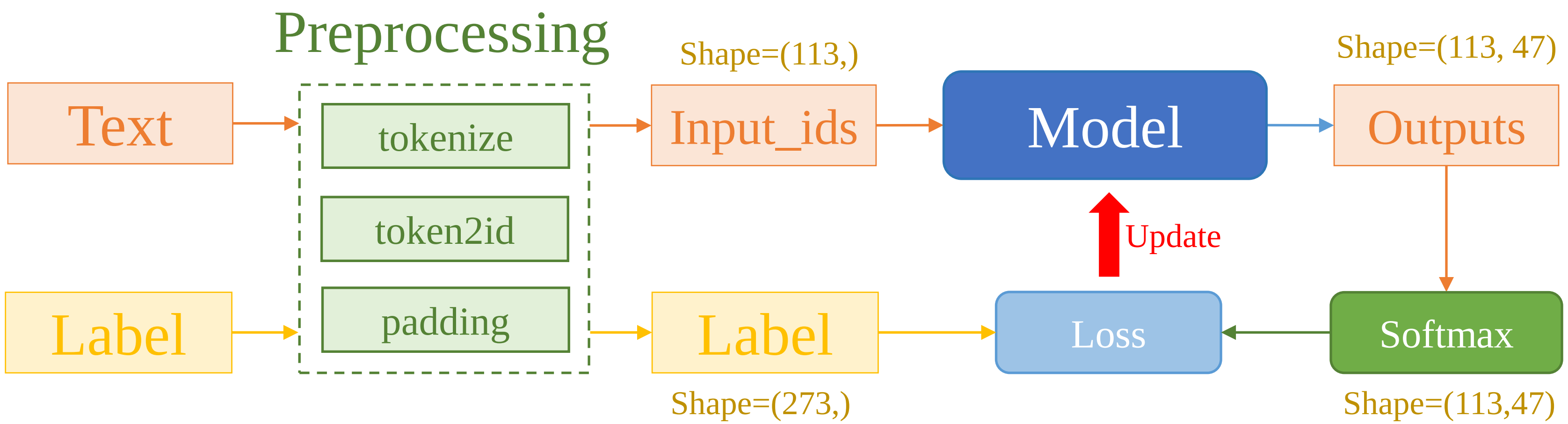
Label-encoded

[22, 39, 30, 6, 16, 21, 21, 24, 15, 22,]

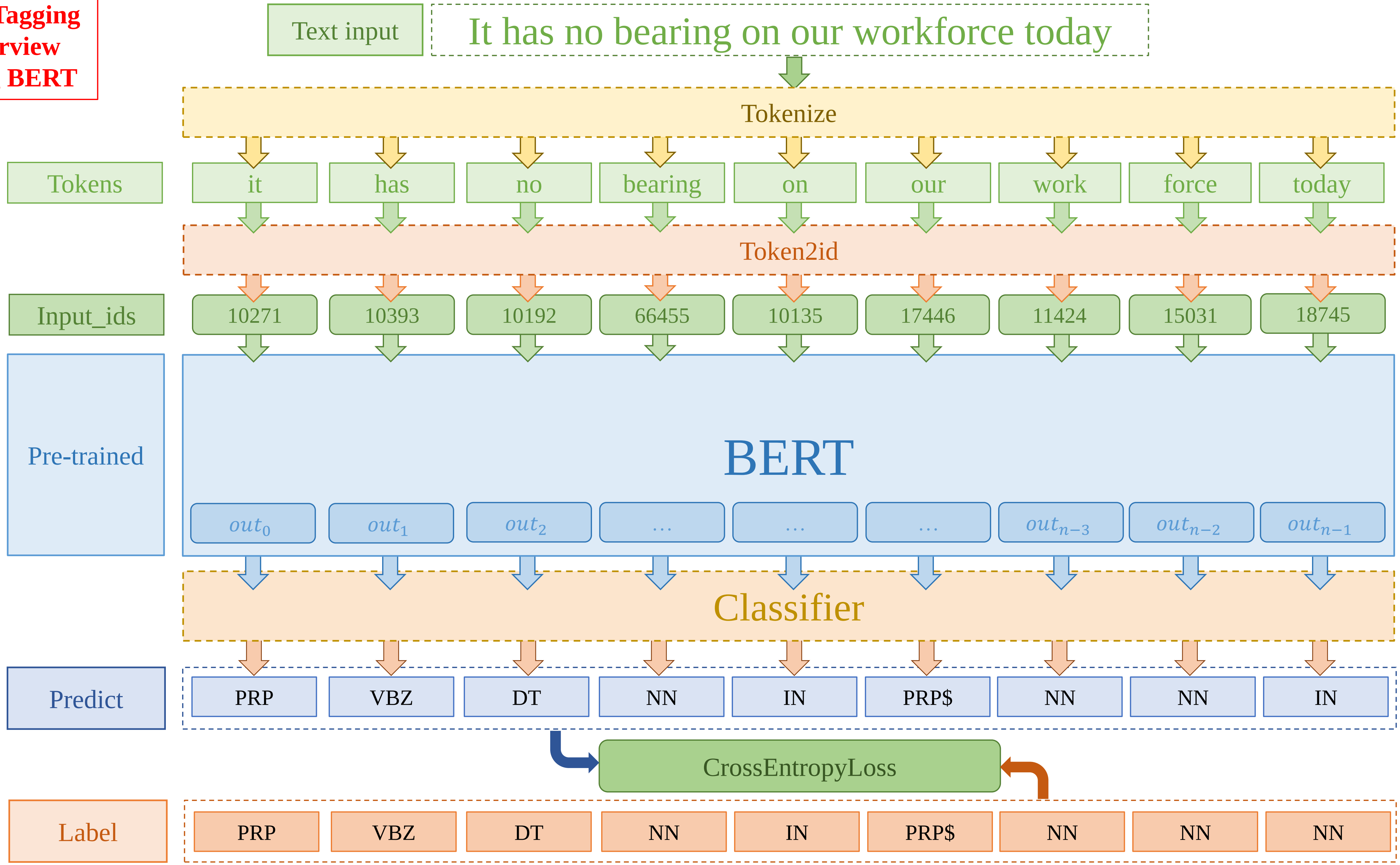
Part-of-speed Tagging



Index	Label
0	<unk>
1	NN
2	IN
3	NNP
…	…
43	LS
44	FW
45	UH
46	SYM



POS Tagging Overview Using BERT



Outline

- What We Have from Text Classification?
- Introduction to POS Tagging
- POS Tagging Using Classification
- Using Different Model Archectures
- PyTorch Implementation

Step-by-Step Example: Text Classification

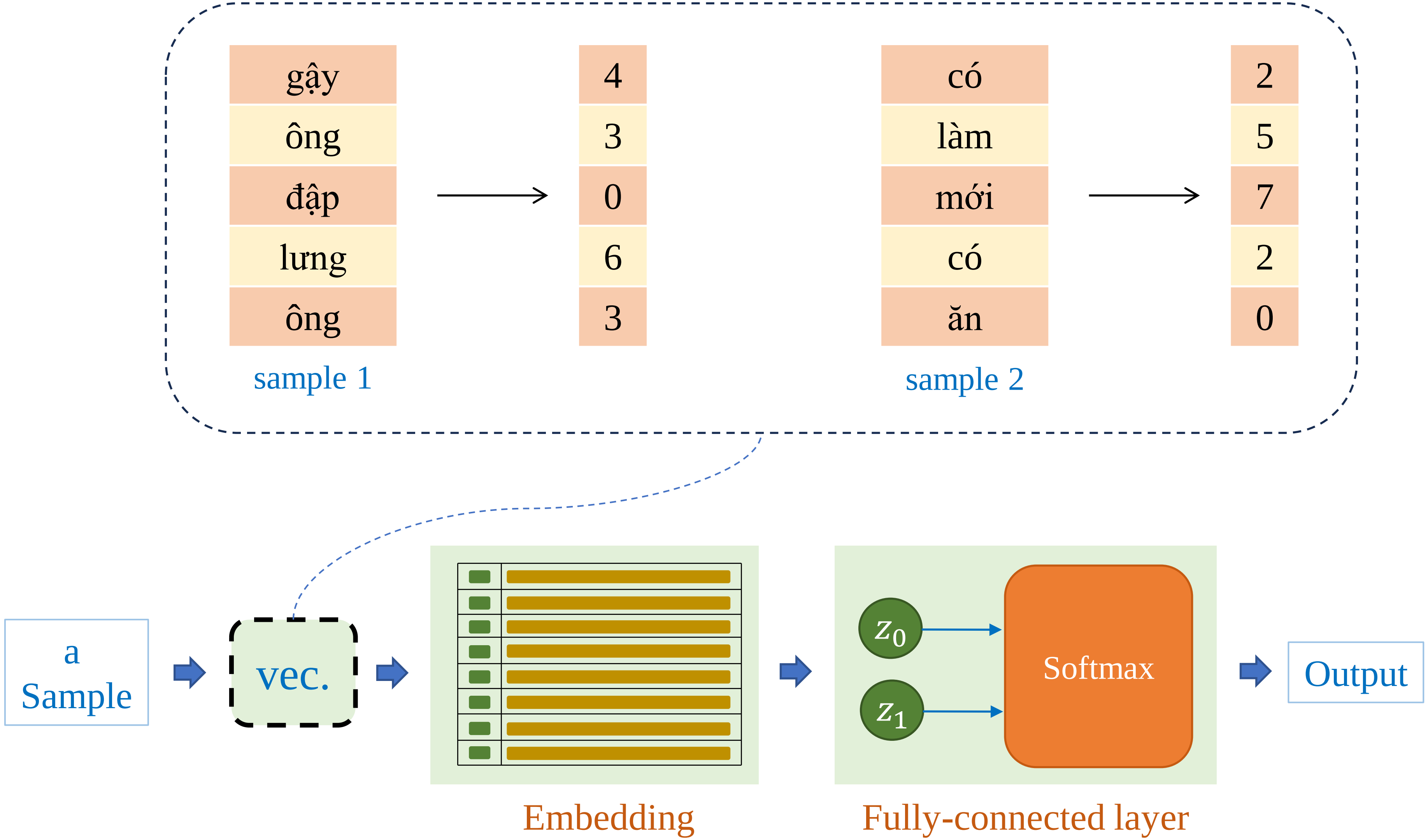
Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

Training data
- negative (0)
- positive (1)

building dictionary

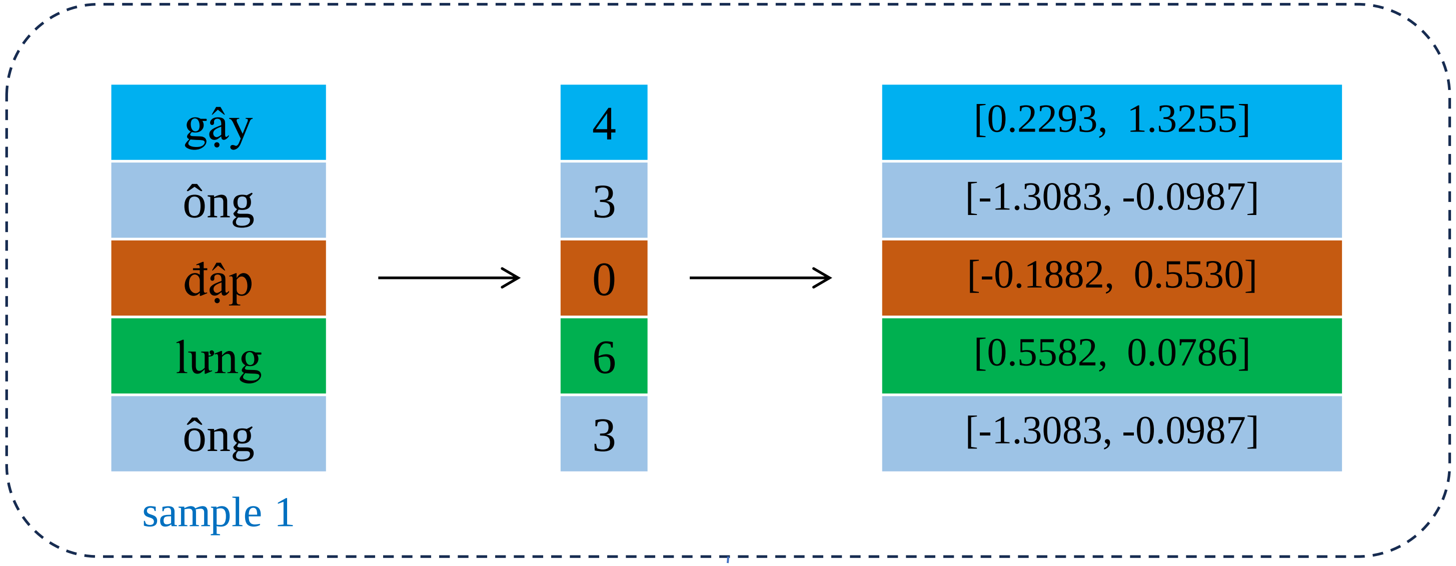
index	word
0	[UNK]
1	[pad]
2	có
3	ông
4	gập
5	làm
6	lưng
7	mới

vocab size = 8
sequence length = 5



Step-by-Step Example: Text Classification

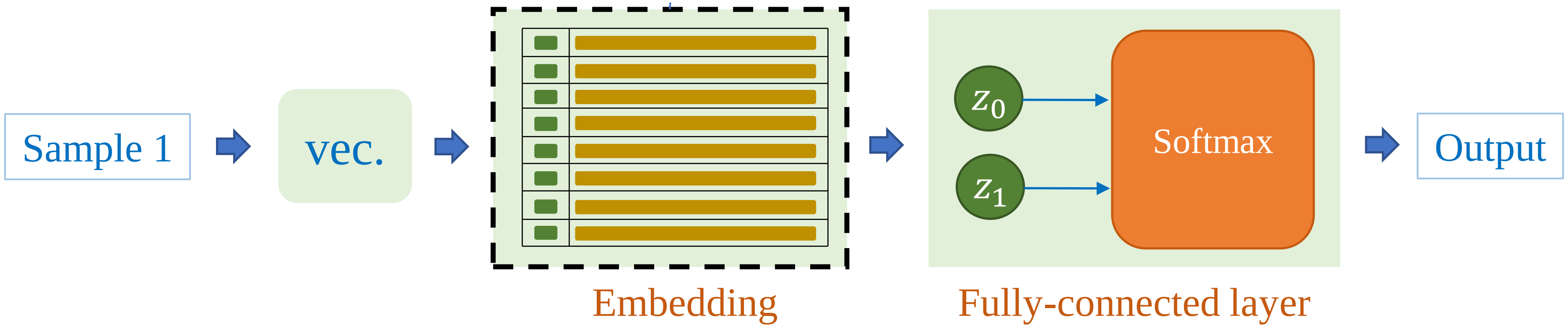
Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1



0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding 8x2
(Random initialization)

parameter

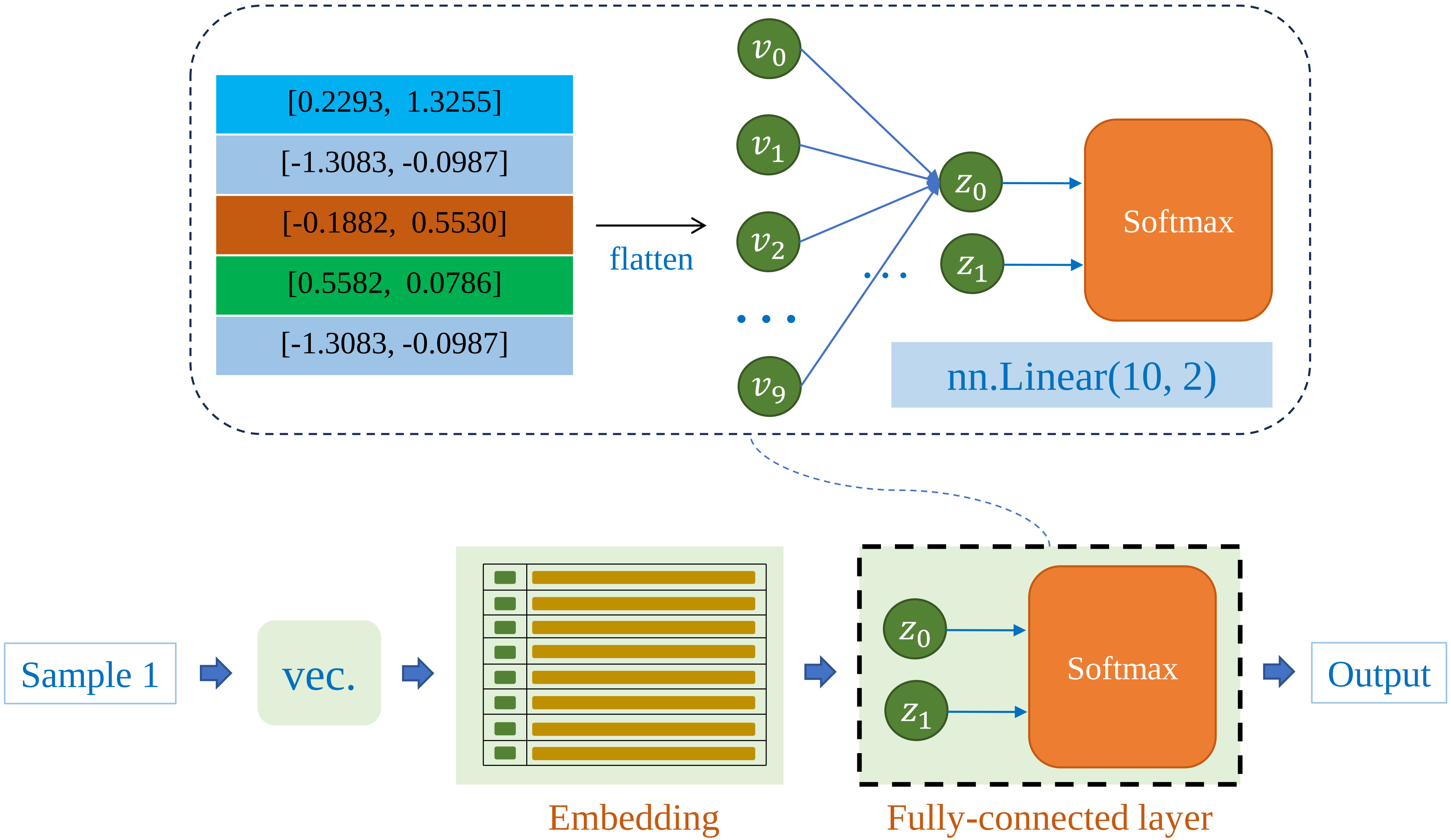


Step-by-Step Example: Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding



Example
Text Classification

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982]

