AI VIETNAM
All-in-One Course
(TA Session)

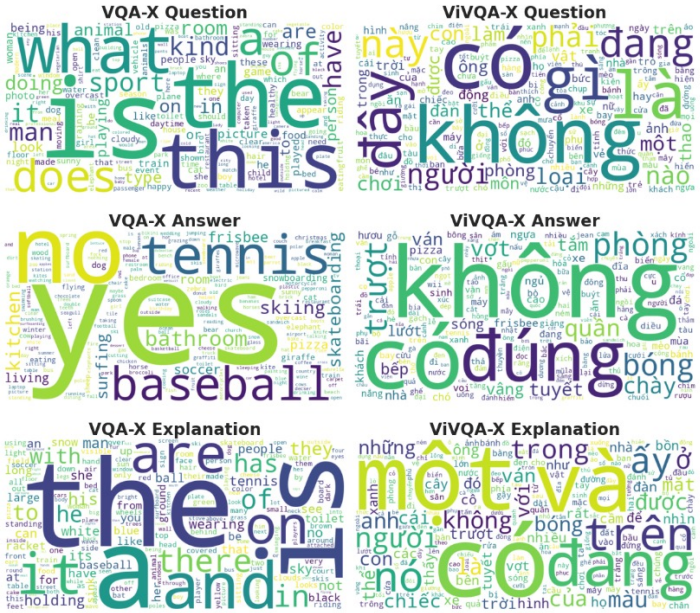
Introduction to Text Classification and POS Tagging

Thuan Duong – TA
Nguyen Nha – TA
Thu Van – sTA

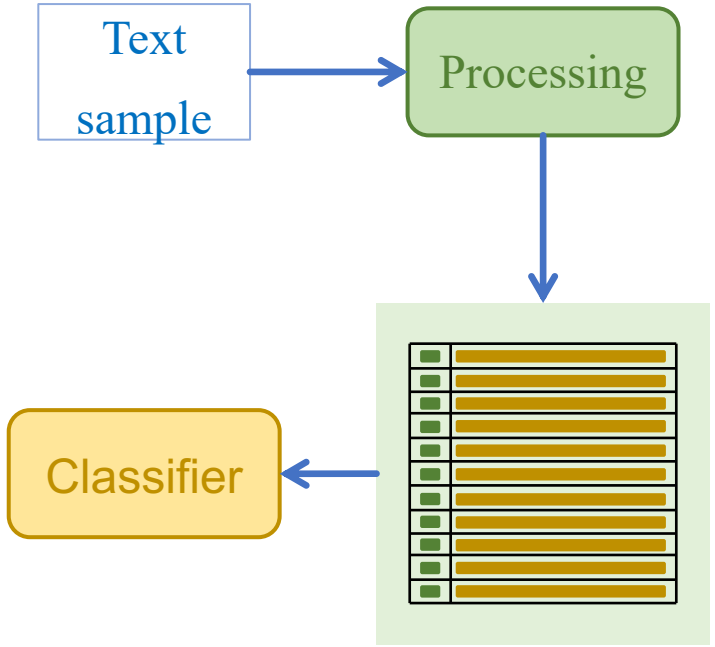
Year 2025

Objectives

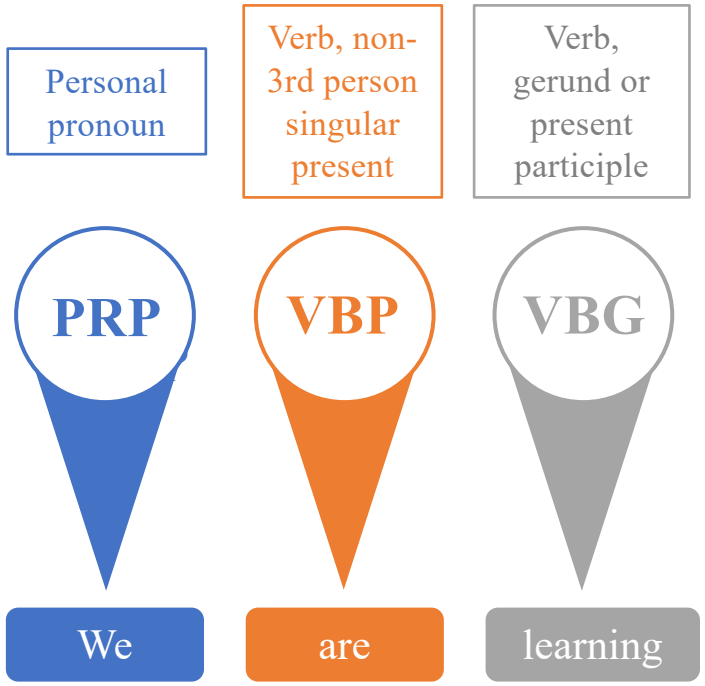
❖ NLP Concepts



❖ Text Classification



❖POS Tagging



Outline

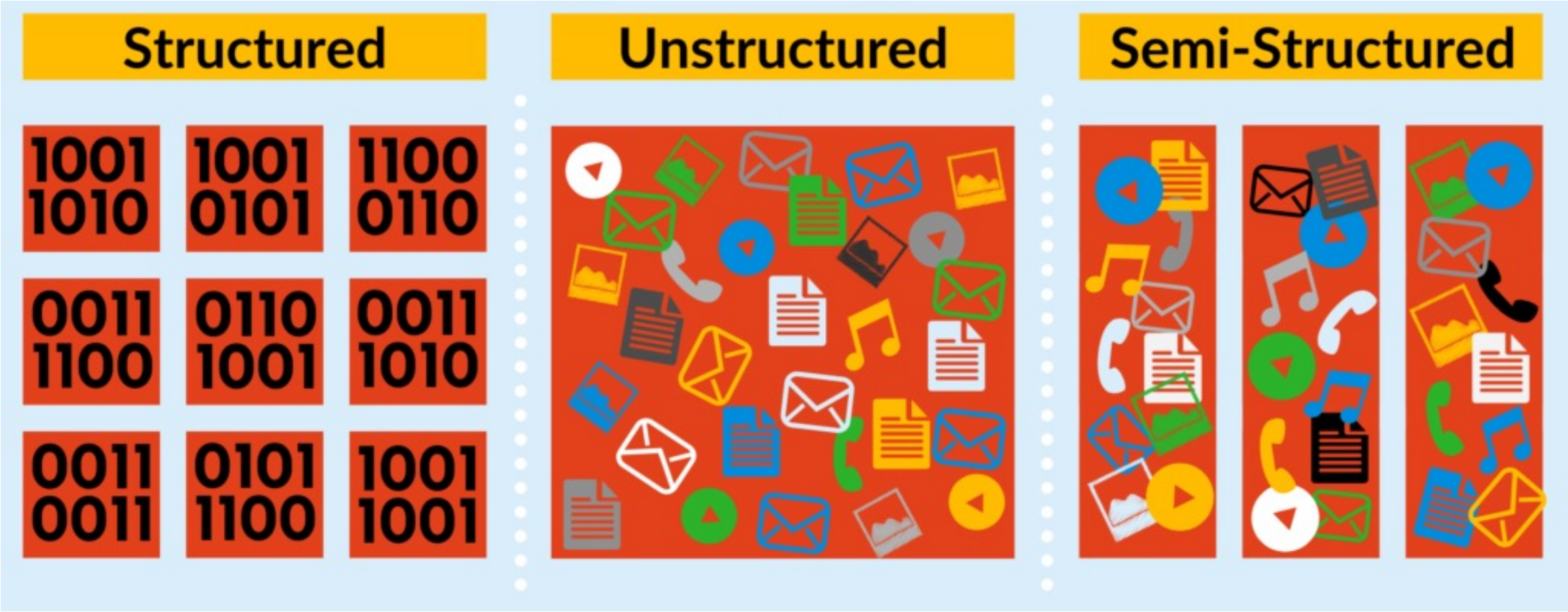
- Introduction to NLP
- Text Classification
- Part-of-Speech Tagging
- Question

AI VIETNAM
All-in-One Course
(TA Session)

Introduction to NLP

Introduction to NLP

❖ Data Types



Introduction to NLP

❖ Structured Data

Data organized in a predefined format (rows and columns)

order_id	name	order_date	total_amount
1	Alice Johnson	2024-02-01	150
2	Alice Johnson	2024-02-05	200
3	Bob Smith	2024-02-03	300
4	Charlie Brown	2024-02-06	50

name	total_spent
Bob Smith	300
Alice Johnson	350
Charlie Brown	50

Tables in database

Excel sheets

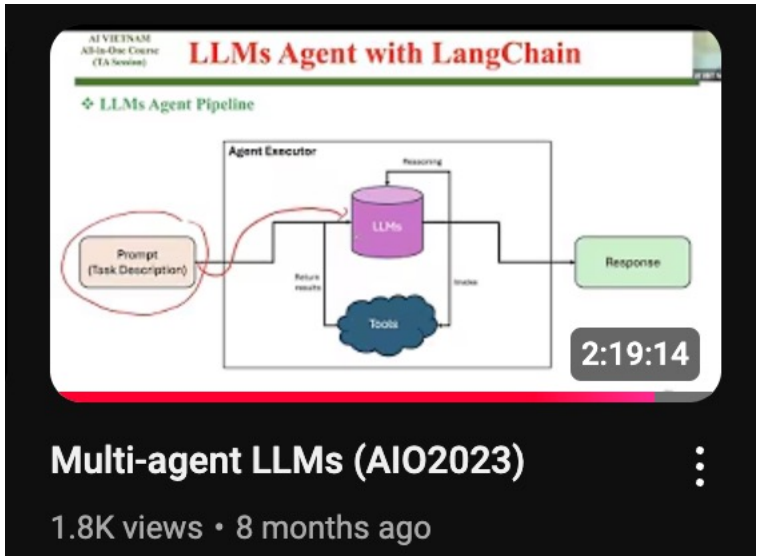
Pandas DataFrame

Introduction to NLP

❖ Unstructured Data

Data that does not have a predefined structure or format.