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985]

b [-0.3049, 0.1028]

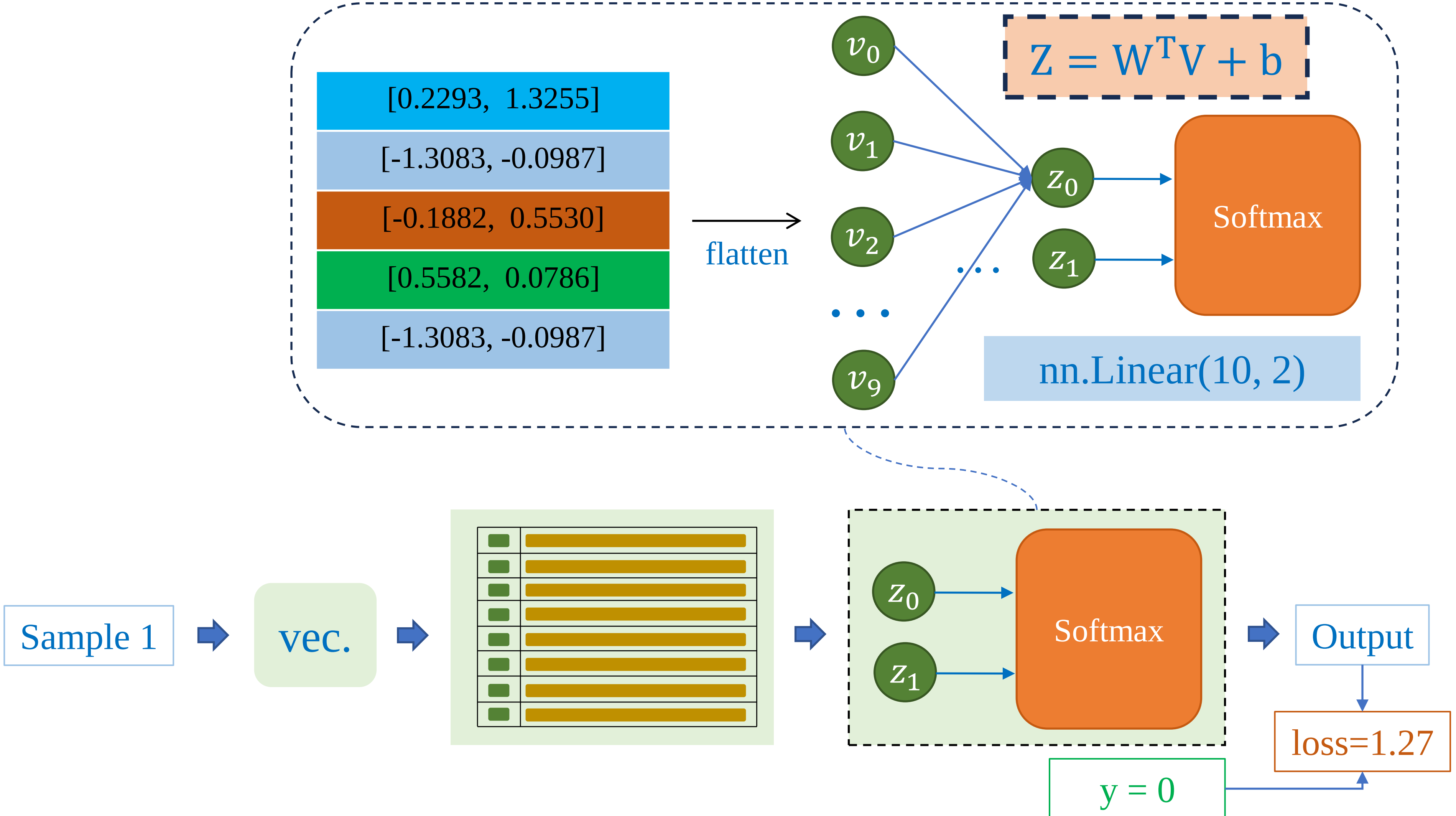
z [-0.7875, 0.1221]

$\hat{y}$ [0.28, 0.71]

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

0	[-0.1882, 0.5530]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.3083, -0.0987]
4	[0.2293, 1.3255]
5	[0.4058, -0.6624]
6	[0.5582, 0.0786]
7	[0.4309, -1.3067]

Embedding



Example
Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

W

[0.3108, 0.0926, 0.1760, 0.1325, -0.1518, -0.0876, 0.1194, 0.1378, -0.0790, 0.1982]

[-0.0716, 0.1968, 0.0740, 0.2251, 0.0718, -0.0825, -0.2817, 0.1483, 0.3338, 0.3985]

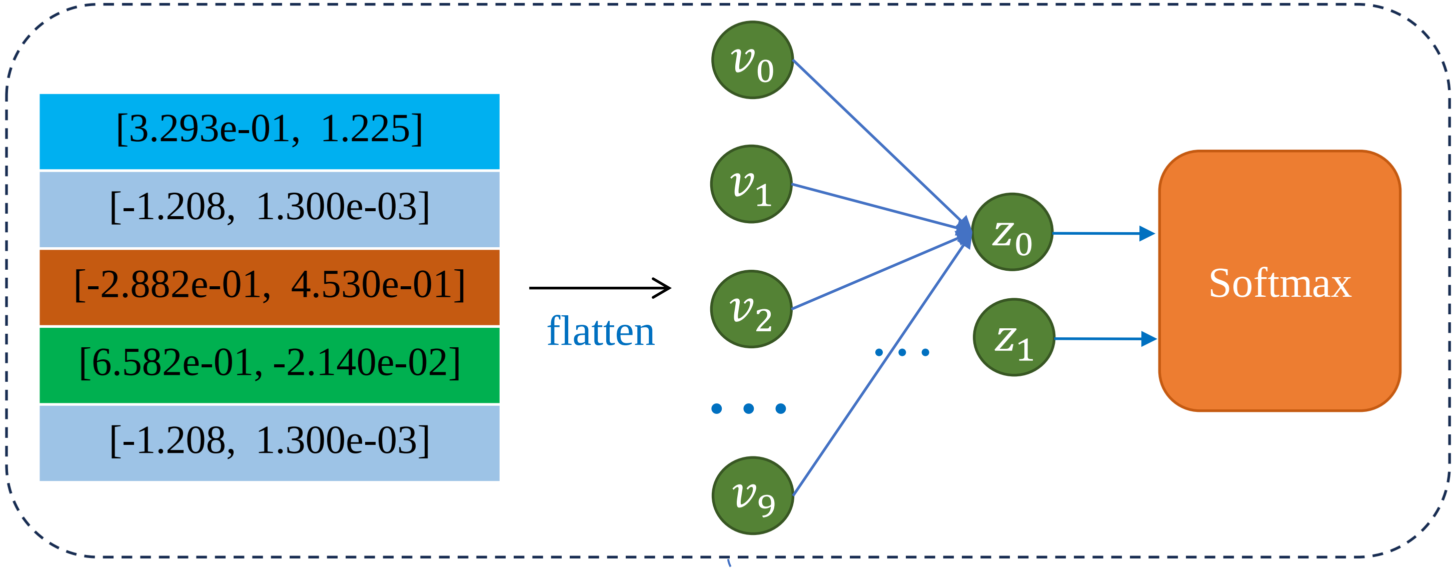
```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

b

[-0.2049, 0.0028]

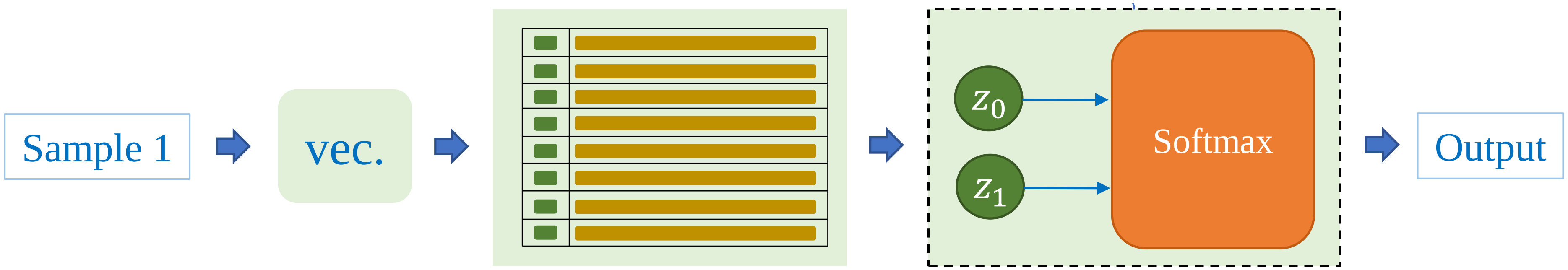
$$\theta_t = \theta_{t-1} - \frac{\dots \nabla_\theta L}{\dots}$$

update



0	[-2.882e-01, 4.530e-01]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.208, 1.300e-03]
4	[3.293e-01, 1.225]
5	[0.4058, -0.6624]
6	[6.582e-01, -2.140e-02]
7	[0.4309, -1.3067]

Embedding



5

Example
Text Classification

Doc	Label
gậy ông đập lưng ông	0
có làm mới có ăn	1

W

[0.3108, 0.0926, 0.1760, 0.1325, -0.1518, -0.0876, 0.1194, 0.1378, -0.0790, 0.1982]

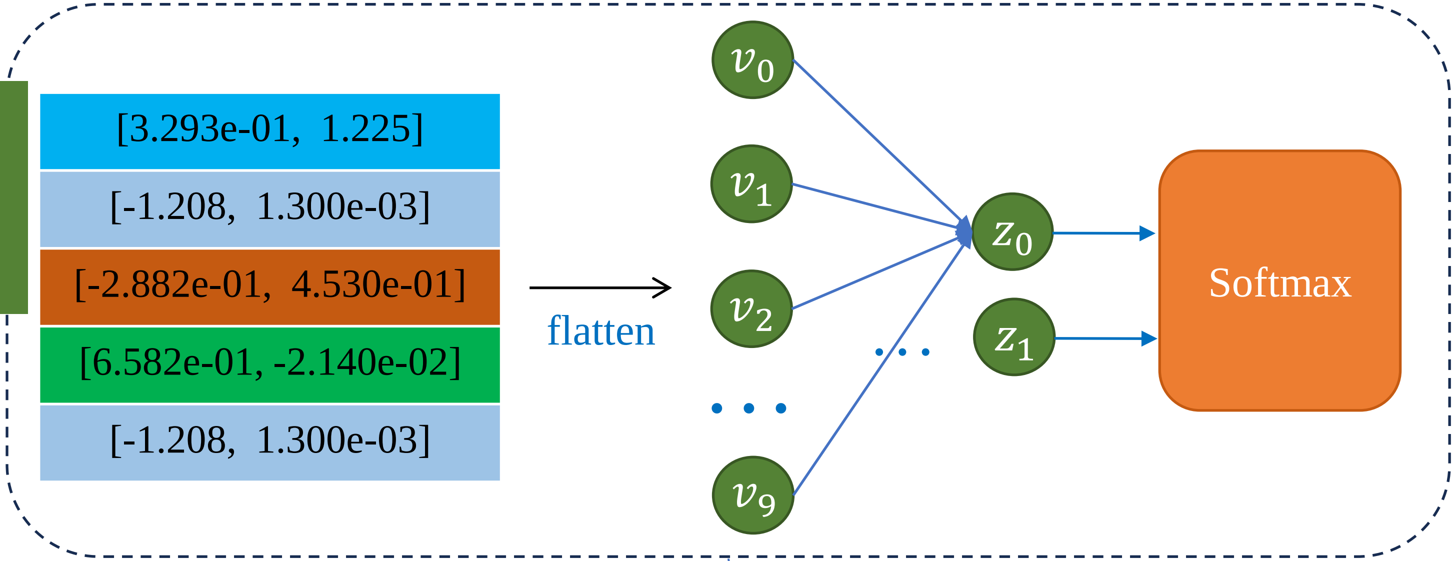
[-0.0716, 0.1968, 0.0740, 0.2251, 0.0718, -0.0825, -0.2817, 0.1483, 0.3338, 0.3985]

b [-0.2049, 0.0028]

z [-0.0261, -0.5182]

$\hat{y}$ [0.63, 0.37]

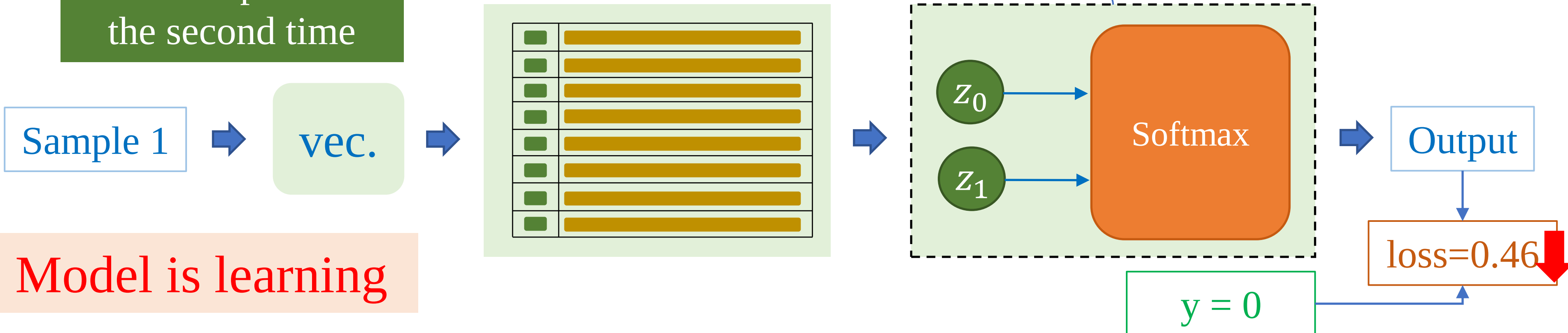
- Loss reduces
- $\hat{y}_0$ increases
- $\hat{y}_1$ reduces



0	[-2.882e-01, 4.530e-01]
1	[1.7840, -0.8278]
2	[1.0281, -1.9094]
3	[-1.208, 1.300e-03]
4	[3.293e-01, 1.225]
5	[0.4058, -0.6624]
6	[6.582e-01, -2.140e-02]
7	[0.4309, -1.3067]

Embedding

Feed sample 1 for the second time



Model is learning

POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

building dictionary

index	word
0	a
1	are
2	books
3	dog
4	expensive
5	i
6	want

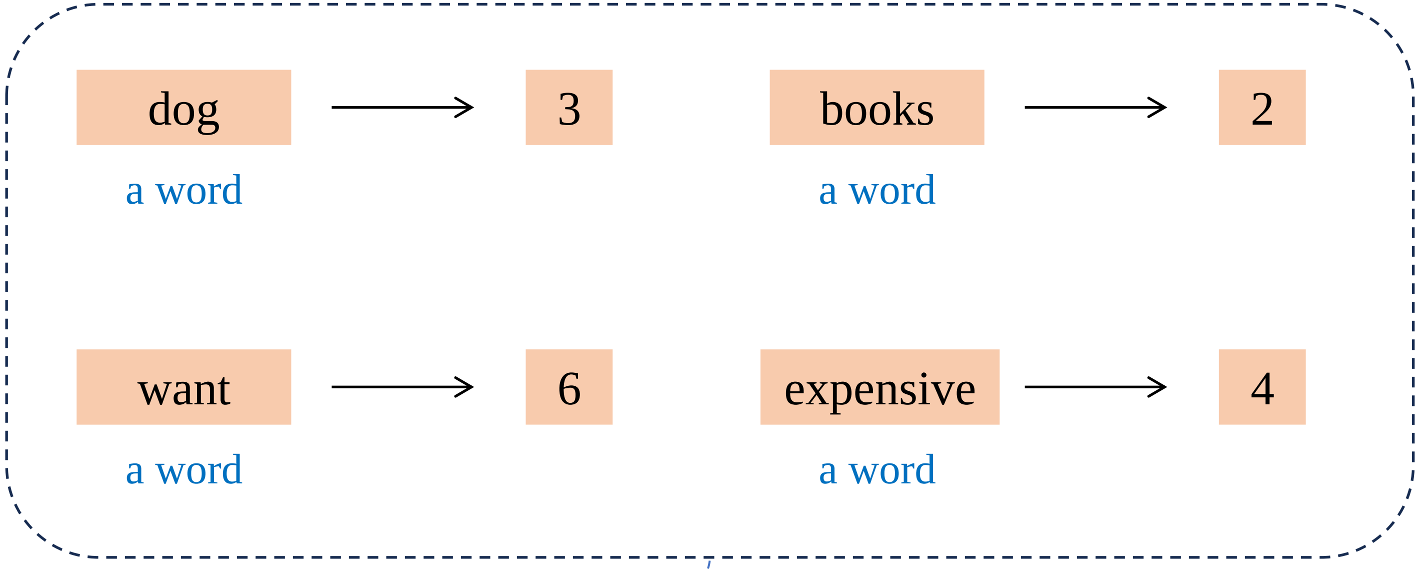
vocab size = 7
word-based classification

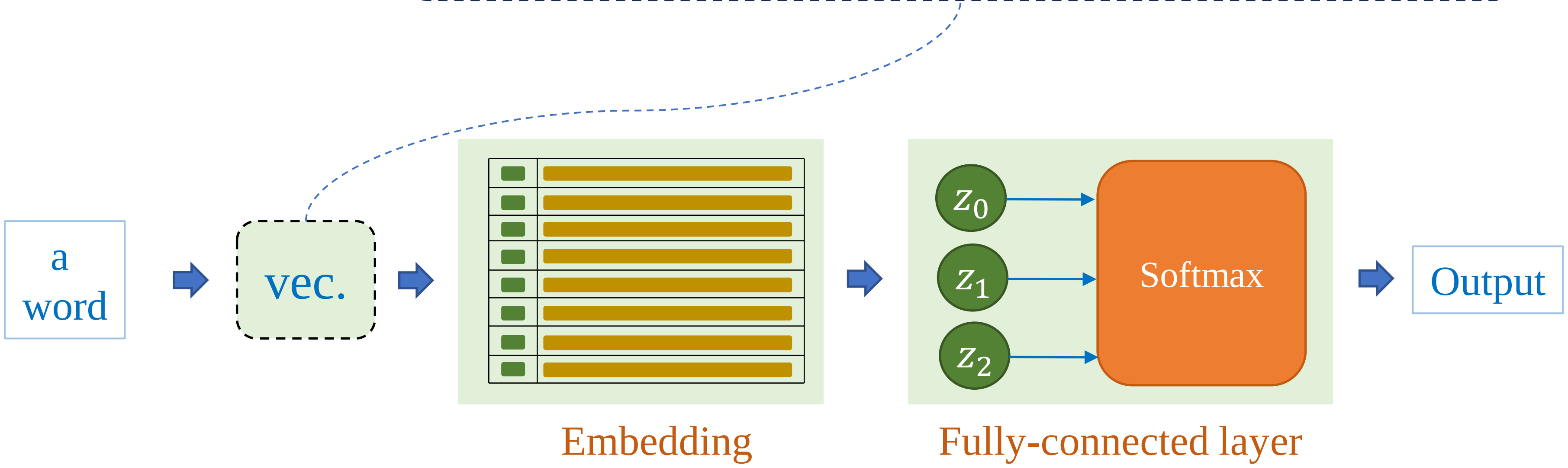
Label

0: (pro)noun

1: verb

2: others

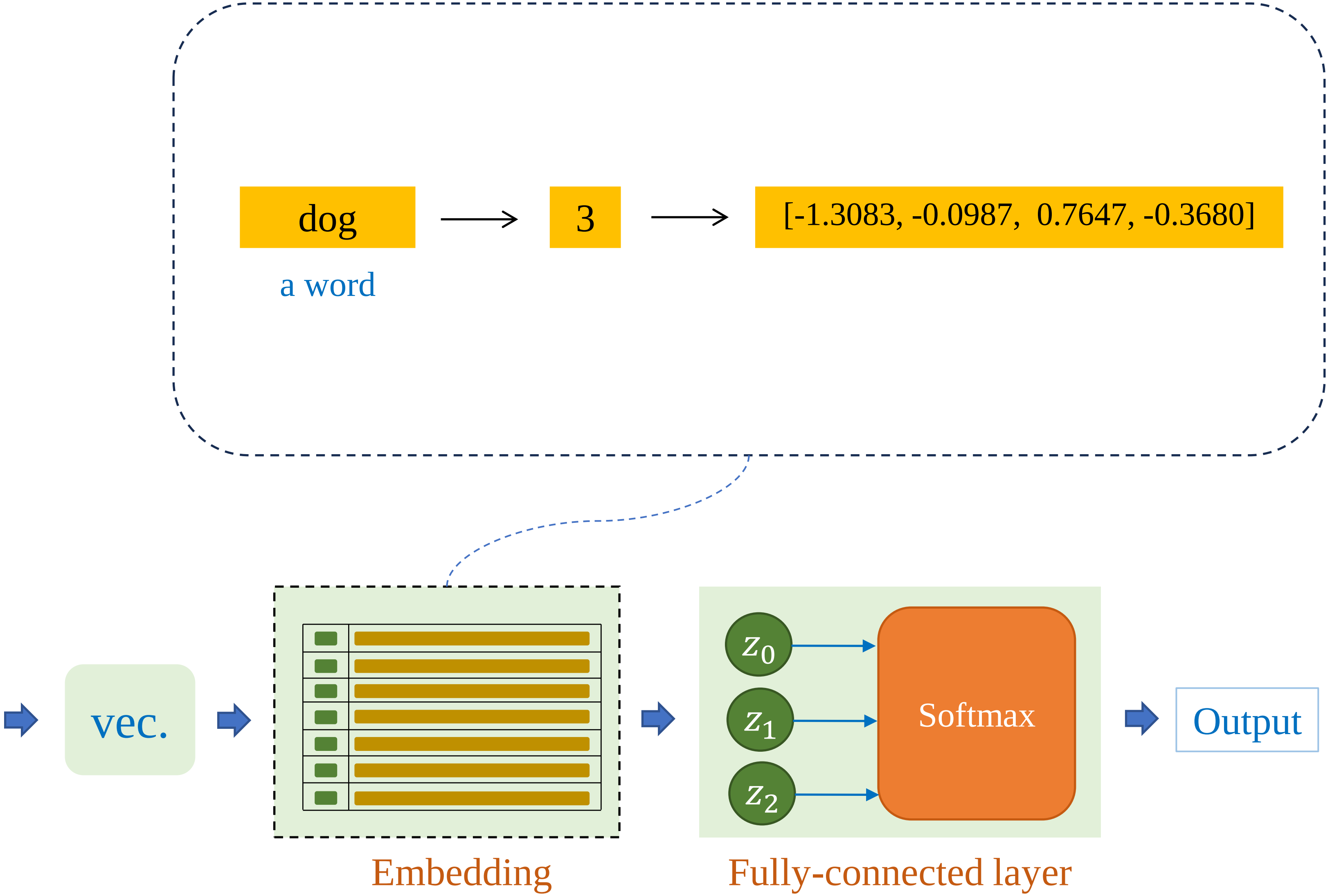




POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

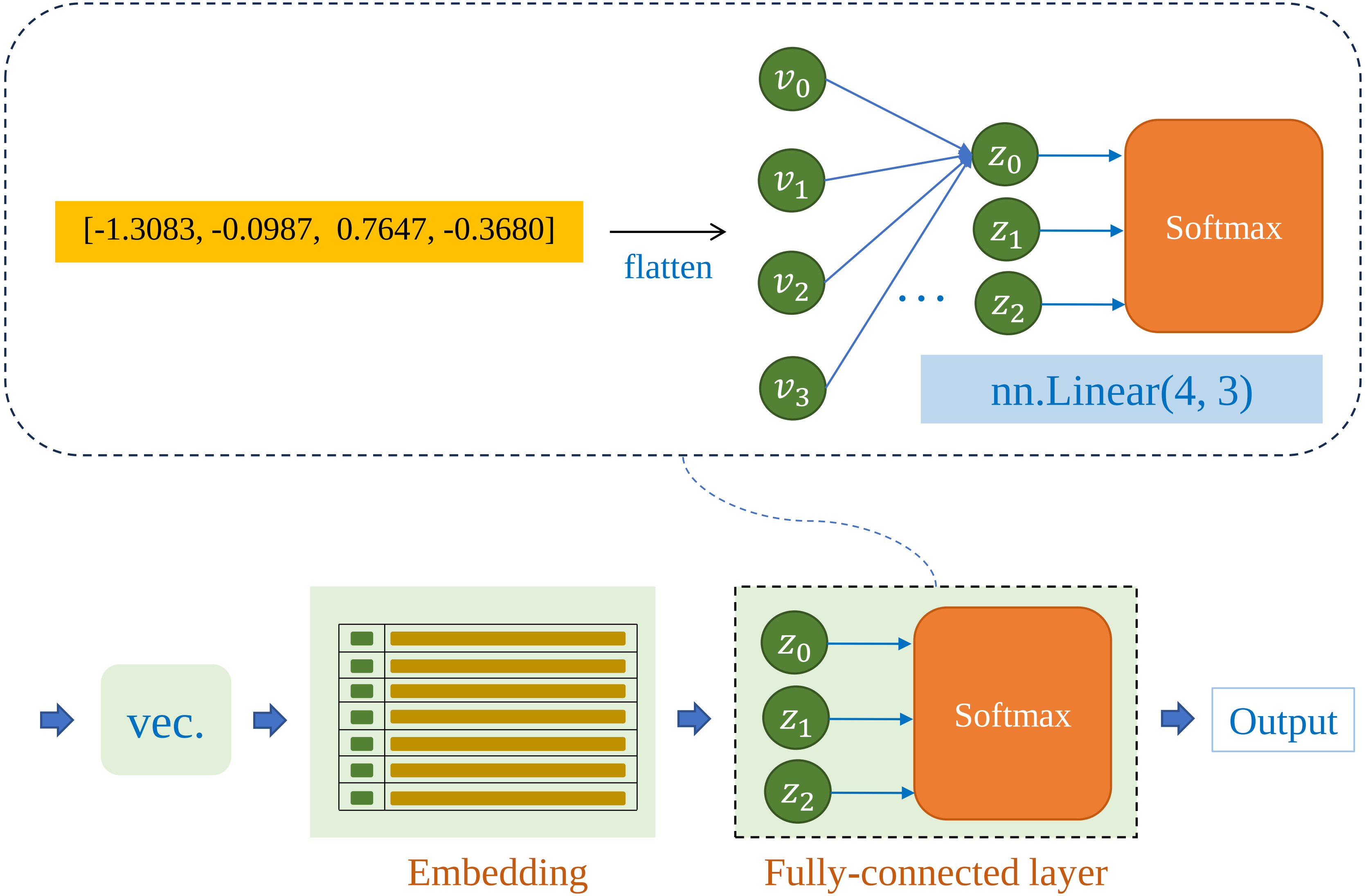
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

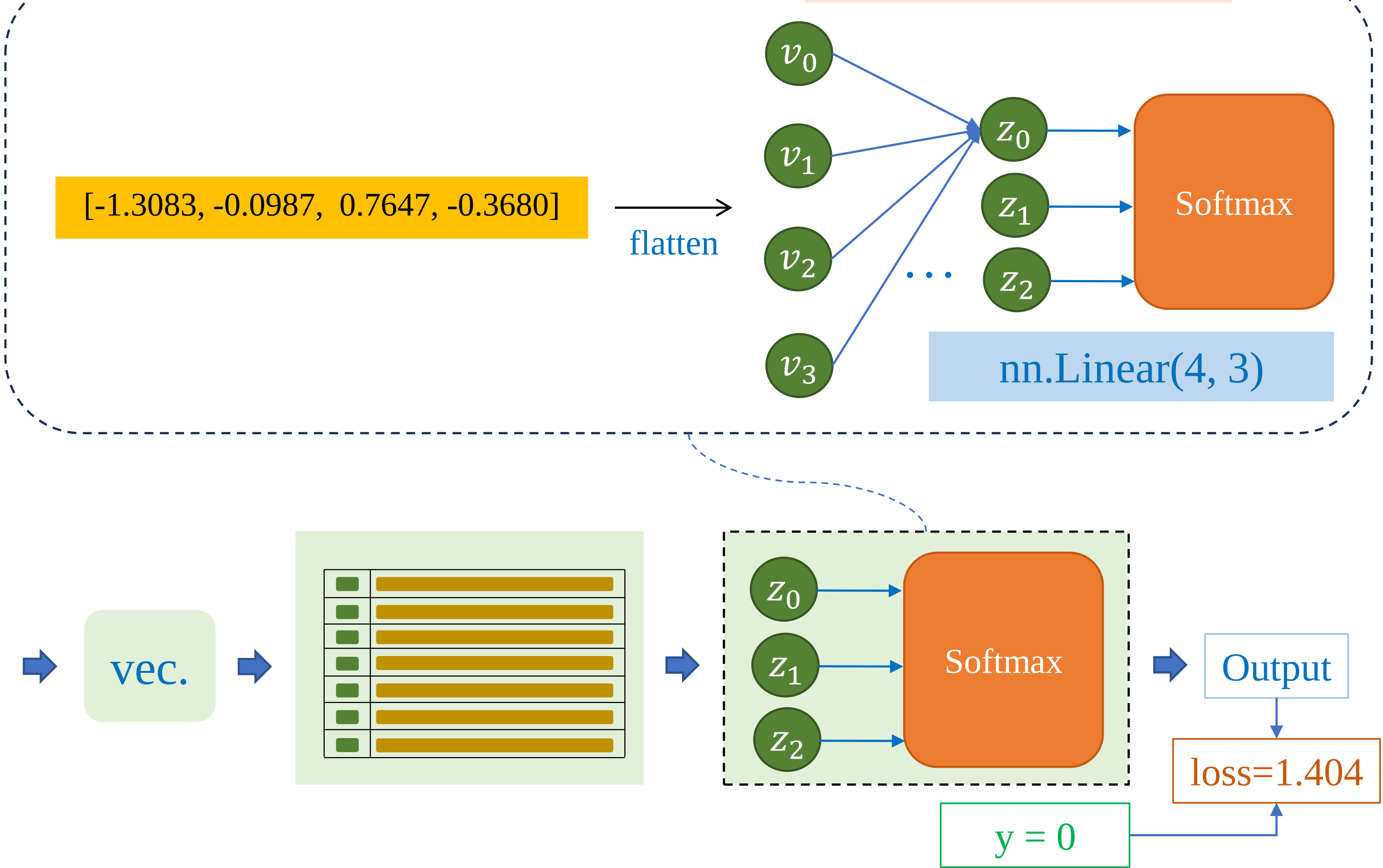
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]

W
[0.3847, -0.4621, 0.1749, -0.0139]
[-0.3024, -0.1529, 0.4329, 0.4254]
[-0.4441, 0.4113, 0.0054, -0.3220]

b [0.3548, -0.2819, -0.0579]

z [0.0360, 0.3033, 0.6051]

$\hat{y}$ [0.24, 0.32, 0.44]



POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

W

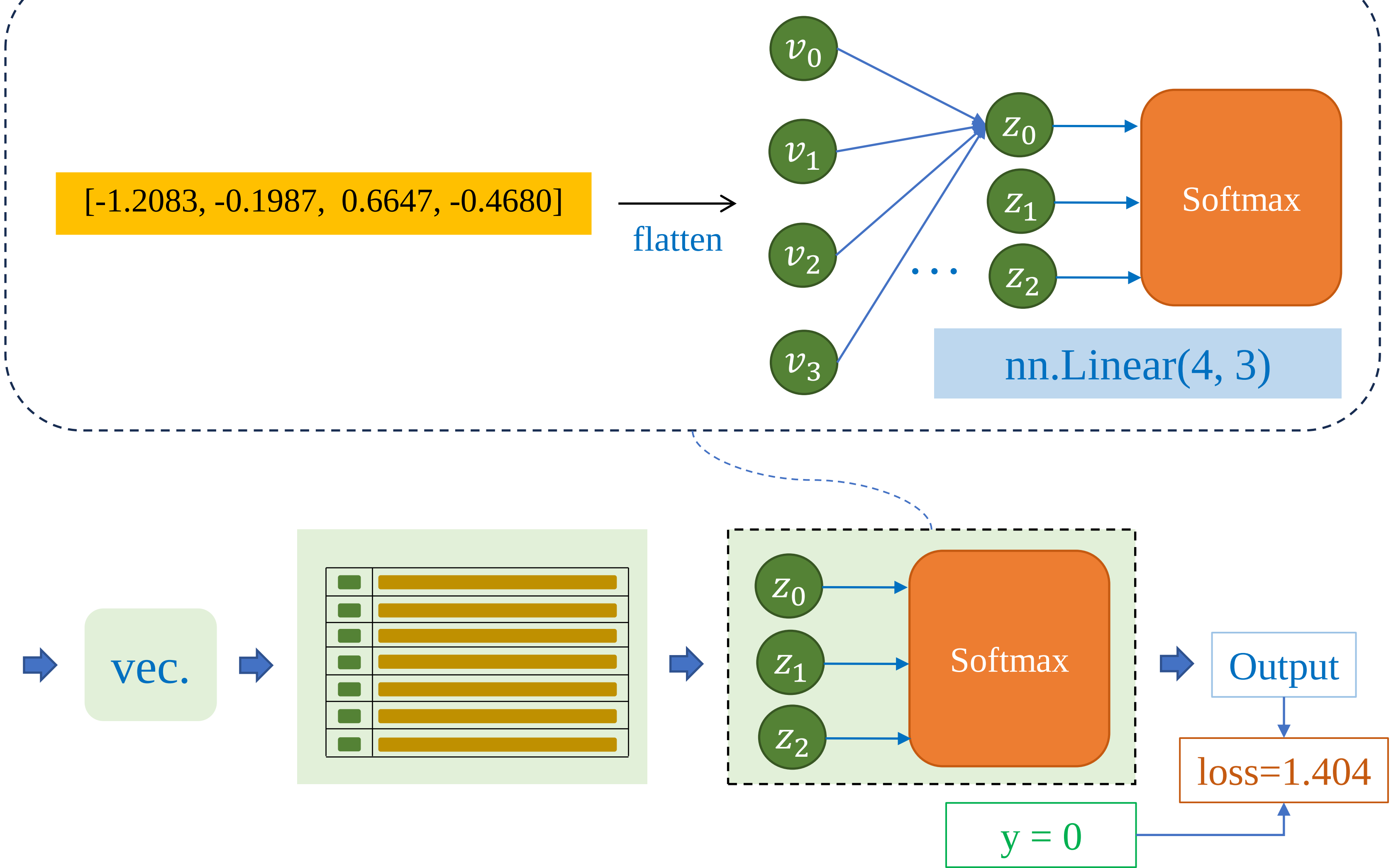
[0.2847, -0.5621, 0.2749, -0.1139]
[-0.2024, -0.0529, 0.3329, 0.5254]
[-0.3441, 0.5113, -0.0946, -0.2220]

b

[0.4548, -0.3819, -0.1579]

```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

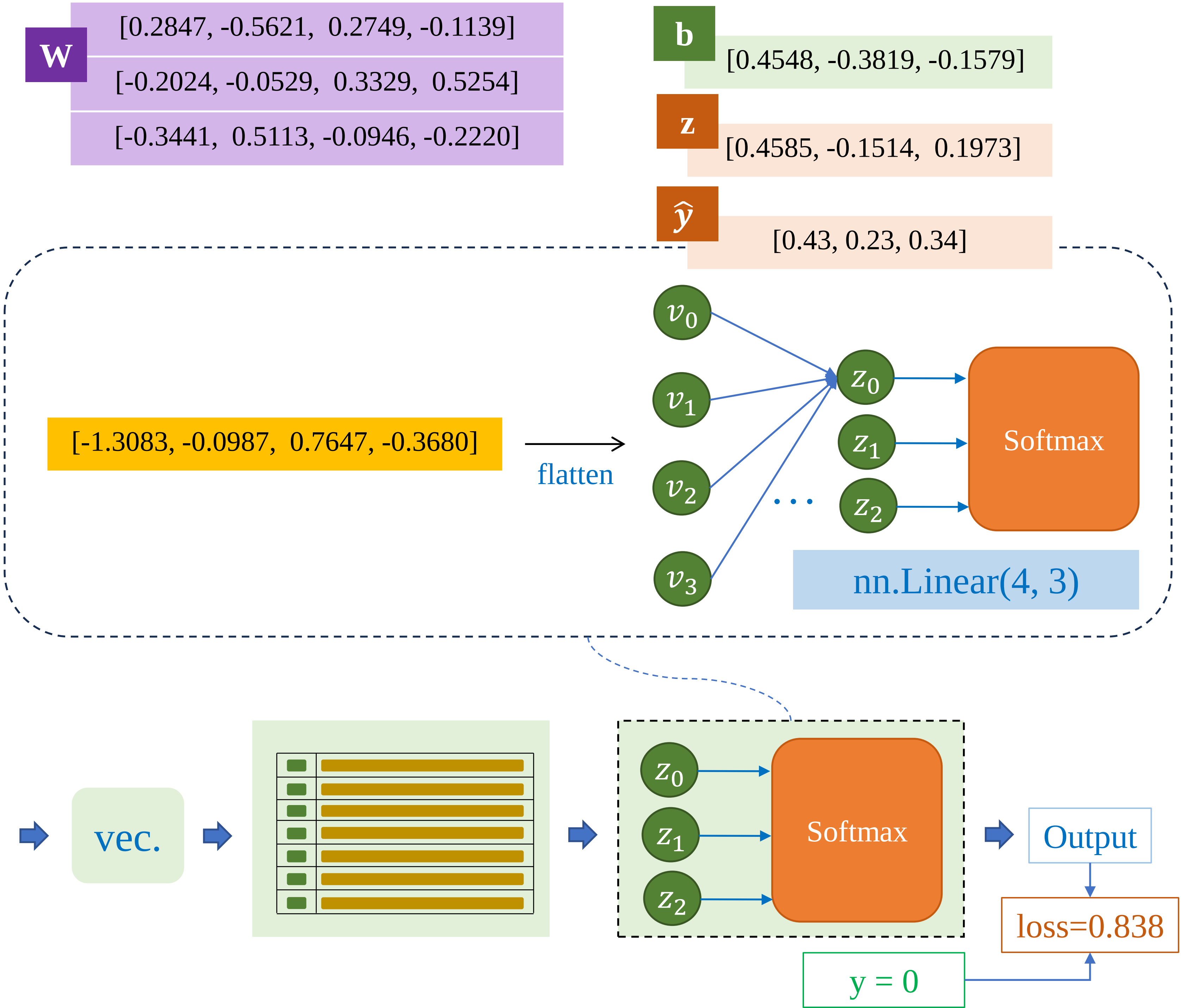
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.2083, -0.1987, 0.6647, -0.4680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



POS Tagging (1): One-Word Input

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]



Problem?