Image



Video

Free-Text

Introduction to NLP

❖ Semi-Structured Data

Semi-structured data is a type of data that does not have a rigid structure like relational databases but still contains some organizational properties.

```
{
  "_id": "123",
  "user": "john_doe",
  "email": "john@example.com",
  "preferences": {
    "theme": "dark",
    "notifications": true
  }
}
```

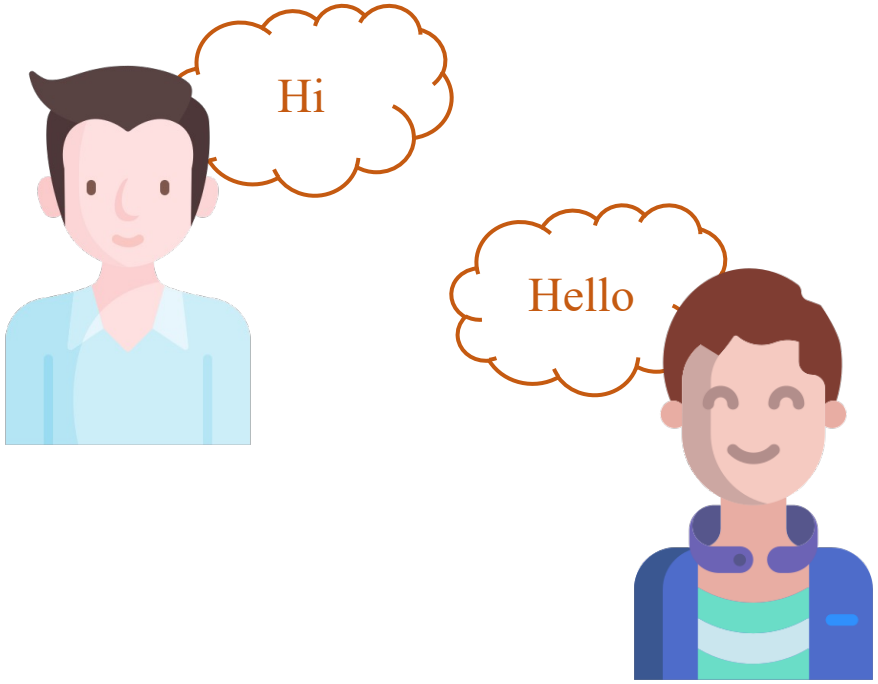
NoSQL Database

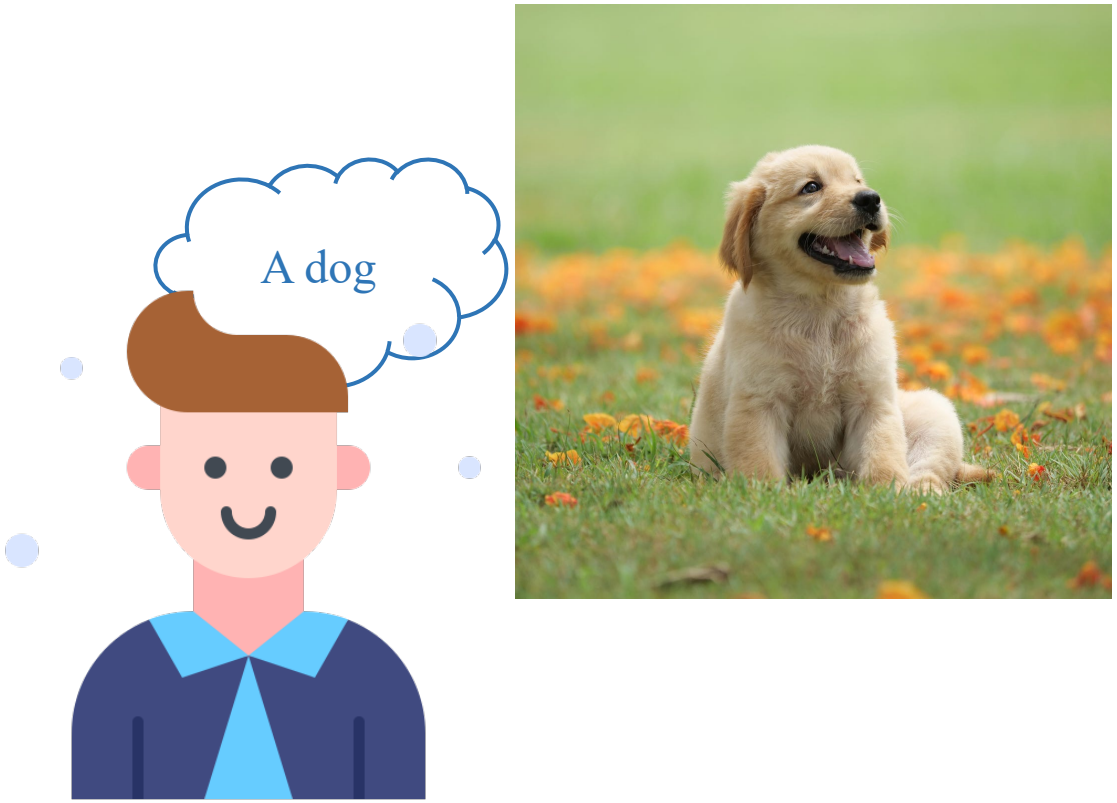
```
Name, Age, Skills
Alice, 25, "Python, Machine Learning"
Bob, 30, "Java, DevOps, Cloud"
```

CSV

Introduction to NLP

❖ Human vs Computer





Language → Speech/Text

Vision → Image

Introduction to NLP

❖ Human vs Computer

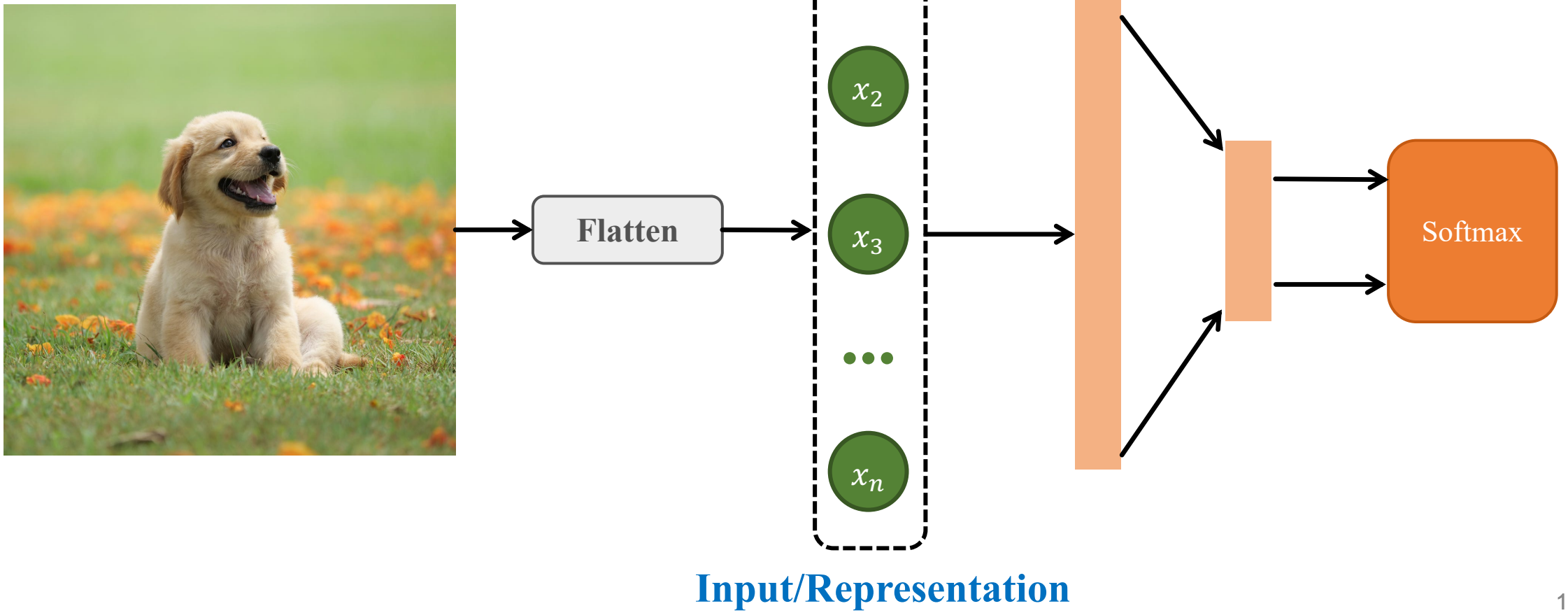
Collaborative coding with AI tools is allowed

Speech/Text

Image

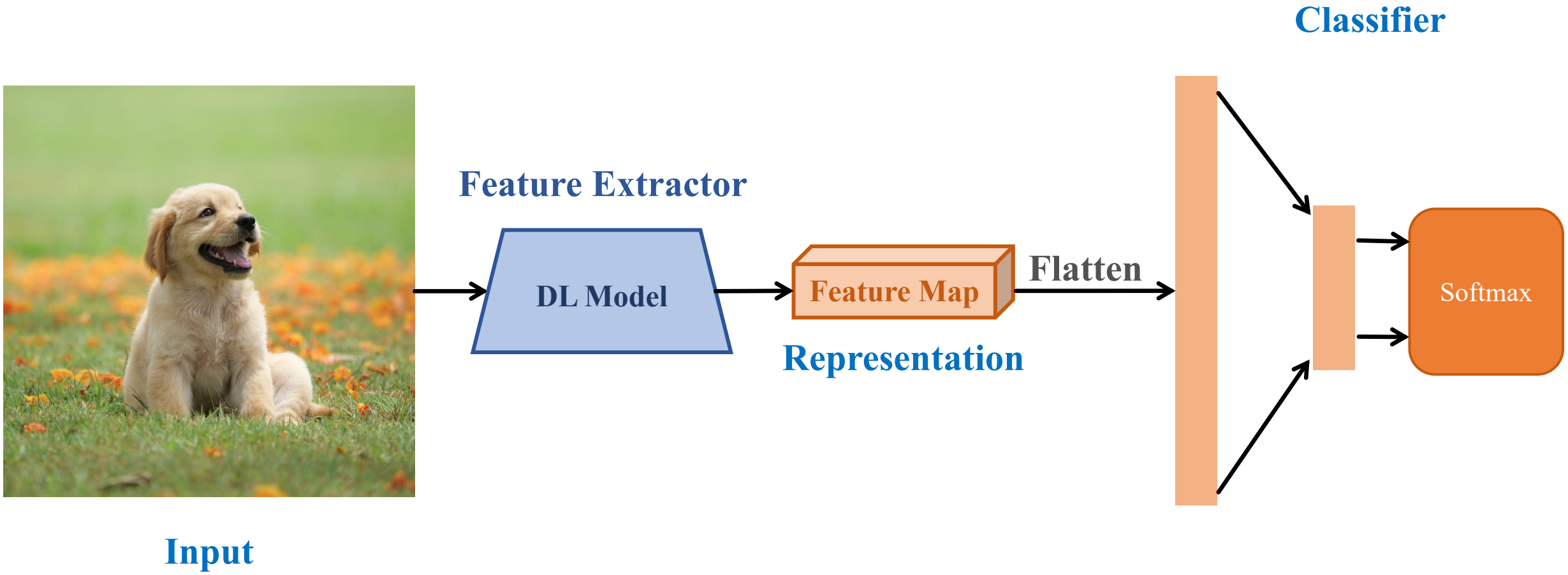
Introduction to NLP

❖ Vision in Computer



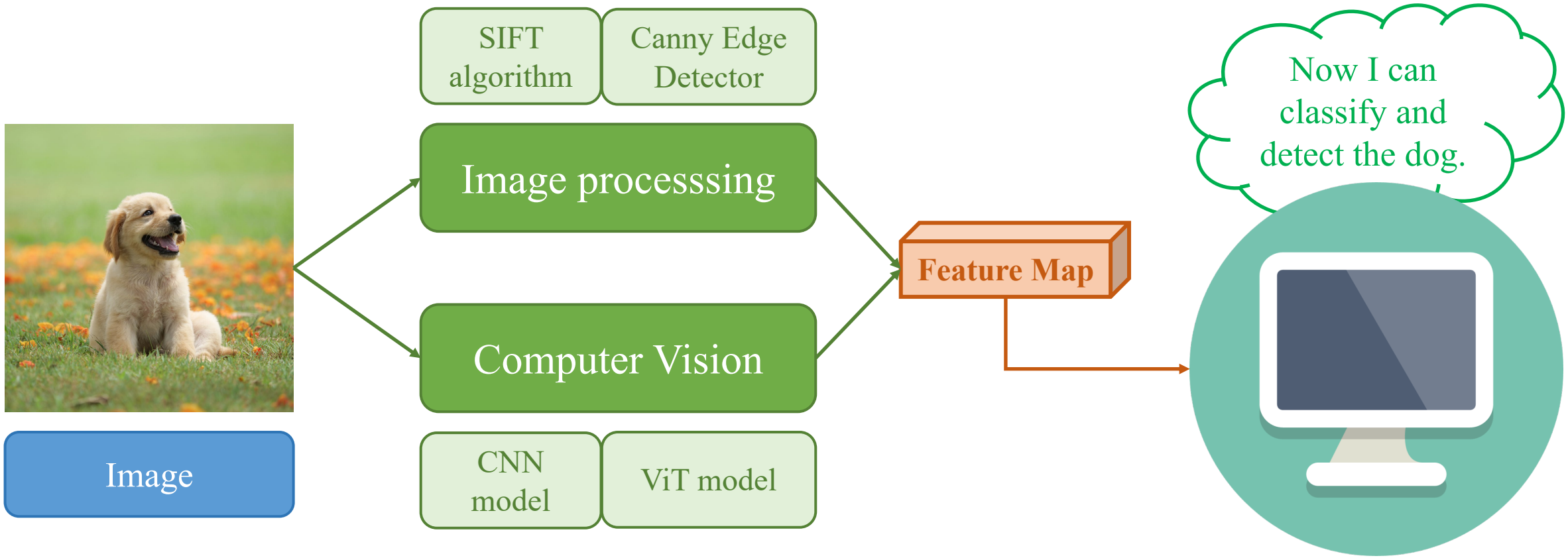
Introduction to NLP

❖ Vision in Computer



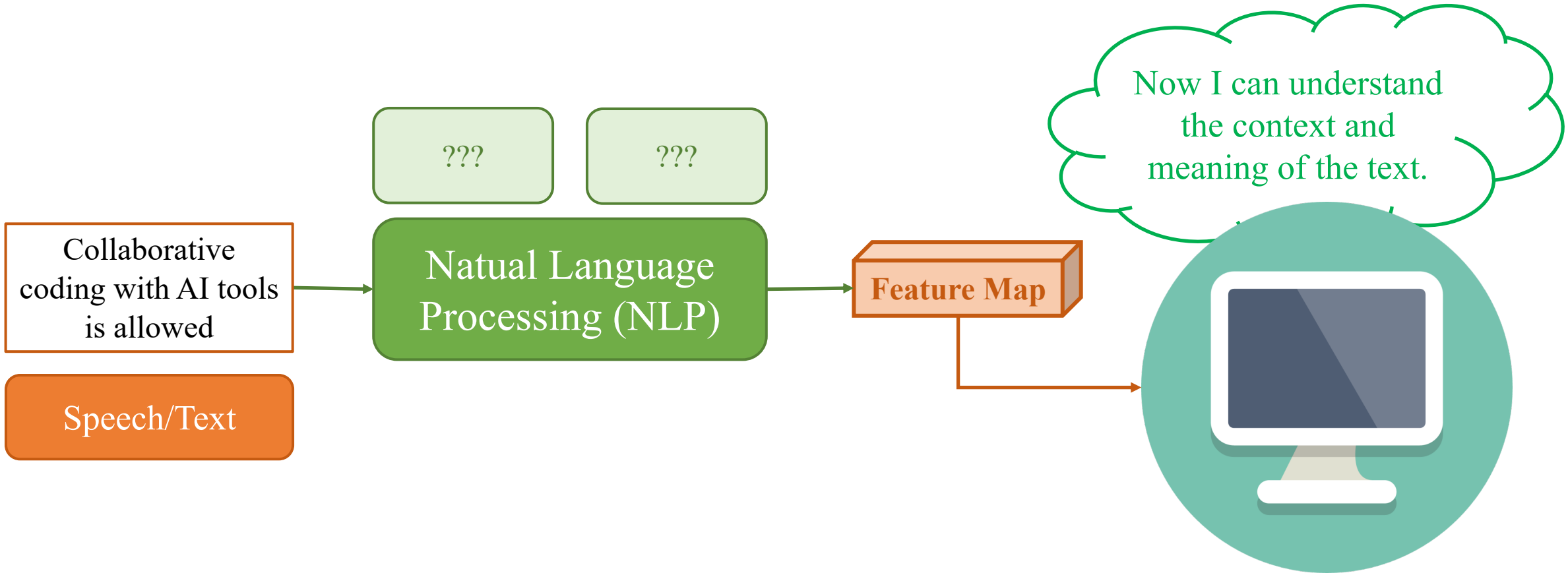
Introduction to NLP

❖ Vision in Computer



Introduction to NLP

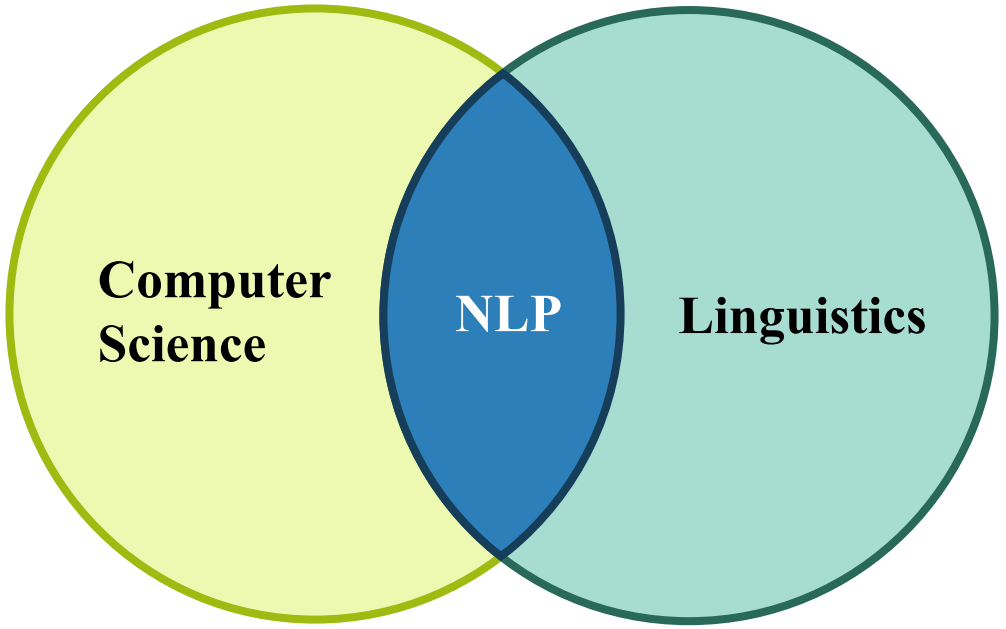
❖ Natural Language in Computer



Introduction to NLP

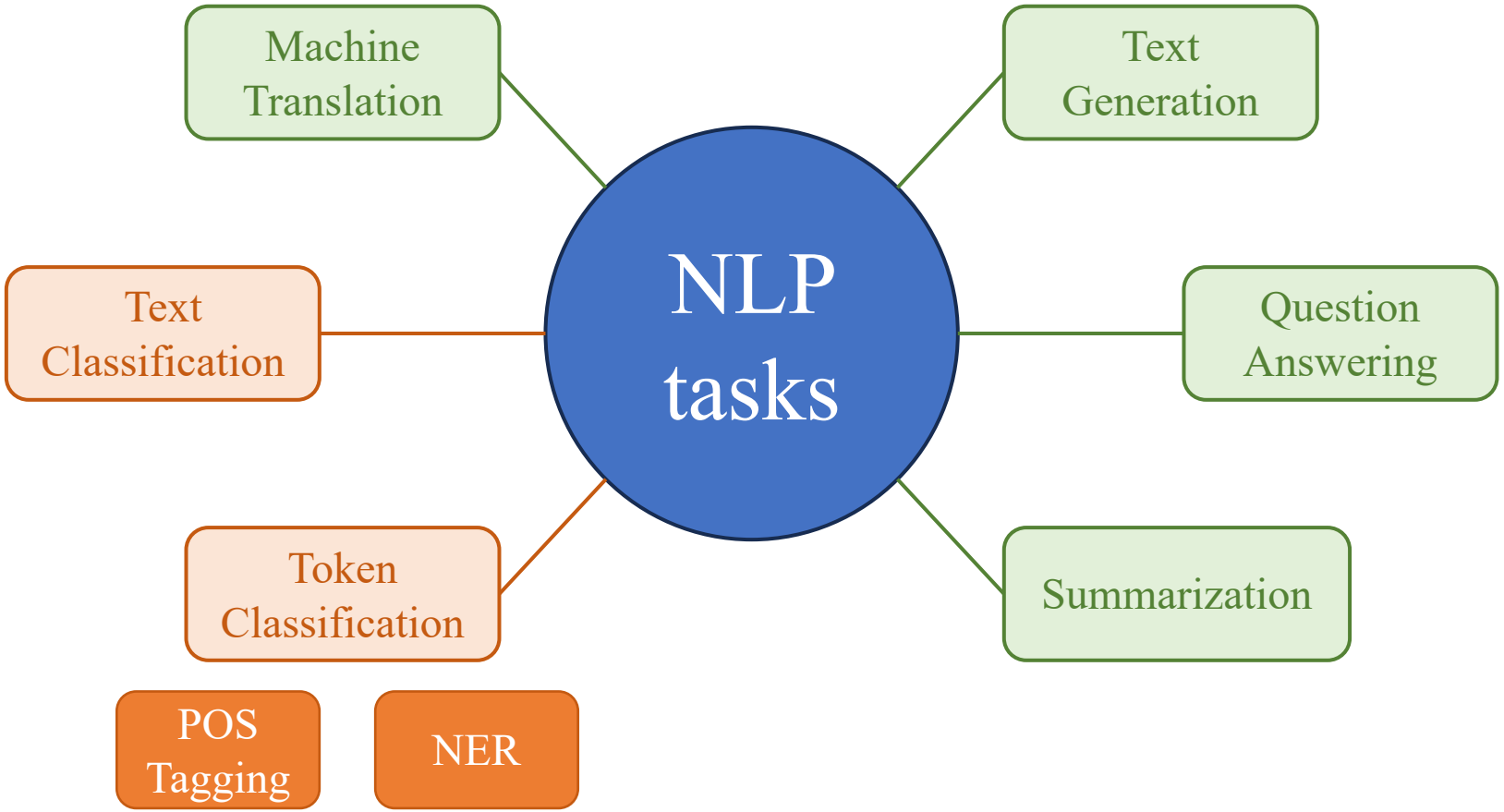
❖ Getting Started

Natural language processing (NLP) is a subfield of computer science and especially artificial intelligence. It is primarily concerned with providing computers with the ability to process data encoded in natural language and is thus closely related to information retrieval, knowledge representation and computational linguistics, a subfield of linguistics.