POS Tagging (2): Sentence + MLP

Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

building dictionary

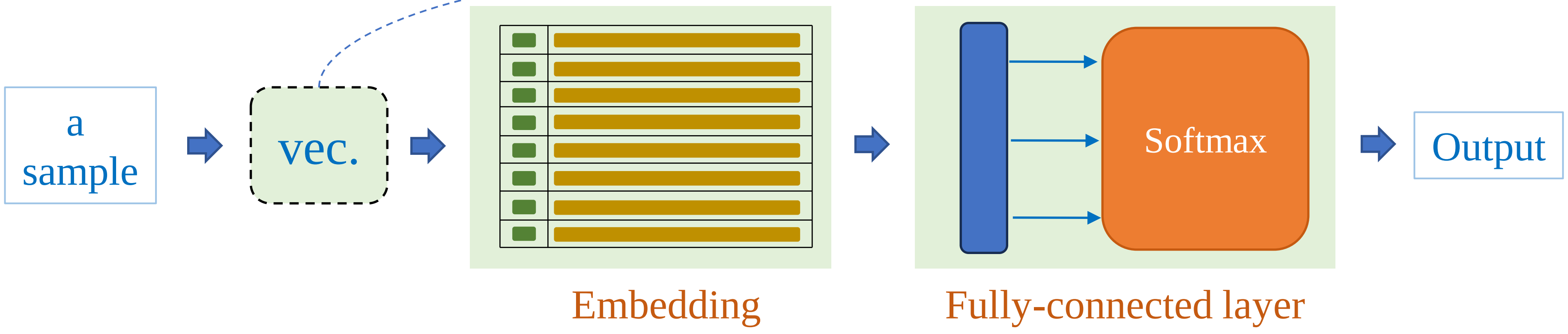
index	word
0	a
1	are
2	books
3	dog
4	expensive
5	i
6	quite
7	want

vocab size = 8
sequence length = 4

Label

0: (pro)noun
1: verb
2: others

sample 1		sample 2	
i	5	books	2
want	7	are	1
a	0	quite	6
dog	3	expensive	4



POS Tagging (2): Sentence + MLP

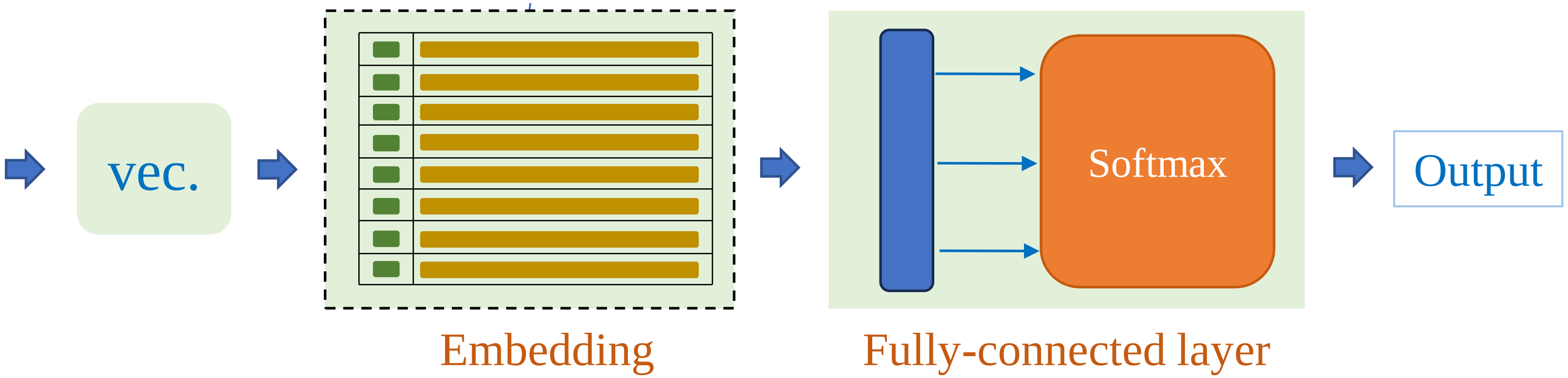
Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

i	5		[0.4058, -0.6624, -0.8745, 0.7203]
want	7		[0.4309, -1.3067, -0.8823, 1.5977]
a	0		[-0.1882, 0.5530, 1.6267, 0.7013]
dog	3		[-1.3083, -0.0987, 0.7647, -0.3680]

sample 1

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4



W	
	[0.3847, -0.4621, 0.1749, -0.0139]
	[-0.3024, -0.1529, 0.4329, 0.4254]
	[-0.4441, 0.4113, 0.0054, -0.3220]

b: [0.3548, -0.2819, -0.0579]

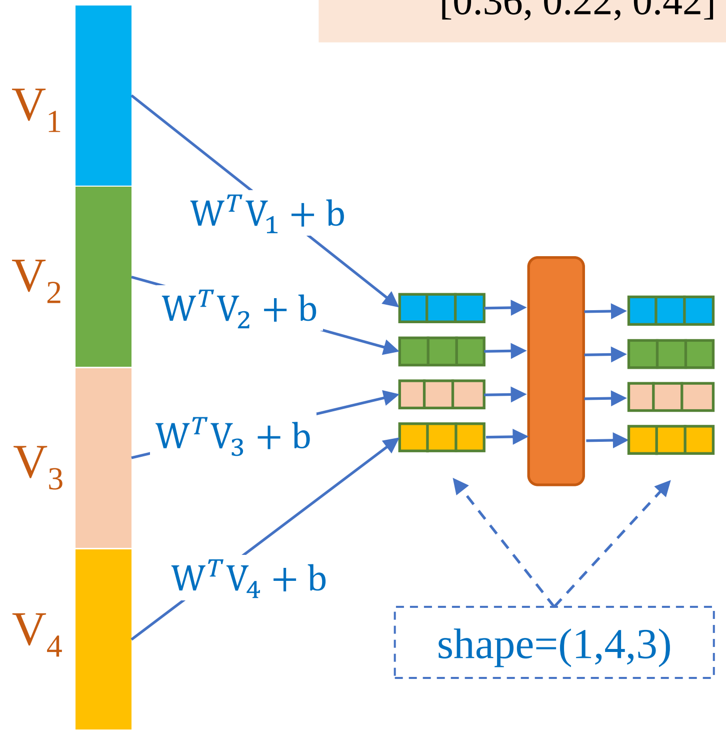
$\hat{y}$
[0.58, 0.26, 0.16]
[0.47, 0.44, 0.09]
[0.35, 0.29, 0.36]
[0.36, 0.22, 0.42]

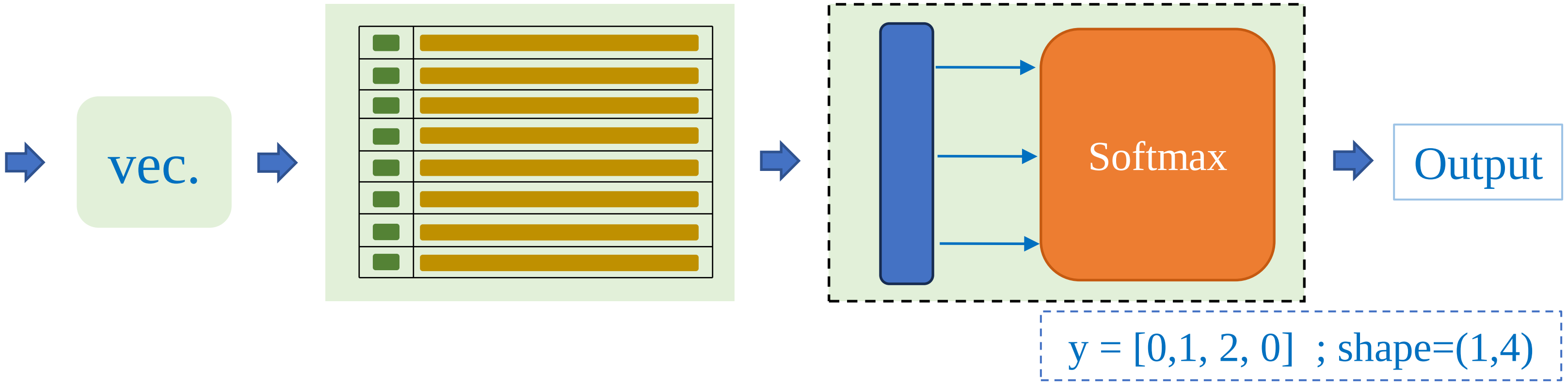
Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4

[0.4058, -0.6624, -0.8745, 0.7203]
[0.4309, -1.3067, -0.8823, 1.5977]
[-0.1882, 0.5530, 1.6267, 0.7013]
[-1.3083, -0.0987, 0.7647, -0.3680]





W	
	[0.3847, -0.4621, 0.1749, -0.0139]
	[-0.3024, -0.1529, 0.4329, 0.4254]
	[-0.4441, 0.4113, 0.0054, -0.3220]

b	
	[0.3548, -0.2819, -0.0579]

$\hat{y}$	
	[0.58, 0.47, 0.35, 0.36]
	[0.26, 0.44, 0.29, 0.22]
	[0.16, 0.22, 0.36, 0.42]

Shape of logits = (N, C, d)

Shape of target = (N, d)

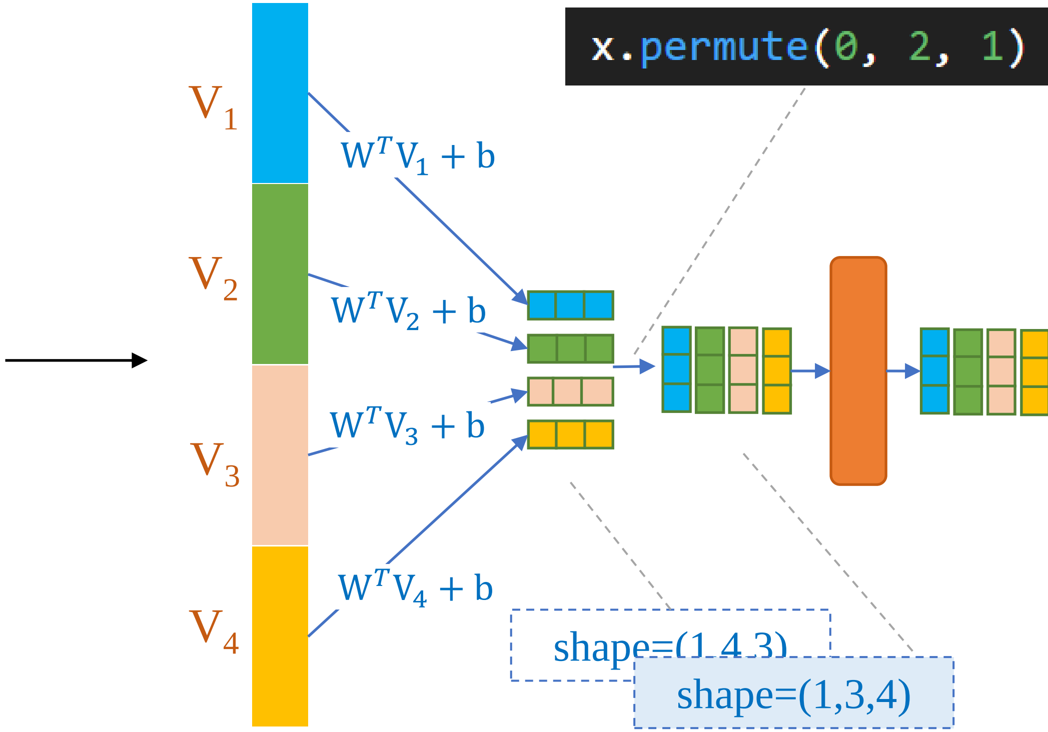
pytorch requirement

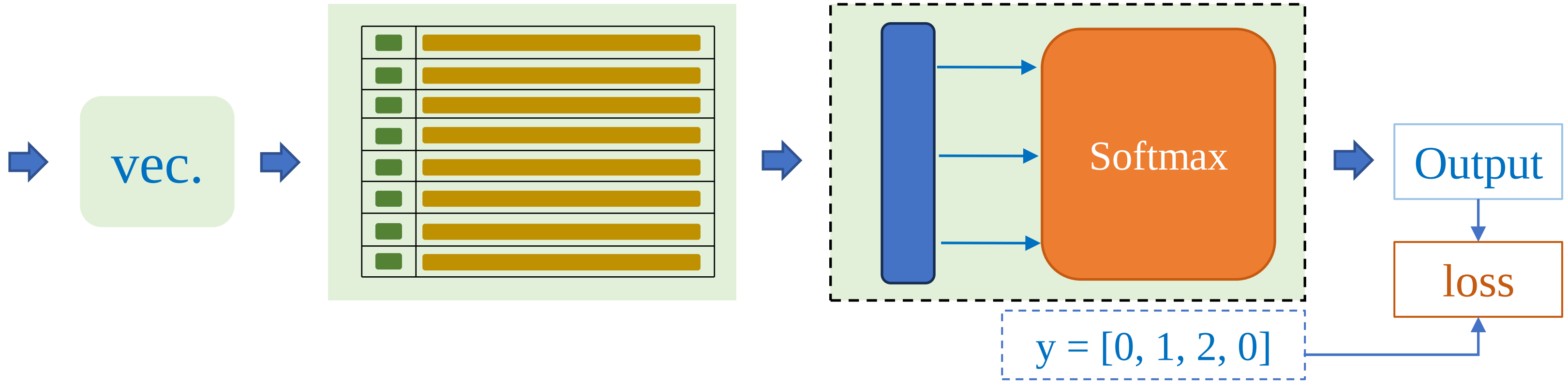
Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4

[0.4058, -0.6624, -0.8745, 0.7203]
[0.4309, -1.3067, -0.8823, 1.5977]
[-0.1882, 0.5530, 1.6267, 0.7013]
[-1.3083, -0.0987, 0.7647, -0.3680]





W

[0.3847, -0.4621, 0.1749, -0.0139]
[-0.3024, -0.1529, 0.4329, 0.4254]
[-0.4441, 0.4113, 0.0054, -0.3220]

b

[0.3548, -0.2819, -0.0579]

Doc	Label
i want a dog	[0, 1, 2, 0]
books are quite expensive	[0, 1, 2, 2]

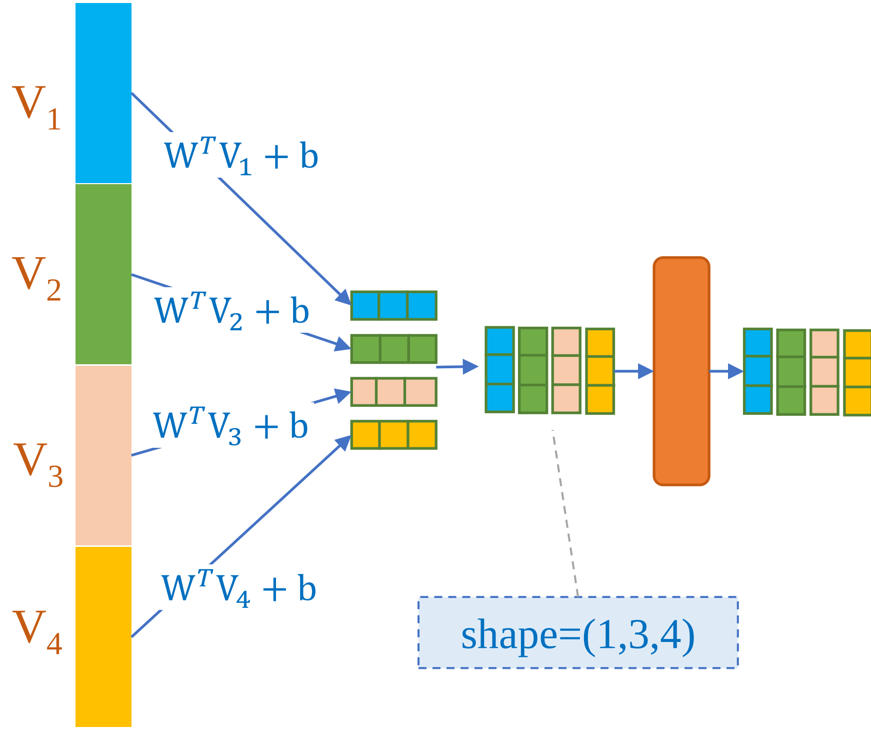
0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]

vocab size = 8
sequence length = 4

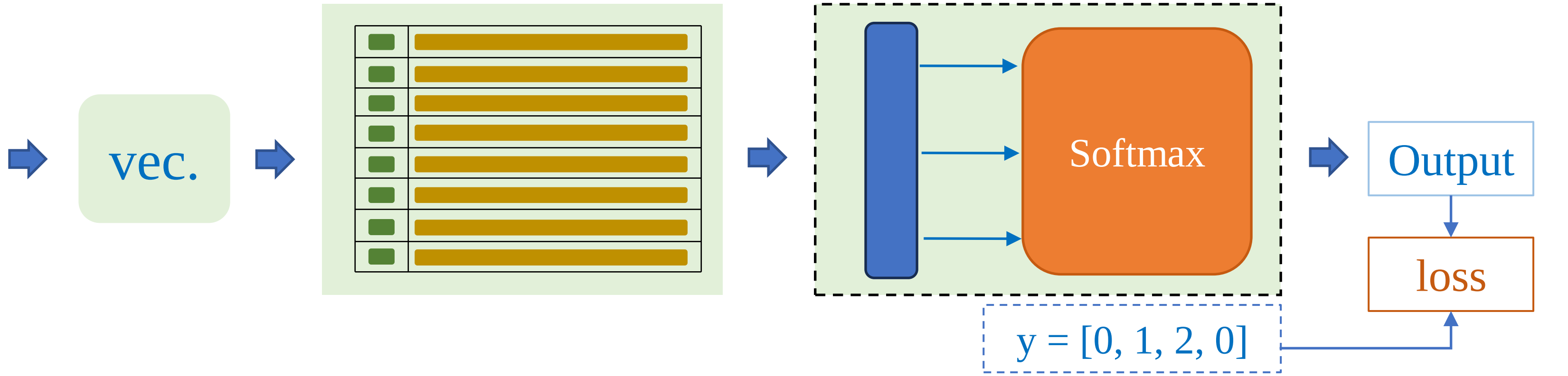
```
embedding = nn.Embedding(8, 4)
fc = nn.Linear(4, 3)

# forward
x = embedding(x)
x = fc(x)
x = x.permute(0, 2, 1)
```

```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

[0.4058, -0.6624, -0.8745, 0.7203]
[0.4309, -1.3067, -0.8823, 1.5977]
[-0.1882, 0.5530, 1.6267, 0.7013]
[-1.3083, -0.0987, 0.7647, -0.3680]



Problem?



POS Tagging (3): Using Padding

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

building dictionary

index	word
0	[UNK]
1	[pad]
2	a
3	are
4	books
5	dog
6	expensive
7	i
8	want

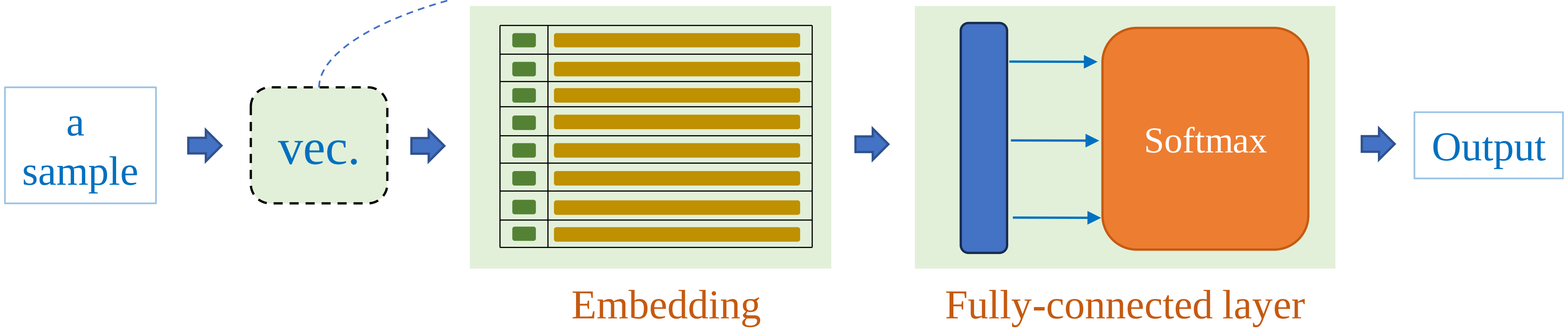
vocab size = 9
sequence length = 4

Label

0: (pro)noun
1: verb
2: others

sample 1		vec 1	label 1
i	→	7	0
want		8	1
a		2	2
dog		5	0

sample 2		vec 2	label 2
books	→	4	0
are		3	1
expensive		6	2
<pad>		1	?



1

POS Tagging (3): Using Padding

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

building dictionary

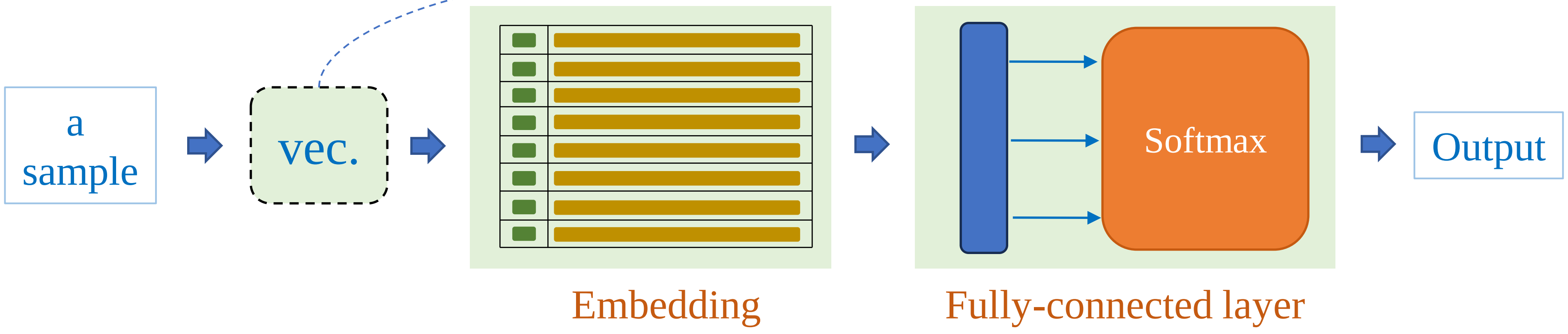
index	word
0	[UNK]
1	[pad]
2	a
3	are
4	books
5	dog
6	expensive
7	i
8	want

vocab size = 9
sequence length = 4

Label

0: (pro)noun
1: verb
2: others
3: <pad>

sample 1	vec 1	label 1	sample 2	vec 2	label 2
i	7	0	books	4	0
want	8	1	are	3	1
a	2	2	expensive	6	2
dog	5	0	<pad>	1	3

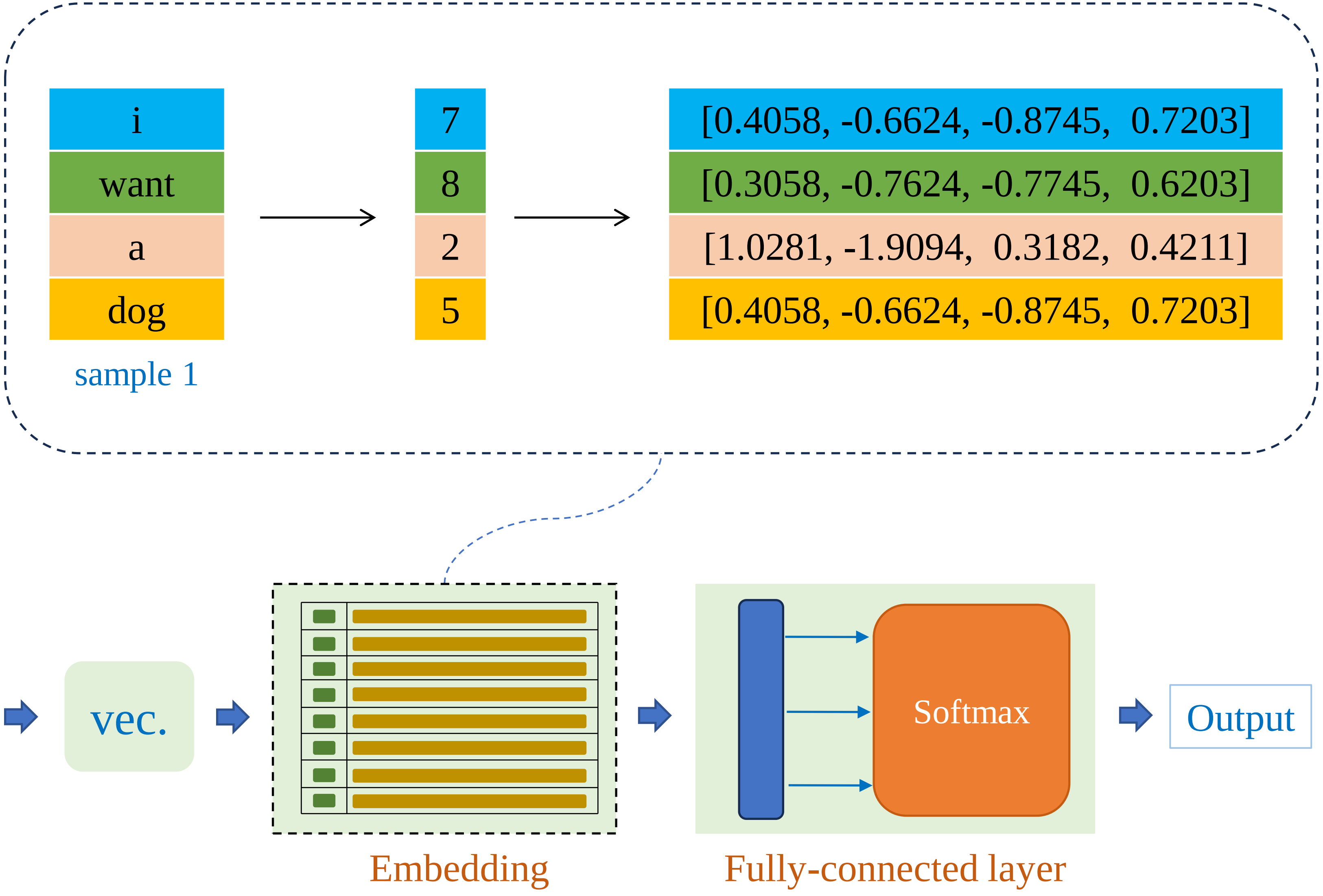


POS Tagging (3): Using Padding

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]
8	[0.3058, -0.7624, -0.7745, 0.6203]

vocab size = 9
sequence length = 4



W

[-0.3875, -0.3519, -0.1275, -0.1719]
[0.4391, 0.0455, -0.1566, -0.2897]
[0.1777, -0.1178, -0.3101, -0.2451]
[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

$\hat{y}$

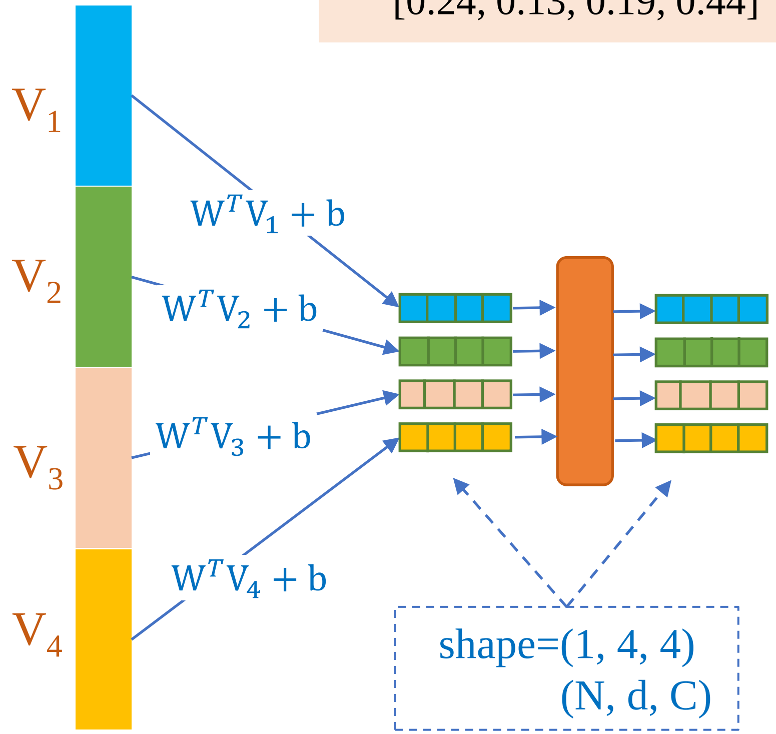
[0.26, 0.09, 0.16, 0.49]
[0.27, 0.13, 0.19, 0.41]
[0.29, 0.15, 0.21, 0.35]
[0.24, 0.13, 0.19, 0.44]

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

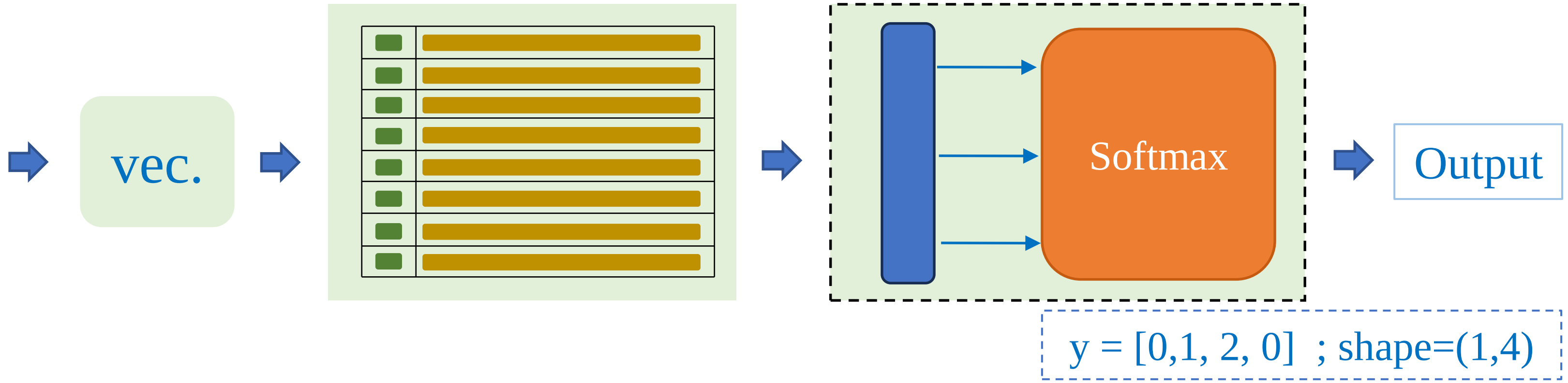
Label	
0: (pro)noun	2: others
1: verb	3: <pad>

0	[-0.1882, 0.5530, 1.6267, 0.7013]
1	[1.7840, -0.8278, -0.2701, 1.3586]
2	[1.0281, -1.9094, 0.3182, 0.4211]
3	[-1.3083, -0.0987, 0.7647, -0.3680]
4	[0.2293, 1.3255, 0.1318, 2.0501]
5	[0.4058, -0.6624, -0.8745, 0.7203]
6	[0.5582, 0.0786, -0.6817, 0.6902]
7	[0.4309, -1.3067, -0.8823, 1.5977]
8	[0.3058, -0.7624, -0.7745, 0.6203]

[0.4058, -0.6624, -0.8745, 0.7203]
[0.3058, -0.7624, -0.7745, 0.6203]
[1.0281, -1.9094, 0.3182, 0.4211]
[0.4058, -0.6624, -0.8745, 0.7203]



vocab size = 9
sequence length = 4



W

[-0.3875, -0.3519, -0.1275, -0.1719]
[0.4391, 0.0455, -0.1566, -0.2897]
[0.1777, -0.1178, -0.3101, -0.2451]
[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

Shape of logits = (N, C, d)

Shape of target = (N, d)

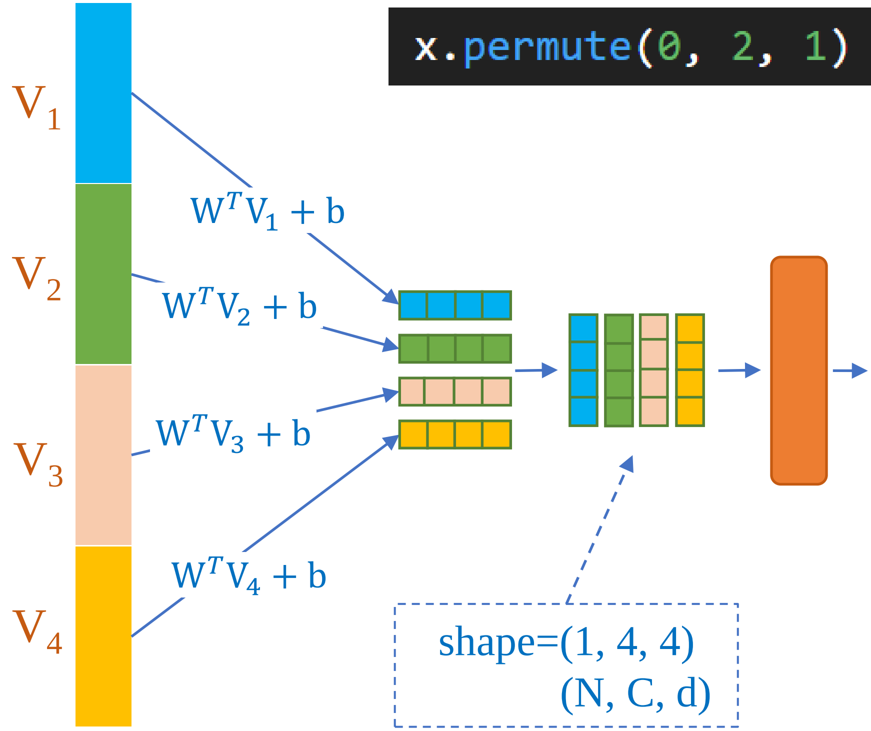
pytorch requirement

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

Label	
0: (pro)noun	2: others
1: verb	3: <pad>

[0.4058, -0.6624, -0.8745, 0.7203]
[0.3058, -0.7624, -0.7745, 0.6203]
[1.0281, -1.9094, 0.3182, 0.4211]
[0.4058, -0.6624, -0.8745, 0.7203]

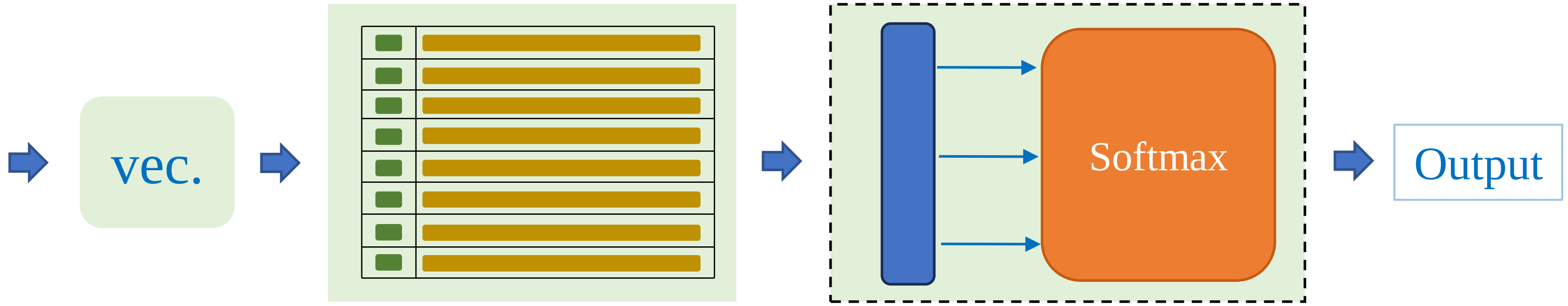
```
x.permute(0, 2, 1)
```