Introduction to NLP

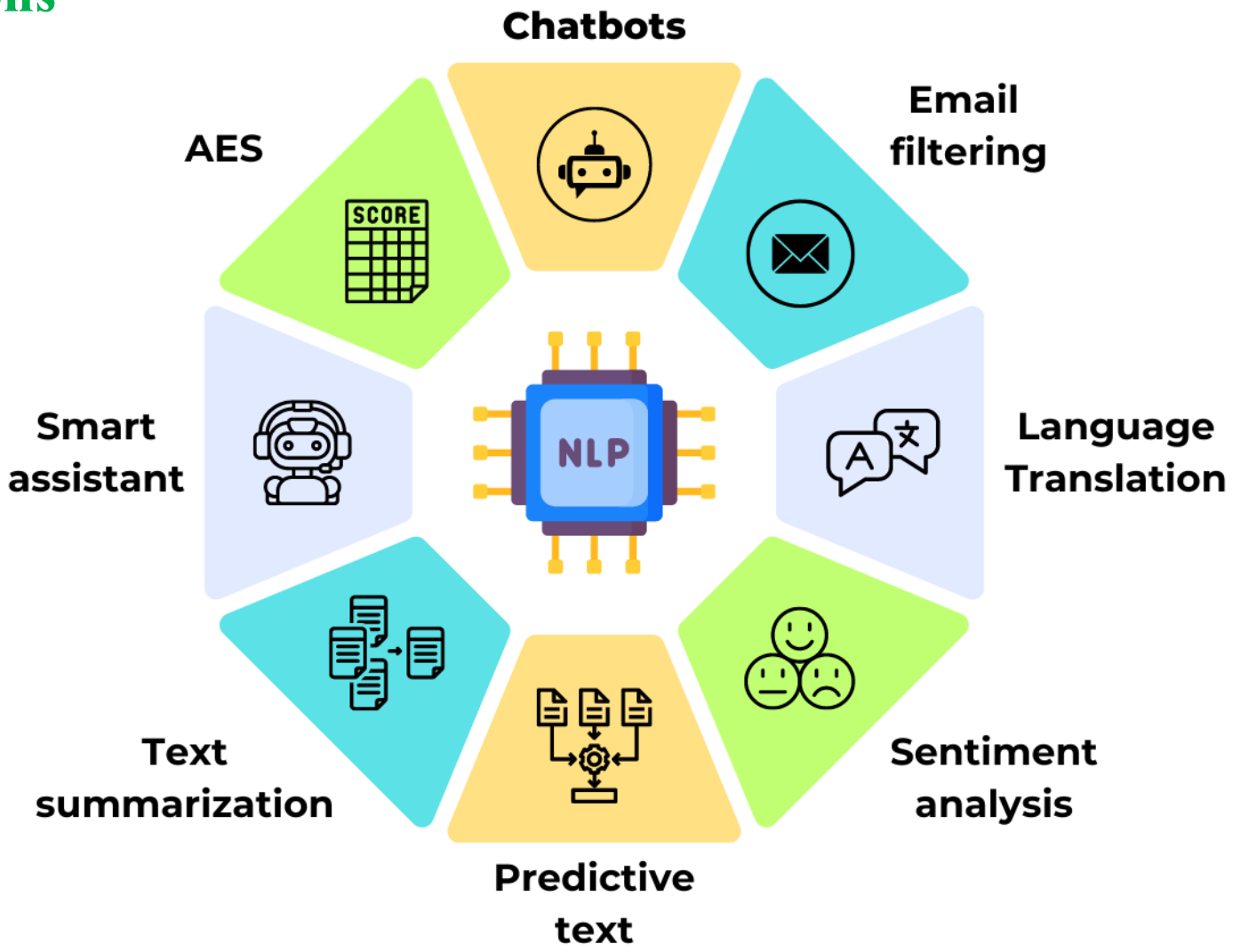
❖ NLP tasks



Ref: https://huggingface.co/tasks

Introduction to NLP

❖ NLP Applications



Introduction to NLP

❖ Translation

Source: Vietnamese

Machine Translation System

Hello

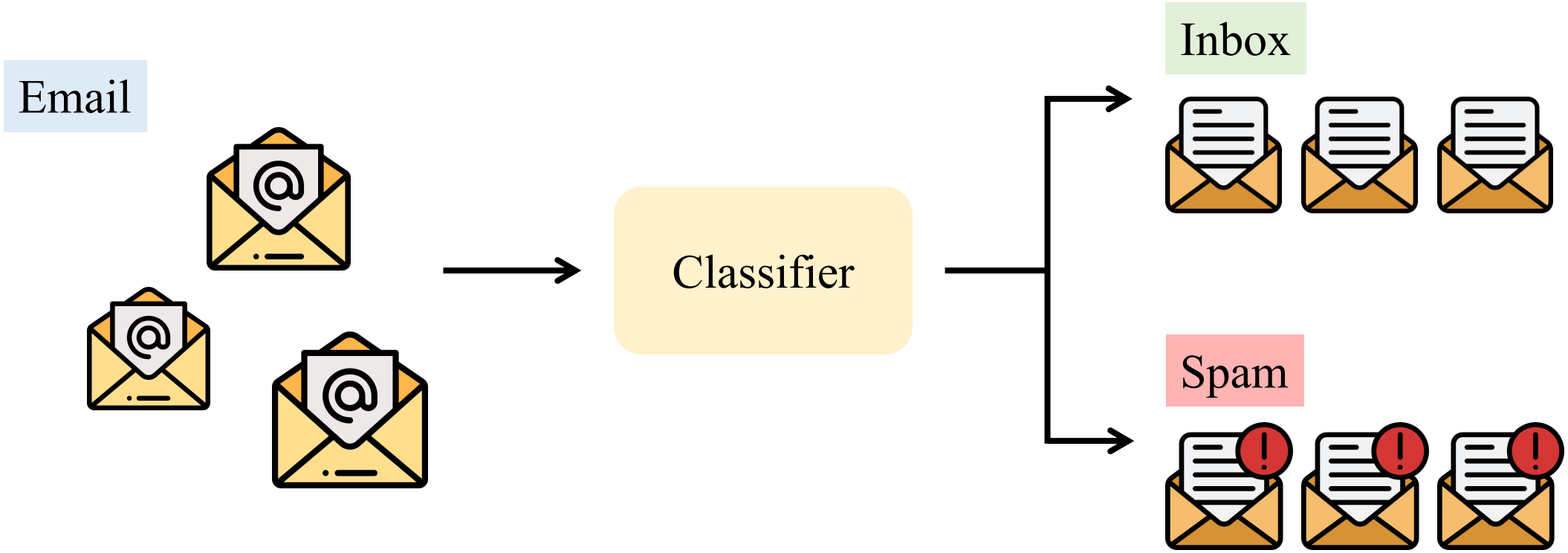
Target: English

こんにちは

Target: Japanese

Introduction to NLP

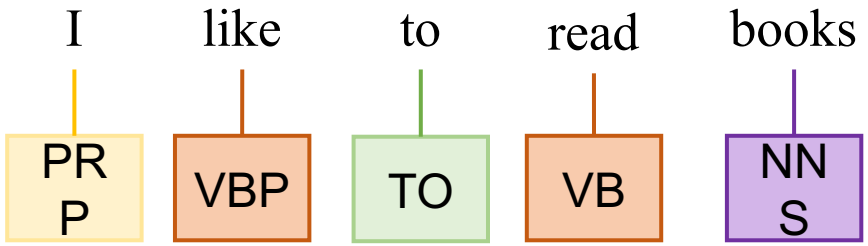
❖ Email filtering