```
embedding = nn.Embedding(9, 4)
fc = nn.Linear(4, 4)

# forward
x = embedding(x)
x = fc(x)
x = x.permute(0, 2, 1)
```

```
nn.CrossEntropyLoss()
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

problem?



vocab size = 9

sequence length = 4

y = [0, 1, 2, 0] ; shape=(1,4)

W

[-0.3875, -0.3519, -0.1275, -0.1719]
[0.4391, 0.0455, -0.1566, -0.2897]
[0.1777, -0.1178, -0.3101, -0.2451]
[0.3730, 0.0996, -0.3004, 0.2219]

b

[0.3548, -0.2819, -0.0579, 0.5113]

Shape of logits = (N, C, d)

Shape of target = (N, d)

pytorch requirement

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

Label

0: (pro)noun

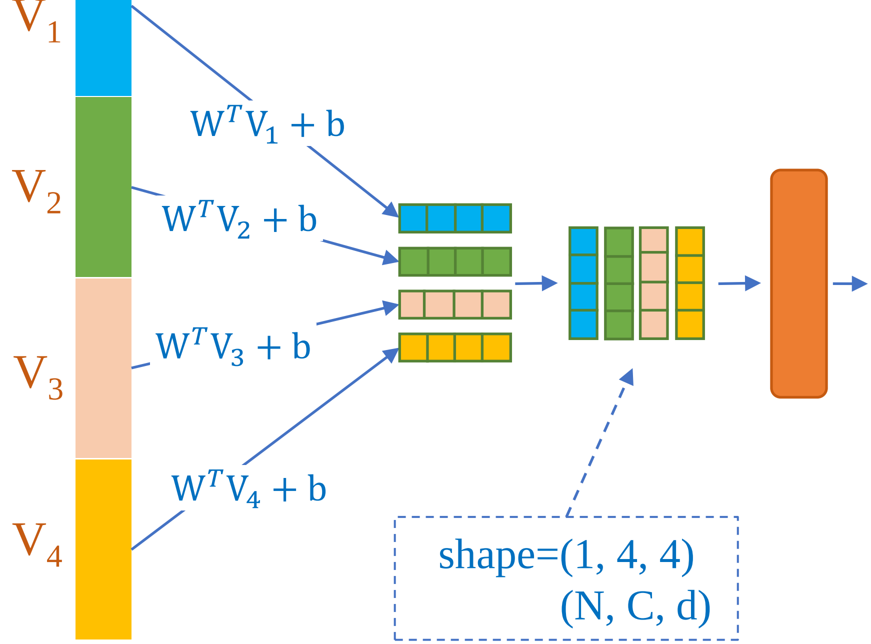
1: verb

2: others

3: <pad>

[0.4058, -0.6624, -0.8745, 0.7203]
[0.3058, -0.7624, -0.7745, 0.6203]
[1.0281, -1.9094, 0.3182, 0.4211]
[0.4058, -0.6624, -0.8745, 0.7203]

```
x.permute(0, 2, 1)
```

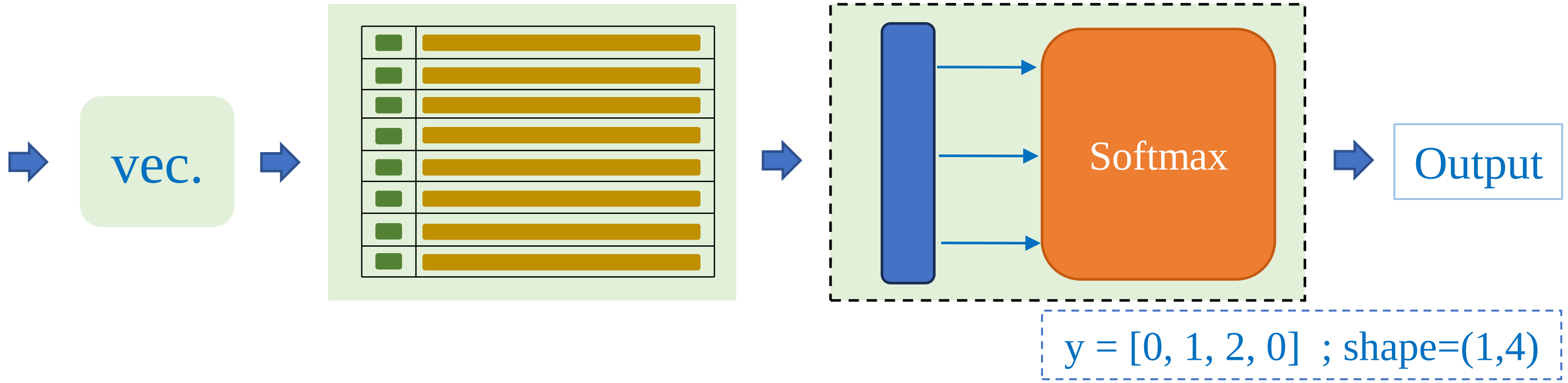



```
embedding = nn.Embedding(9, 4)
fc = nn.Linear(4, 4)

# forward
x = embedding(x)
x = fc(x)
x = x.permute(0, 2, 1)
```

```
nn.CrossEntropyLoss(ignore_index=3)
torch.optim.Adam(model.parameters(),
                 lr=0.1)
```

vocab size = 9

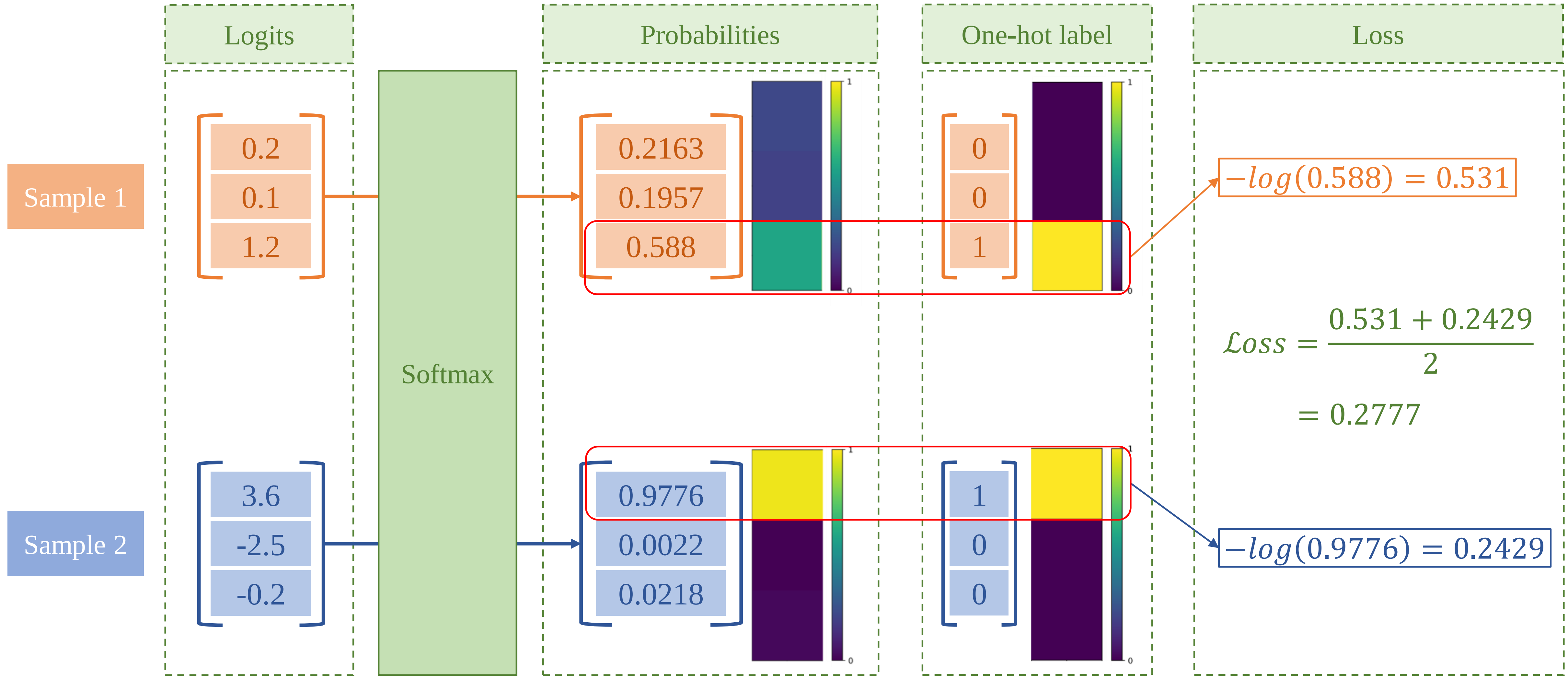
sequence length = 4



Cross Entropy Loss

N_classes = 3

$$L = -\sum_i y_i log(\hat{y}_i)$$

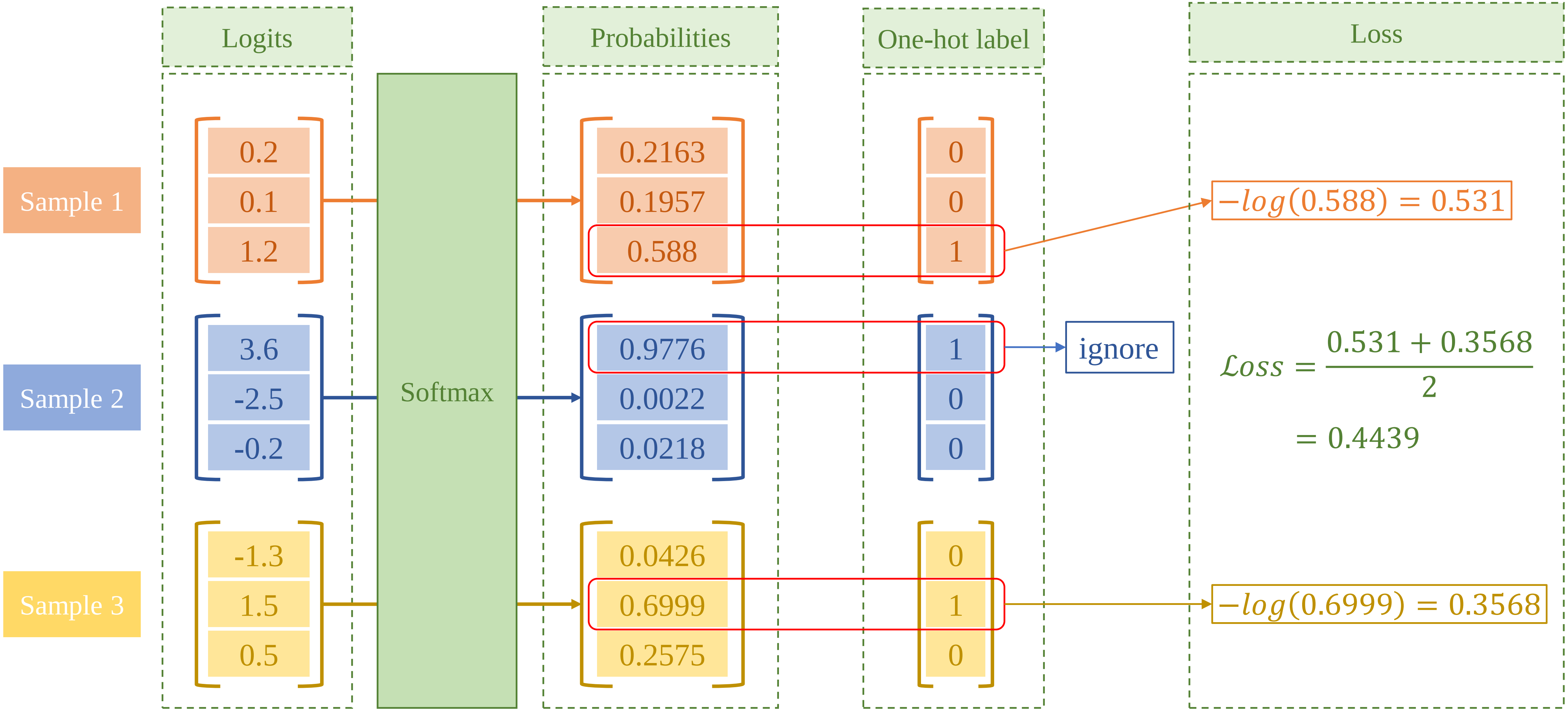


Cross Entropy Loss

N_classes = 3

$$L = -\sum_i y_i log(\hat{y}_i)$$

Ignore_index = 0



Optional Section for Wednesday

Designing a Model for POS Tagging

Doc	Label
i want a dog	[0, 1, 2, 0]
books are expensive	[0, 1, 2]

Label	Meaning
0	Noun/Pronoun
1	Verb
2	Others

building dictionary →

vocab size = 9
sequence length = 4

index	word
0	[UNK]
1	[pad]
2	a
3	are
4	books
5	dog
6	expensive
7	i
8	want

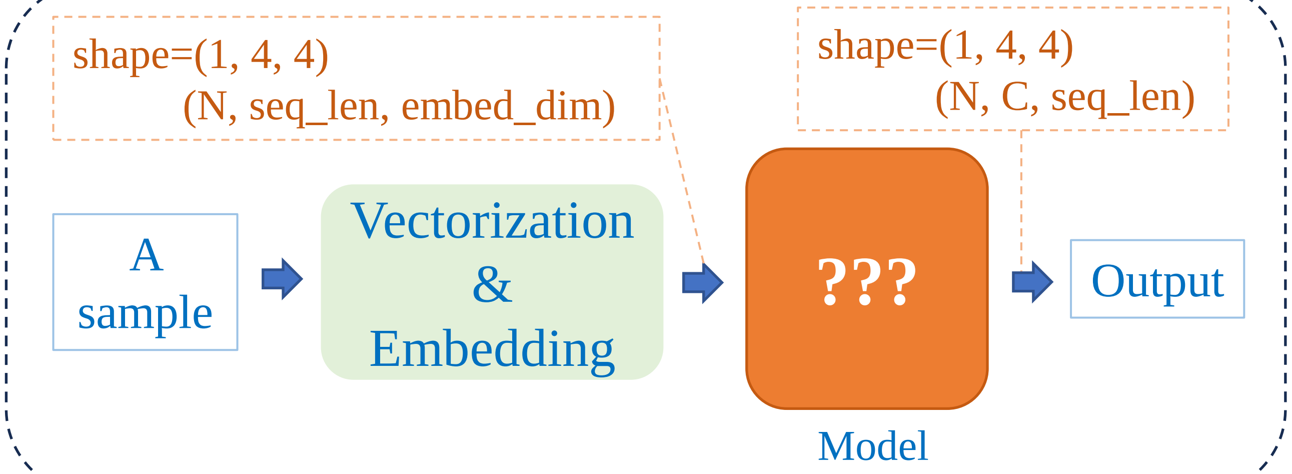
Dictionary

0	[-0.1882, 0.5530, …, 0.7013]
1	[1.7840, -0.8278, …, 1.3586]
2	[1.0281, -1.9094, …, 0.4211]
3	[-1.3083, -0.0987, …, -0.3680]
4	[0.2293, 1.3255, …, 2.0501]
5	[0.4058, -0.6624, …, 0.7203]
6	[0.5582, 0.0786, …, 0.6902]
7	[0.4309, -1.3067, …, 1.5977]
8	[0.3058, -0.7624, …, 0.6203]

Embedding

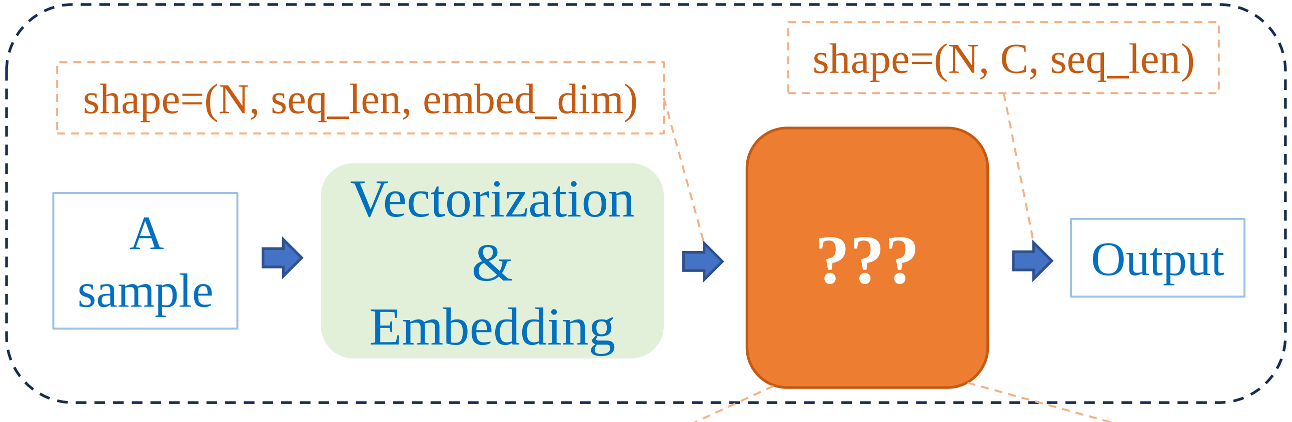
sample 1				sample 1 _ Embedding
i	→	7	→	[0.4058, -0.6624, …, 0.7203]
want	→	8	→	[0.3058, -0.7624, …, 0.6203]
a	→	2	→	[1.0281, -1.9094, …, 0.4211]
dog	→	5	→	[0.4058, -0.6624, …, 0.7203]