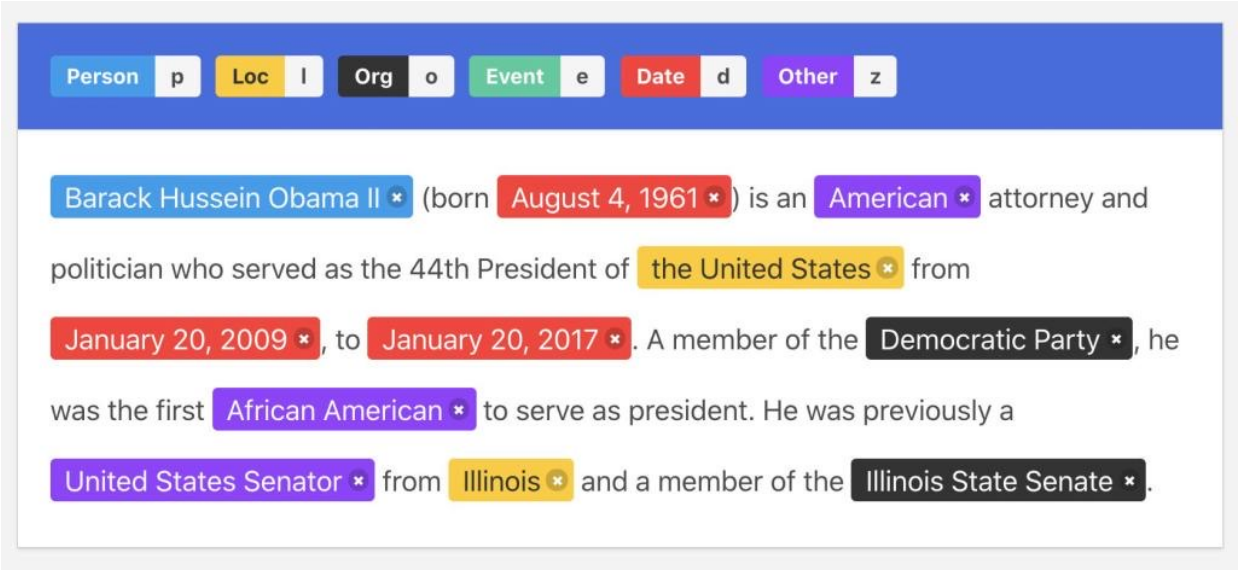
Introduction to NLP

❖ Information Extraction

Part of Speech Tagging (POS Tagging)



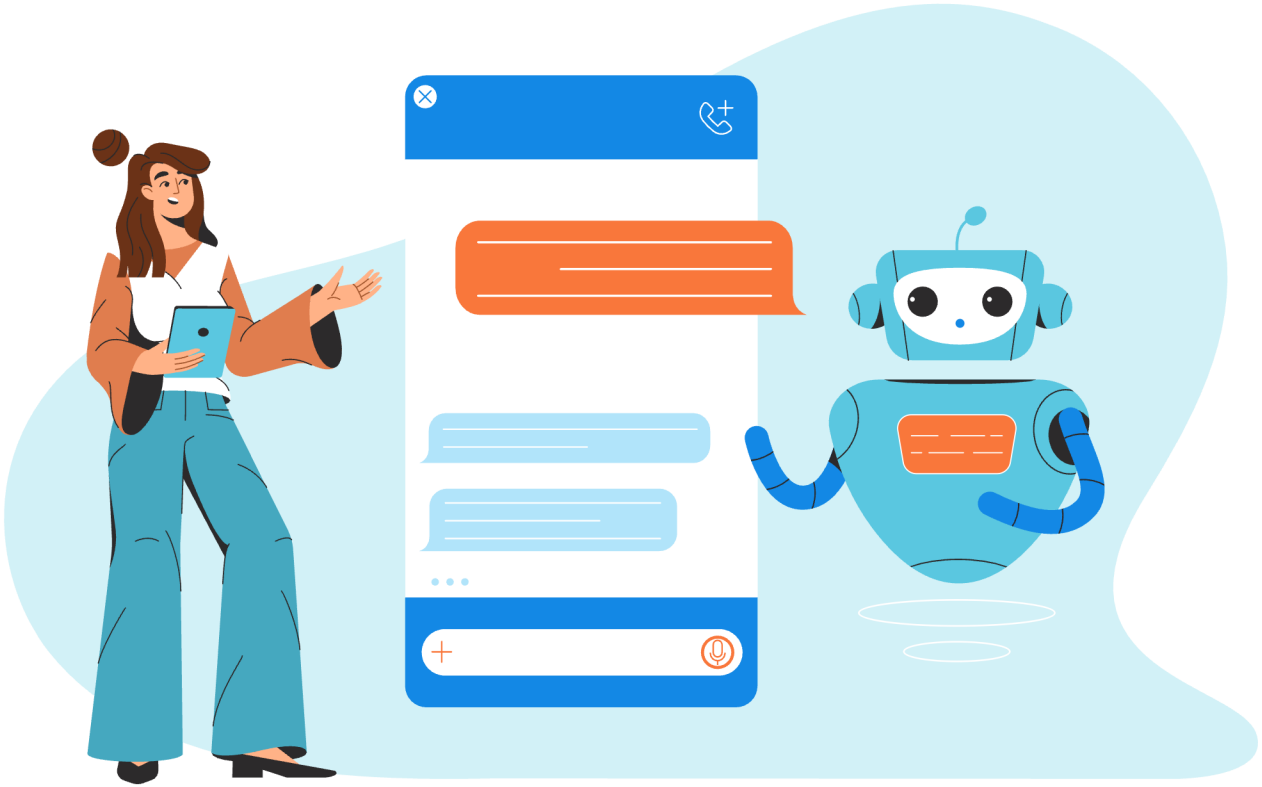
Named Entity Recognition (NER)



Introduction to NLP

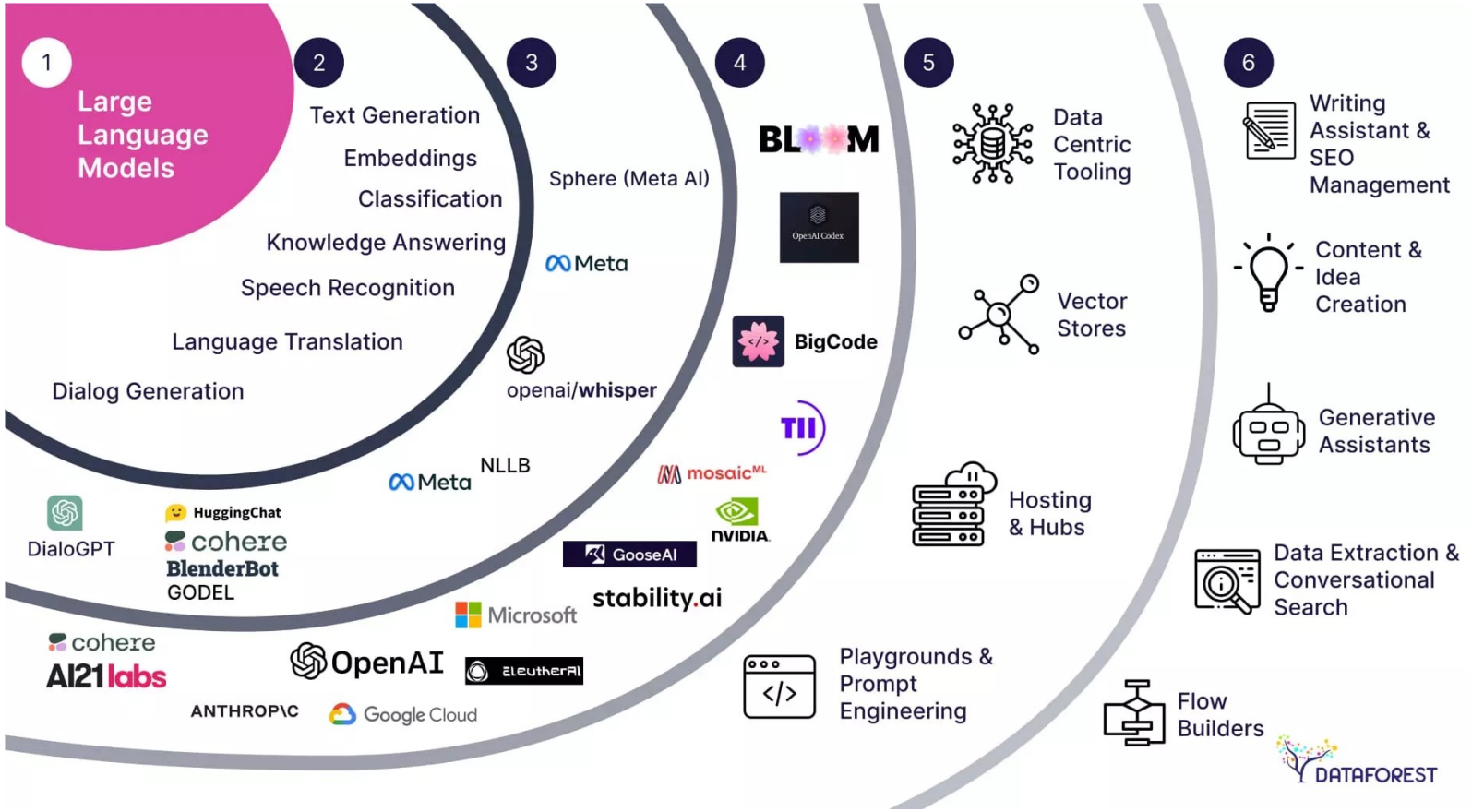
❖ Chatbot

Chatbot is a software application or web interface that is designed to mimic human conversation through text or voice interactions.



Introduction to NLP

❖ Large Language Models (LLMs)



A **large language model (LLM)** is a type of machine learning model designed for natural language processing tasks such as language generation. LLMs are language models with many parameters and are trained with self-supervised learning on a vast amount of text.

Introduction to NLP

❖ Large Language Models (LLMs)

GPT-4

LLaMA

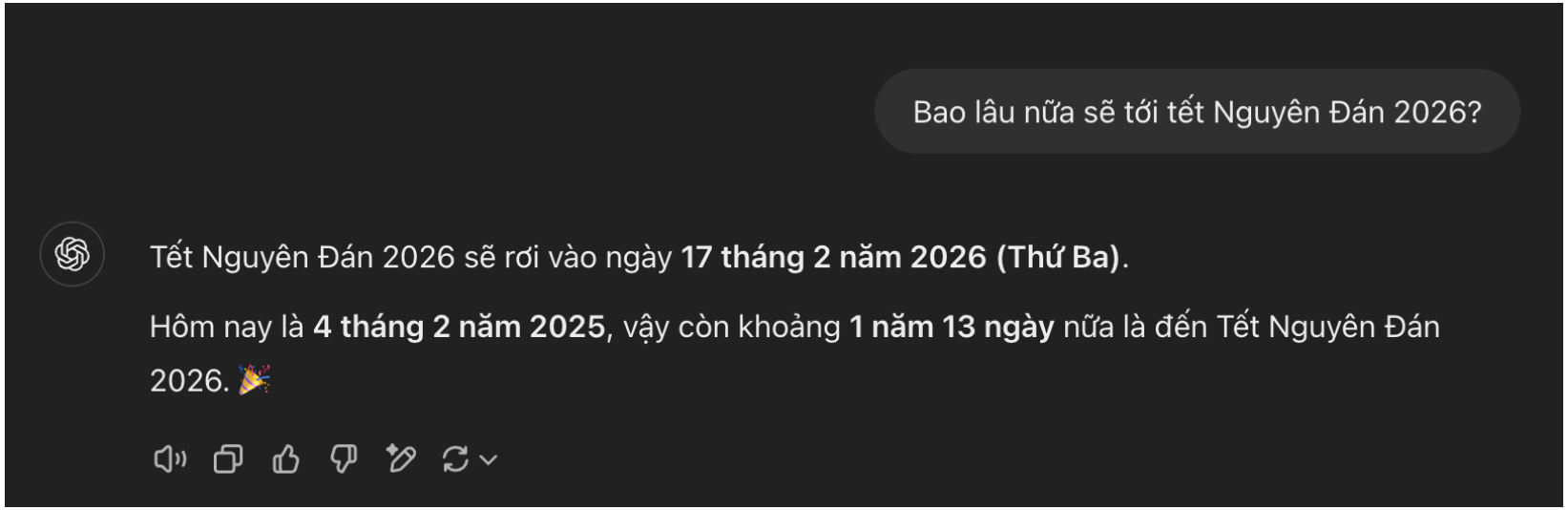
Qwen

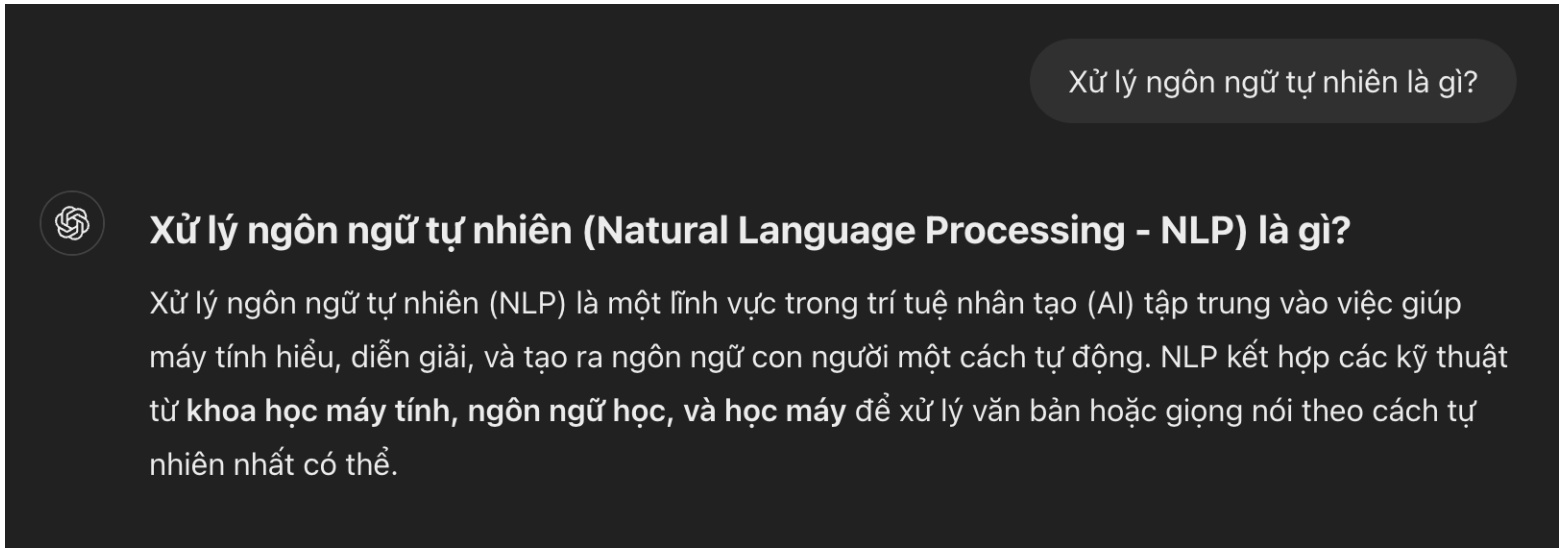
BERT

Claude

Introduction to NLP

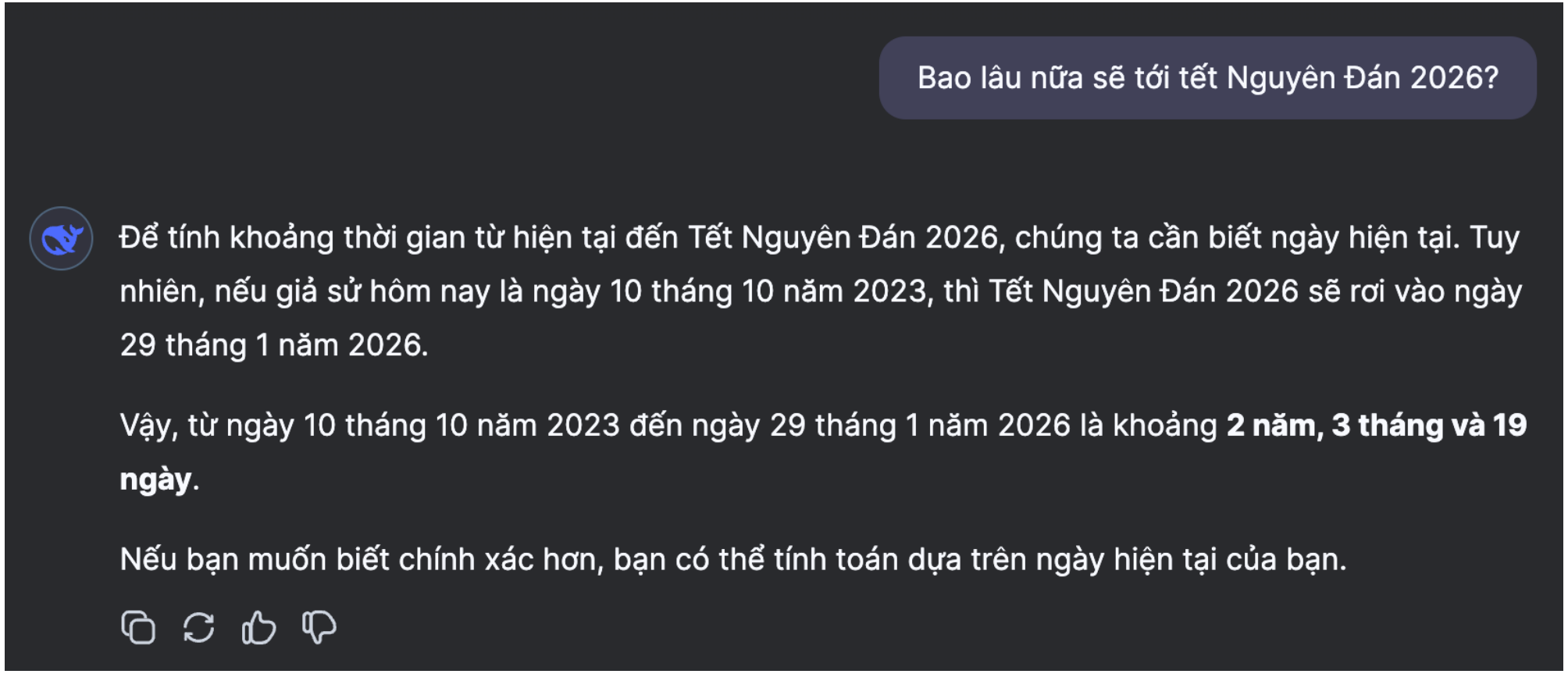
❖ LLM: ChatGPT





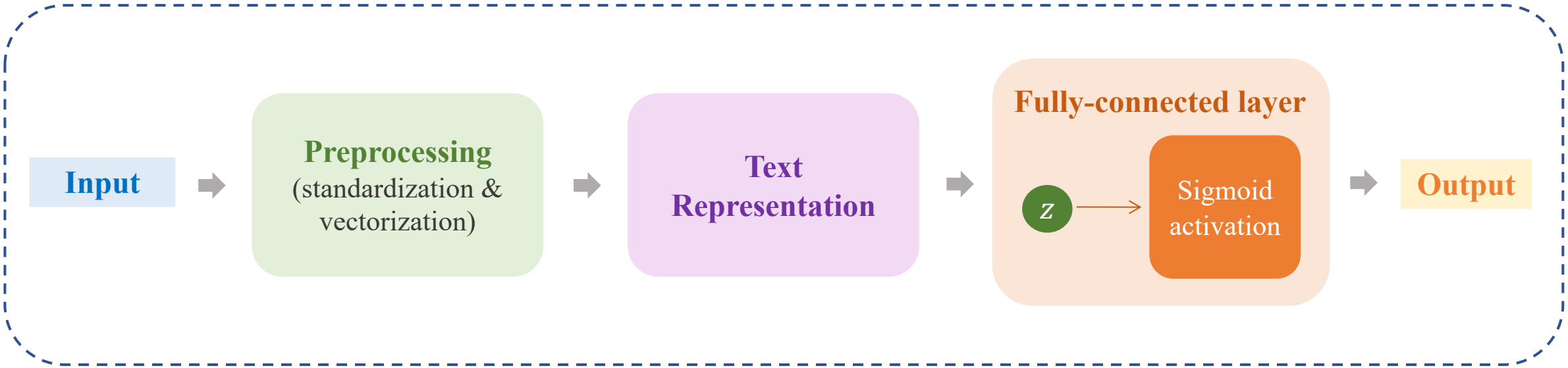
Introduction to NLP

❖ LLM: DeepSeek



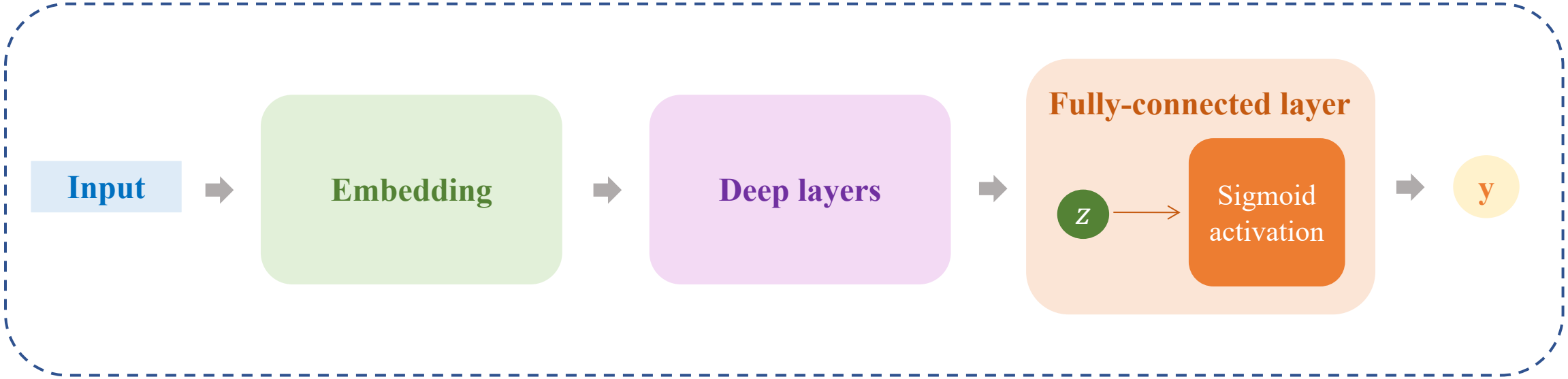
Introduction to NLP

❖ Basic NLP pipeline



Text Representation

❖ Sentence Classification



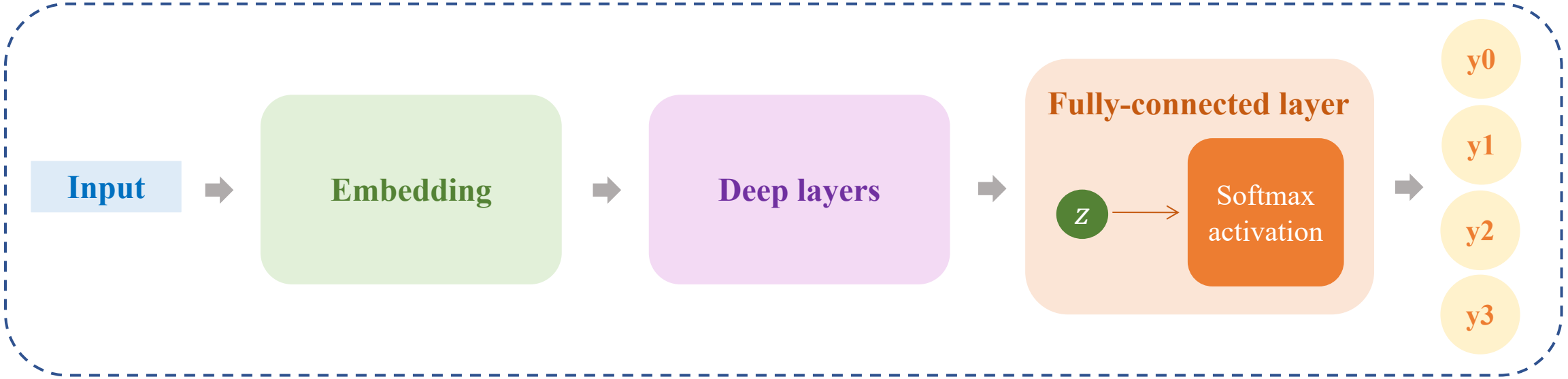
Text Classification

Sentiment Analysis

Spam/ Not Spam

Text Representation

❖ Token Classification



POS Tagging

Named-entity recognition

Text Classification

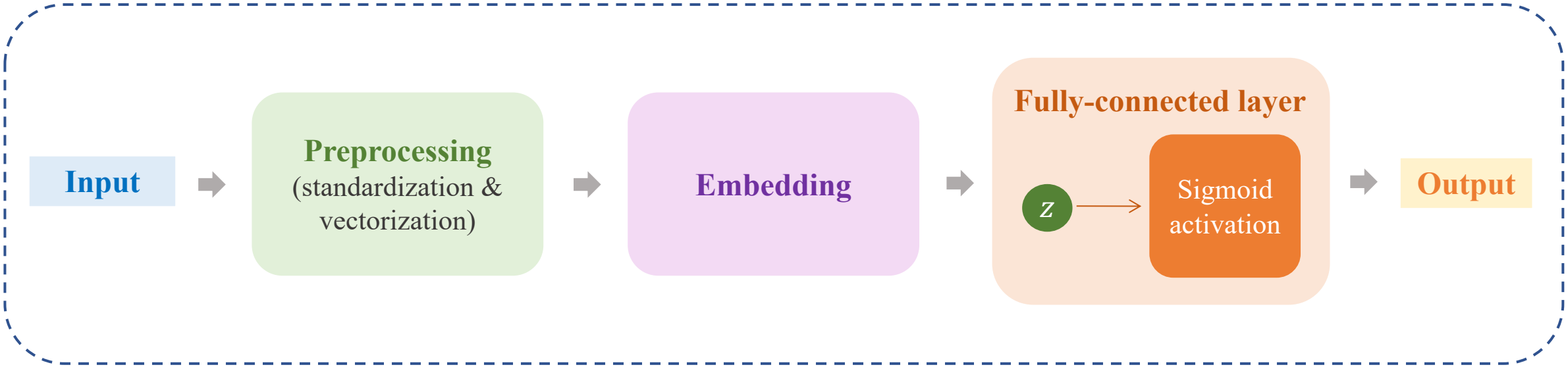
Code

Text Classification

- Preprocessing
- Embedding