Vectorization and Embedding

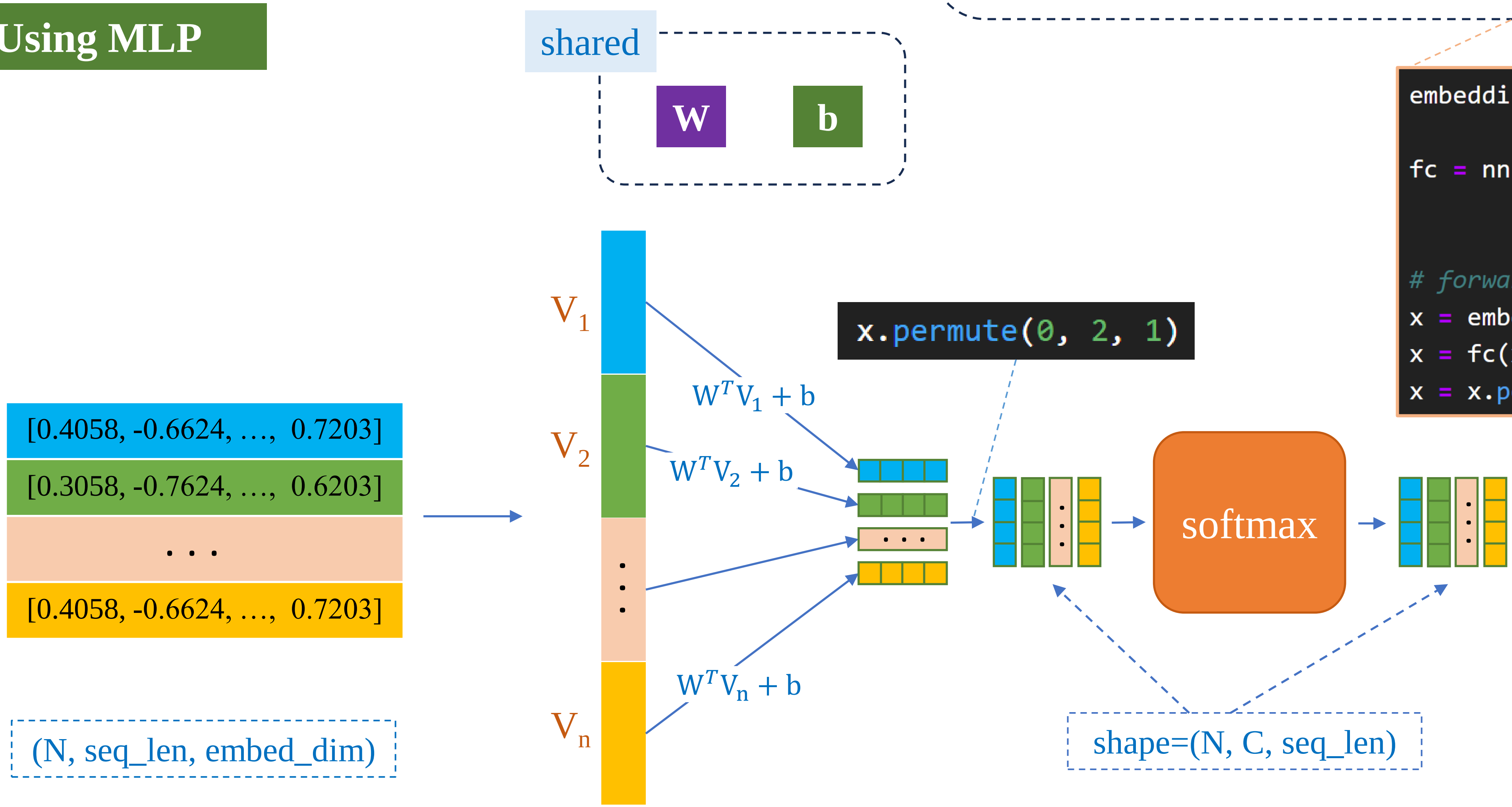


Model Pipeline

Designing a Model for POS Tagging

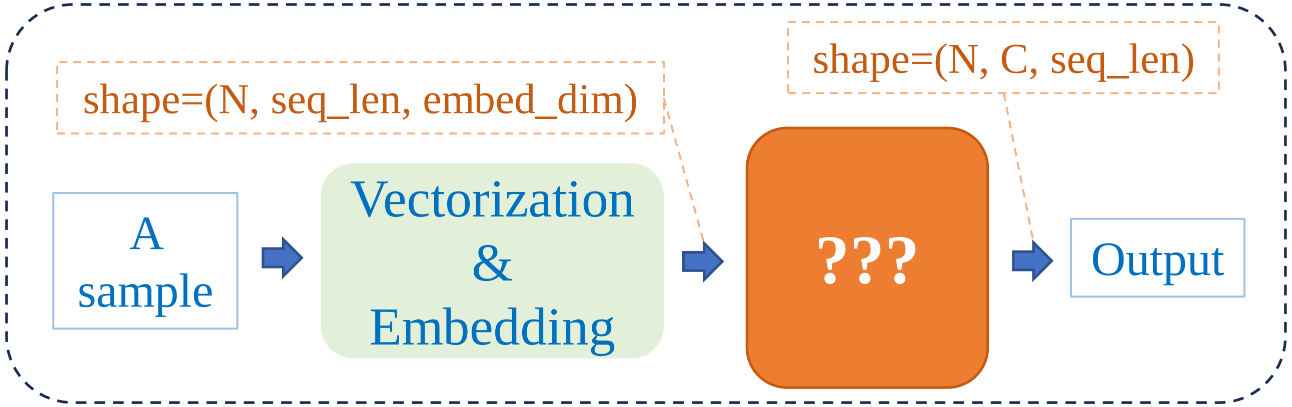


Using MLP




```python
embedding = nn.Embedding(vocab_size,
                         embed_dim)
fc = nn.Linear(embed_dim,
               num_classes)

# forward
x = embedding(x)
x = fc(x)
x = x.permute(0, 2, 1)
```

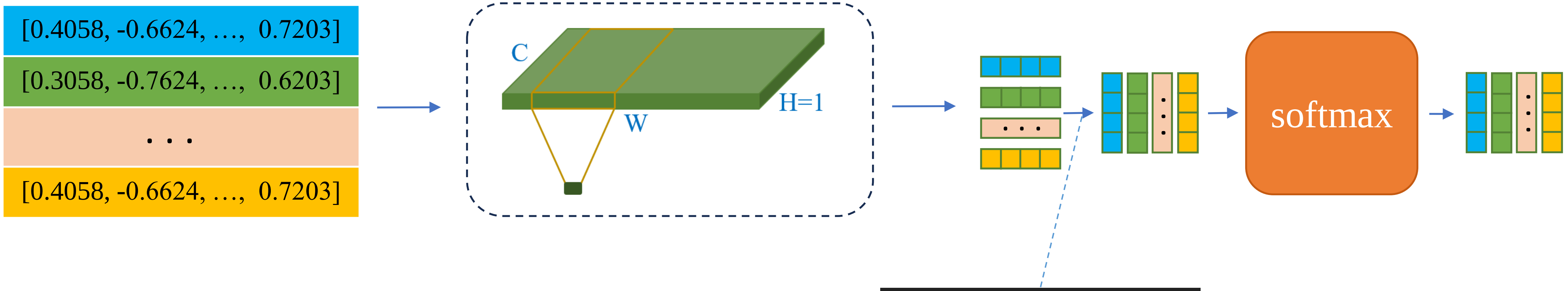
Designing a Model for POS Tagging



Using CNN

```
embedding = nn.Embedding(vocab_size, 3)
conv1d = nn.Conv1d(3, num_classes,
                   kernel_size=2, padding='same')
# forward
x = self.embedding(x)
x = self.conv1d(x)
x = x.permute(0, 2, 1)
```

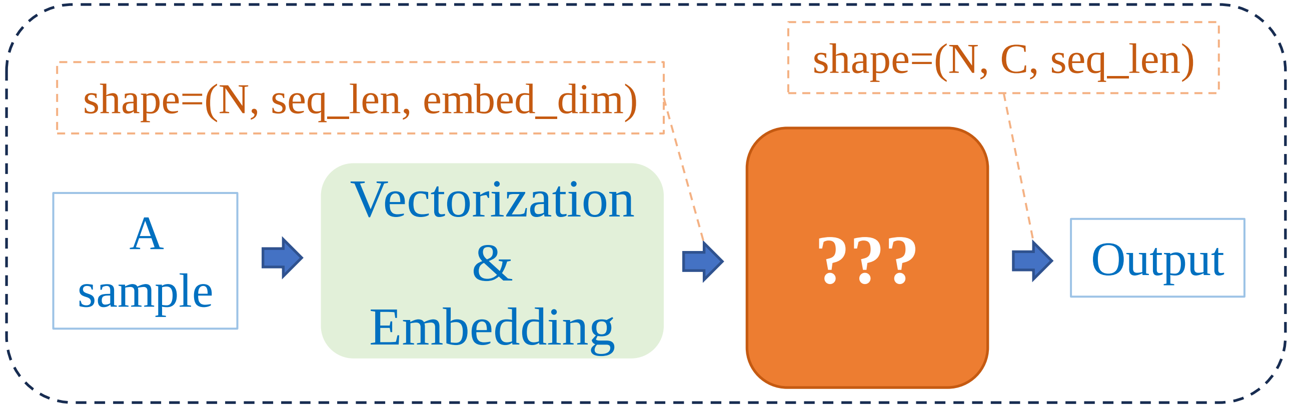
This pipeline is wrong.
Let's find out!




```
x.permute(0, 2, 1)
```

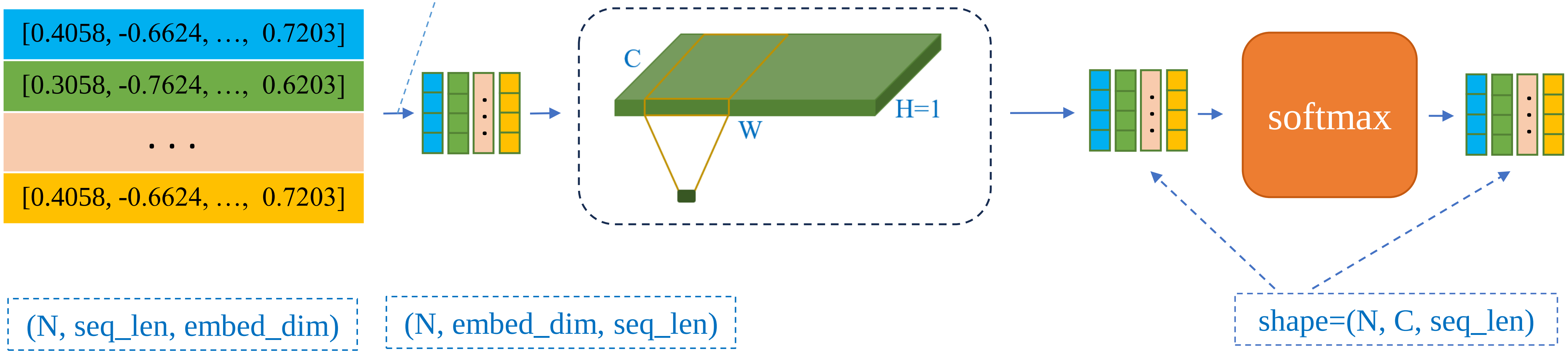
Designing a Model for POS Tagging

Using CNN



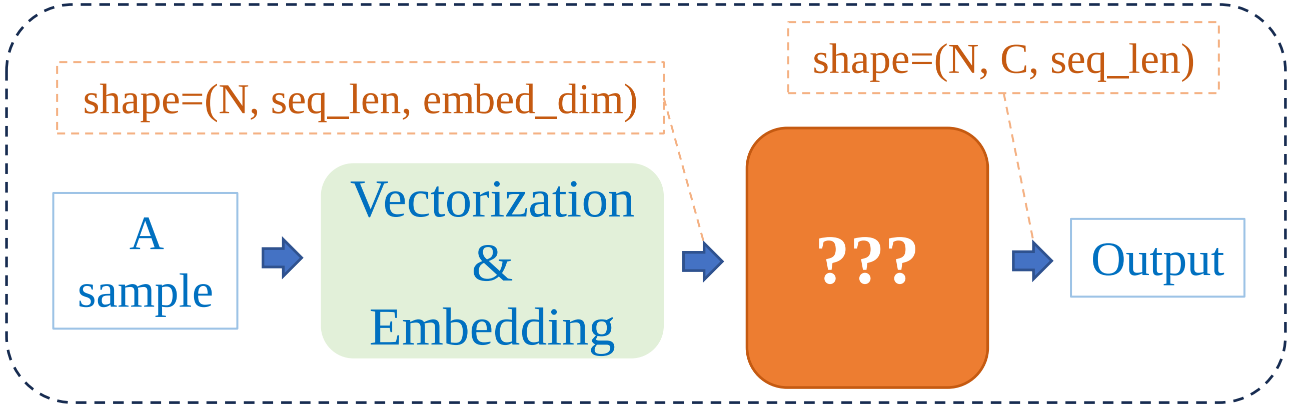

```python
embedding = nn.Embedding(vocab_size, 3)
conv1d = nn.Conv1d(3, num_classes,
                   kernel_size=2, padding='same')
# forward
x = self.embedding(x)
x = x.permute(0, 2, 1)
x = self.conv1d(x)
```

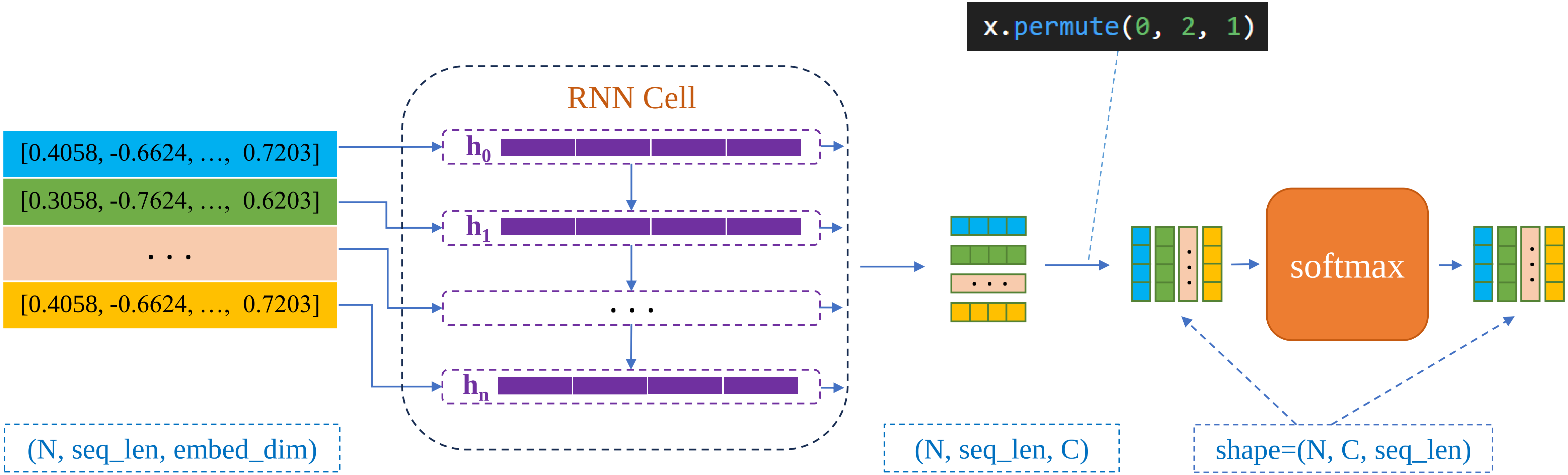
```python
x.permute(0, 2, 1)
```



Designing a Model for POS Tagging

Using RNN

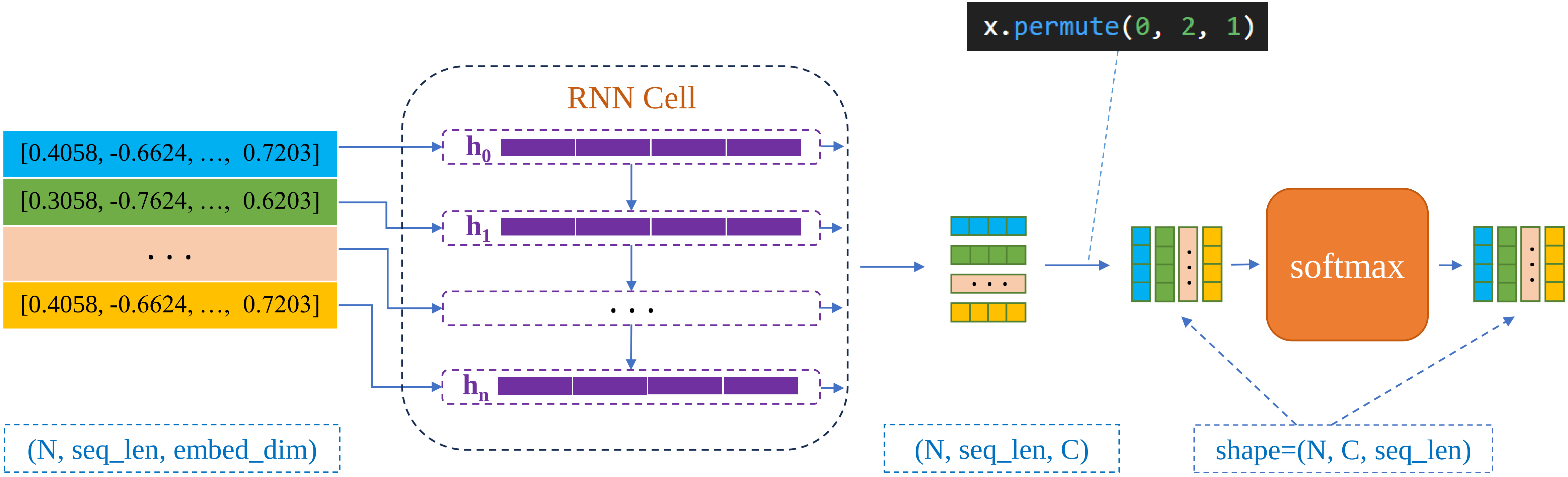




Designing a Model for POS Tagging

Using RNN: Implementation

```python
embedding = nn.Embedding(vocab_size, emb_dim)
recurrent = nn.RNN(emb_dim, num_classes, batch_first=True)

# forward
x = embedding(x)
output, _ = recurrent(x)
x = output.permute(0, 2, 1)
```