- Text Classification

Text Classification

❖ Simple approach



Preprocessing

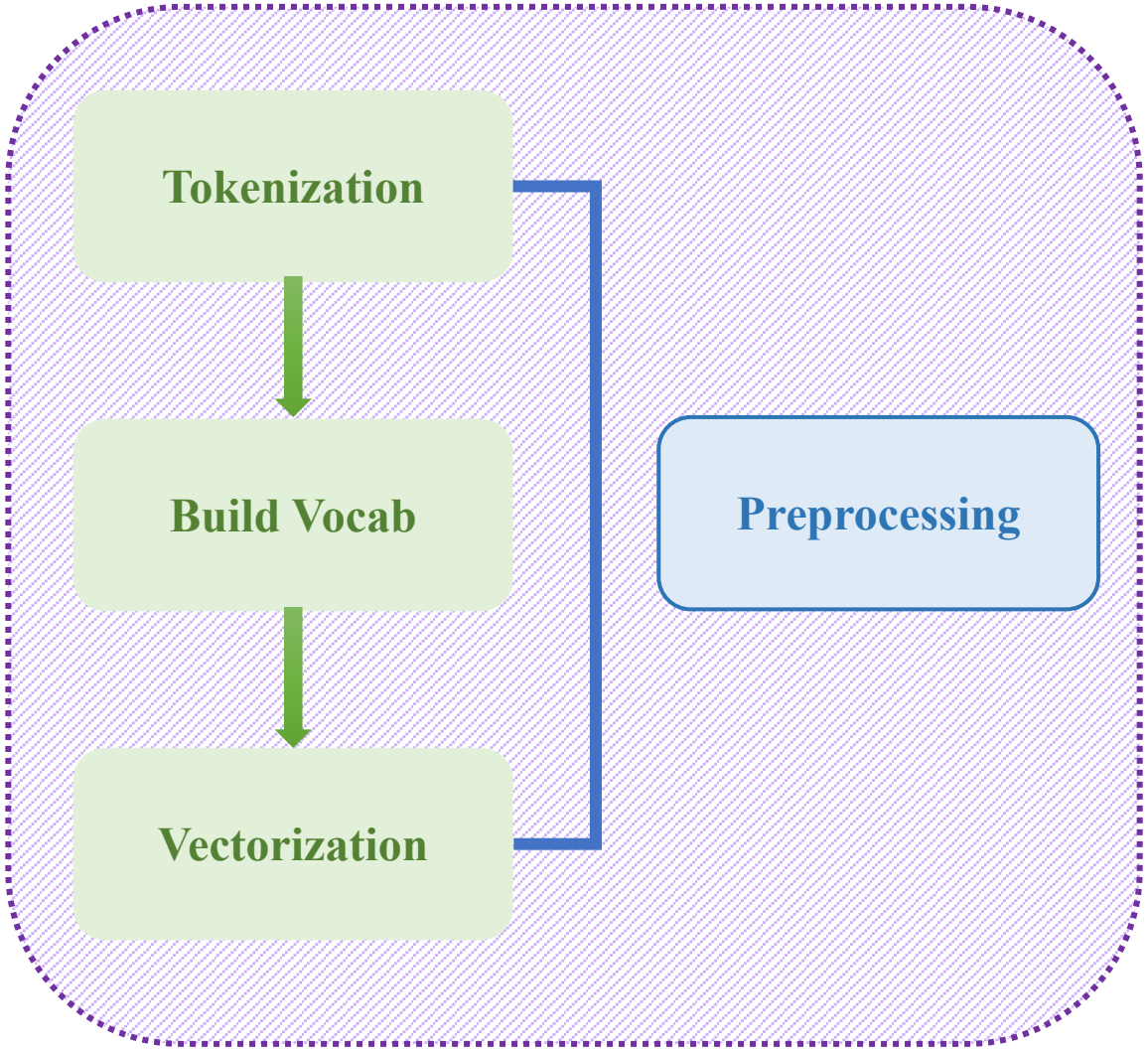
❖ Text Corpus

Deepseek sell off stock market

Deepseek release all their technical documents

```
corpus = [
    "Deepseek sell off stock market",
    "Deepseek release all their technical documents"
]
```

```
# 0: negative - 1: positive
labels = [0, 1]
```



Tokenization

Example corpus

sample1: ‘Deepseek sell off stock market’

sample2: ‘Deepseek release all their technical documents’

Deepseek sell off stock market

Deepseek release all their technical documents

Tokenize

Deepseek	sell	off	stock	market

Deepseek	release	all	their	technical	documents

```
from torchtext.data.utils import get_tokenizer
# version 0.17.0

sample1 = corpus[0]
sample2 = corpus[1]

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

print(sample1_tokens)
print(sample2_tokens)
```

```
['deepseek', 'sell', 'off', 'stock', 'market']
['deepseek', 'release', 'all', 'their', 'technical', 'documents']
```

Building Vocab

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