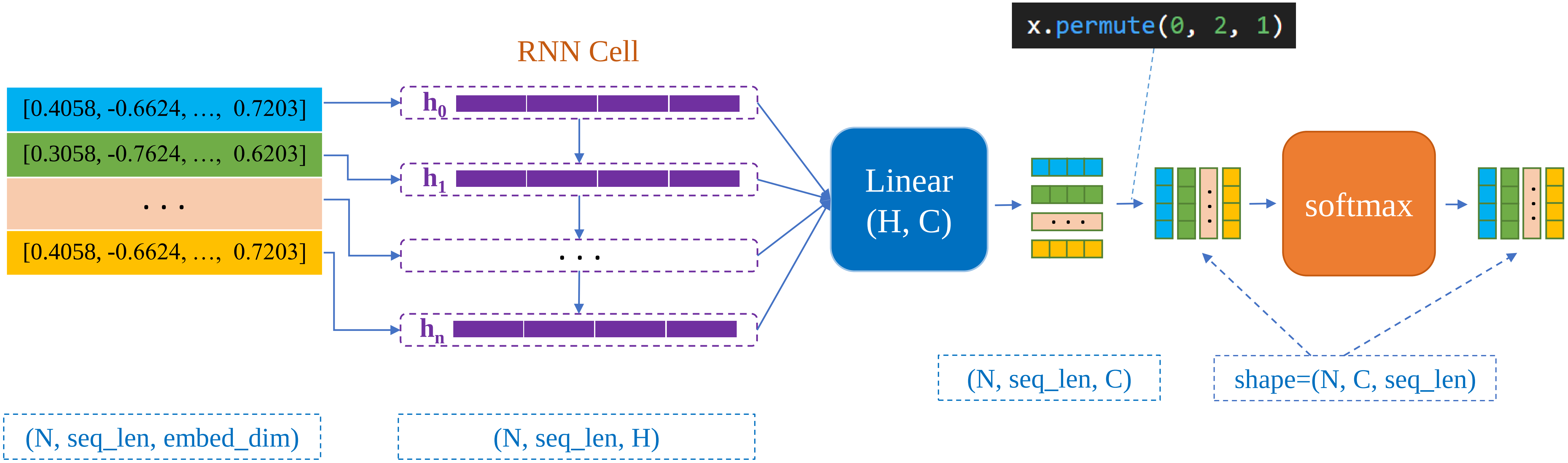


Designing a Model for POS Tagging

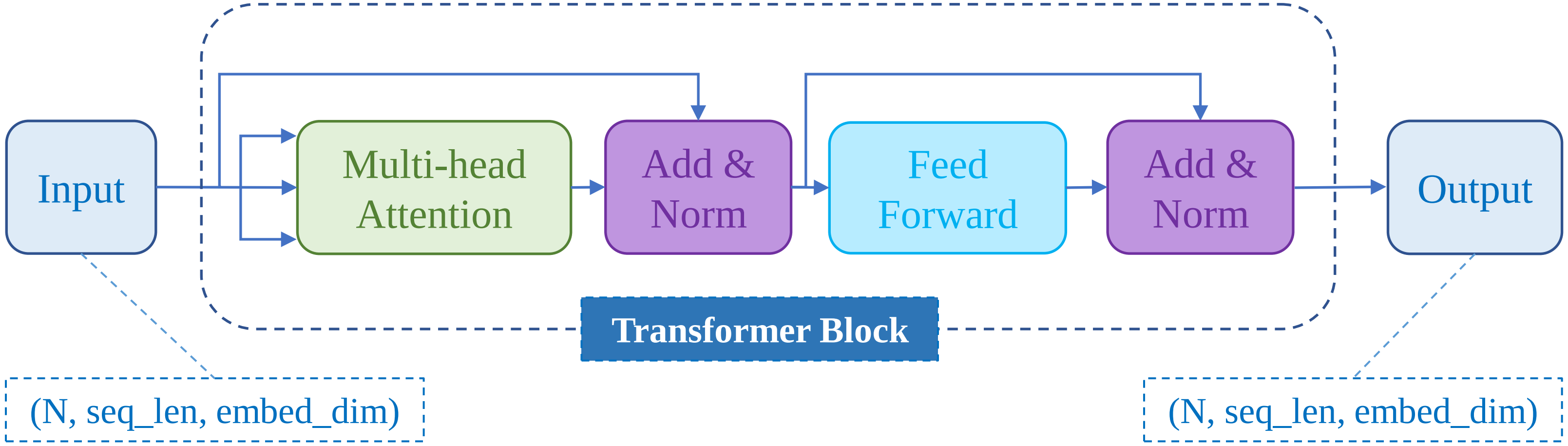
Using RNN + Linear

Similar to LSTM/GRU

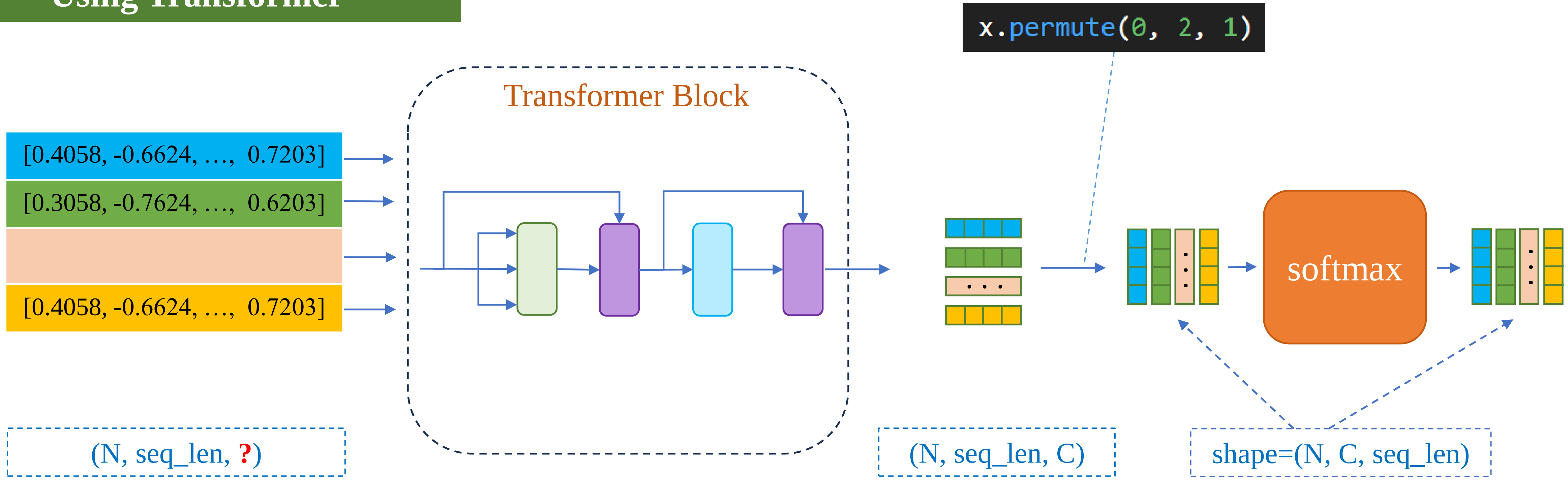
```python
embedding = nn.Embedding(vocab_size, emb_dim)
recurrent = nn.RNN(emb_dim, hidden_size, batch_first=True)
fc = nn.Linear(hidden_size, num_classes)

# forward
x = embedding(x)
output, _ = recurrent(x)
x = fc(output)
x = x.permute(0, 2, 1)
```



Designing a Model for POS Tagging



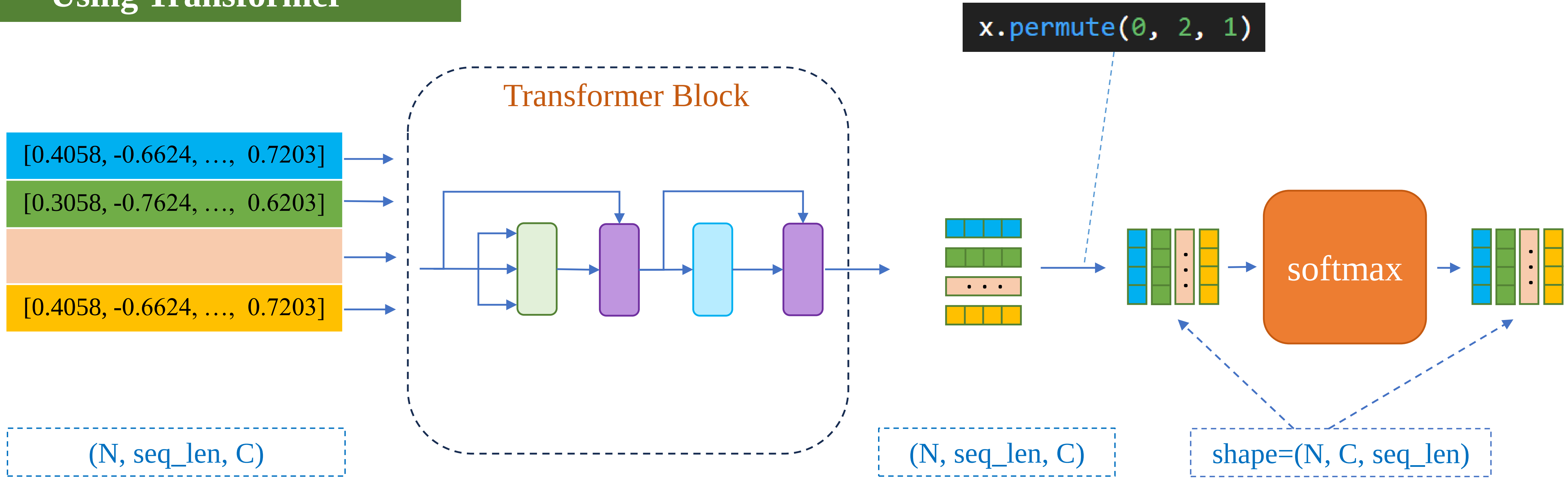
Using Transformer



Designing a Model for POS Tagging

```
embedding = nn.Embedding(vocab_size, 4)
transformer = TransformerBlock(4, 1, 4)
# embed_dim, num_heads, ff_dim

# forward
x = self.embedding(x)
x = self.transformer(x, x, x)
x =  x.permute(0, 2, 1)
```

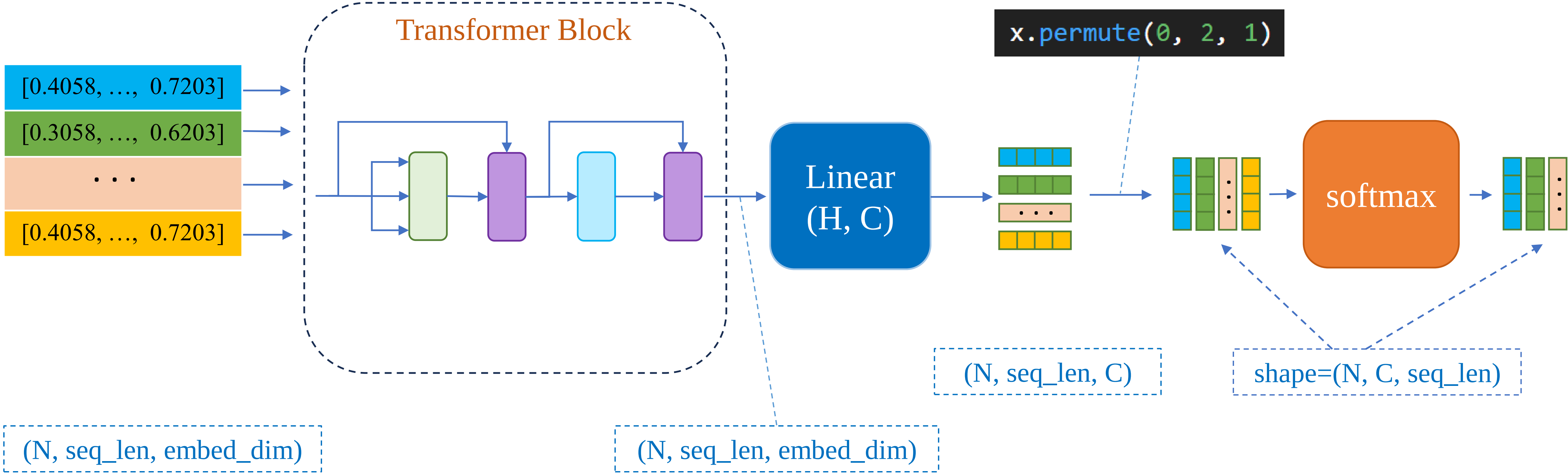
Using Transformer

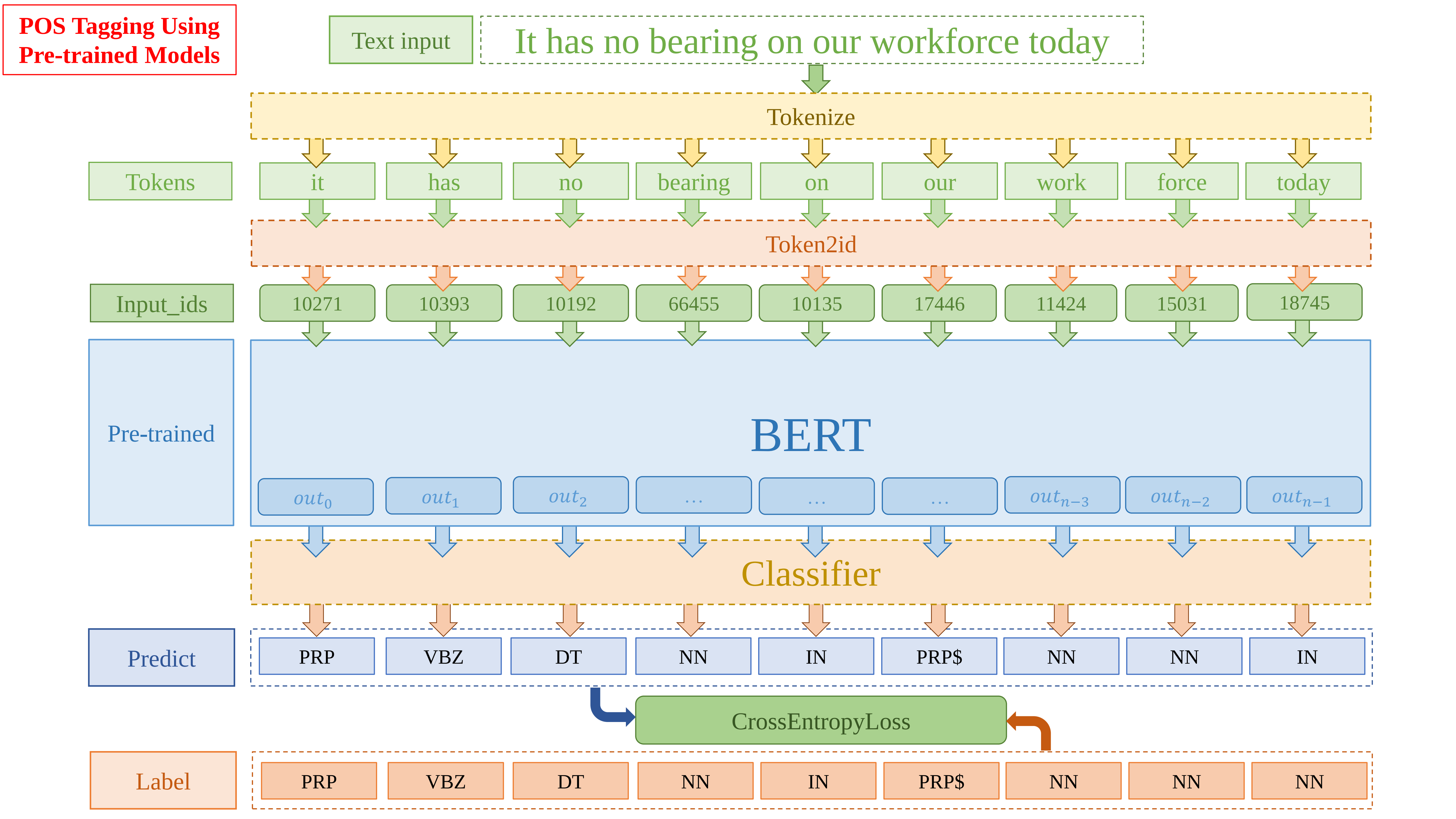


Designing a Model for POS Tagging

Using Transformer + Linear

```python
embedding = nn.Embedding(vocab_size, embed_dim)
transformer = TransformerBlock(embed_dim, 1, embed_dim)
fc = nn.Linear(embed_dim, num_classes)

# forward
x = self.embedding(x)
x = self.transformer(x, x, x)
x = self.fc(x)
x = x.permute(0, 2, 1)
```




POS Tagging Using
Pre-trained Models
Text input
It has no bearing on our workforce today
Tokenize
Tokens
it
has
no
bearing
on
our
work
force
today
Token2id
Input_ids
10271
10393
10192
66455
10135
17446
11424
15031
18745
Pre-trained
BERT
out_0
out_1
out_2
...
...
...
out_{n-3}
out_{n-2}
out_{n-1}
Classifier
Predict
PRP
VBZ
DT
NN
IN
PRP$
NN
NN
IN
CrossEntropyLoss
Label
PRP
VBZ
DT
NN
IN
PRP$
NN
NN
NN