# Define tokenizer function
tokenizer = get_tokenizer('basic_english

# Create a function to yield list of tok
def yield_tokens(examples):
    for text in examples:
        yield tokenizer(text)

# Create vocabulary
vocab = build_vocab_from_iterator(yield_tokens(corpus),
                                  max_tokens=vocab_size,
                                  specials=["<unk>", "<pad>"])
vocab.set_default_index(vocab["<unk>"])
```

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

```
{'release': 7,
 'off': 6,
 'market': 5,
 'documents': 4,
 'deepseek': 2,
 '<pad>': 1,
 'all': 3,
 '<unk>': 0}
```

0	<unk>
1	<pad>
2	deepseek
3	all
4	documents
5	market
6	off
7	release

Vocabulary

Vectorization

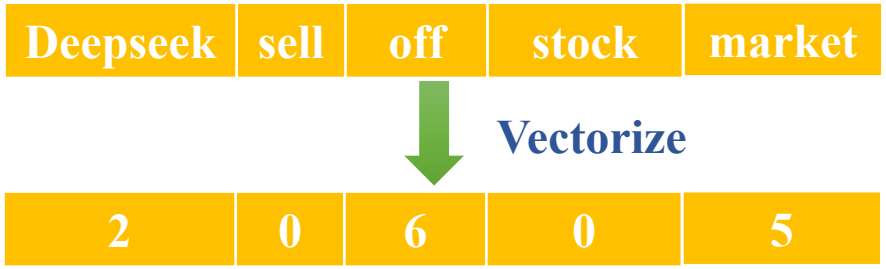
0	<unk>
1	<pad>
2	deepseek
3	all
4	documents
5	market
6	off
7	release

Vocabulary

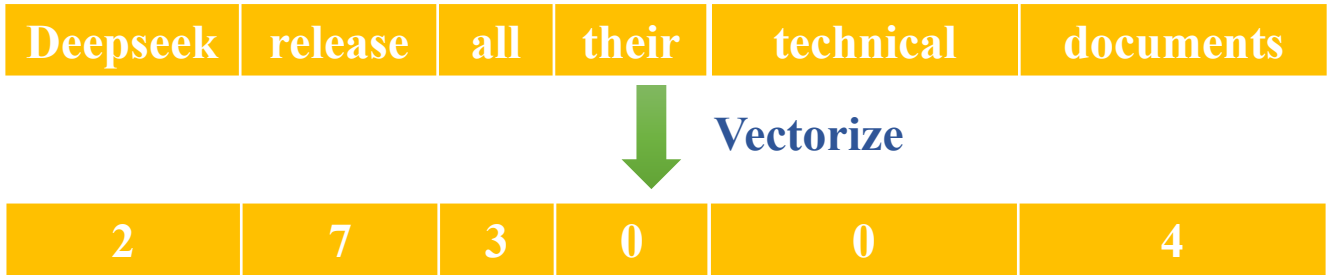
```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 0, 6, 0, 5]
[2, 7, 3, 0, 0, 4]
```

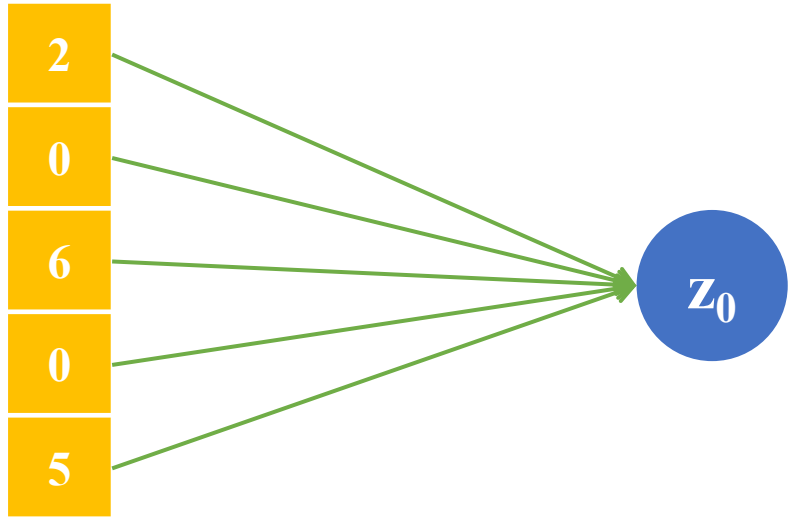


Problem?

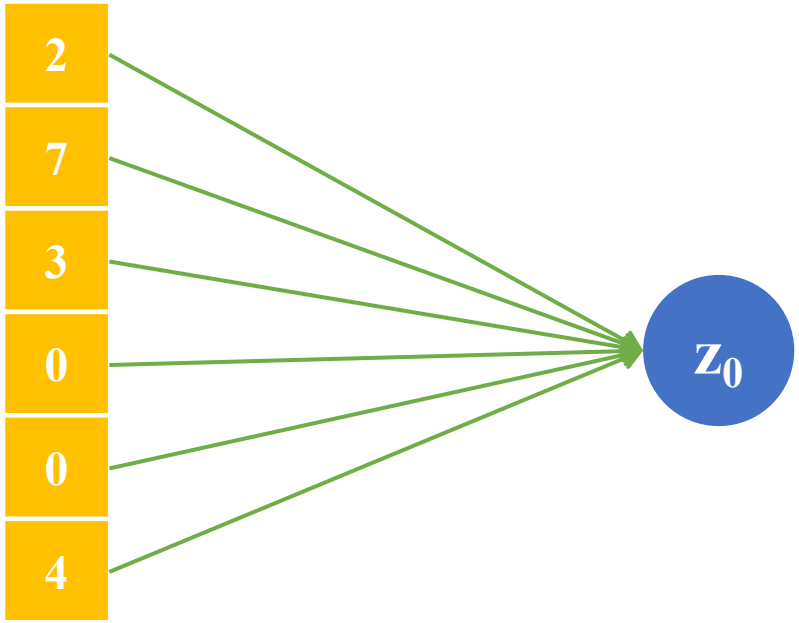


Vectorization

$$Z = X.W^T + b$$



(1, 5)

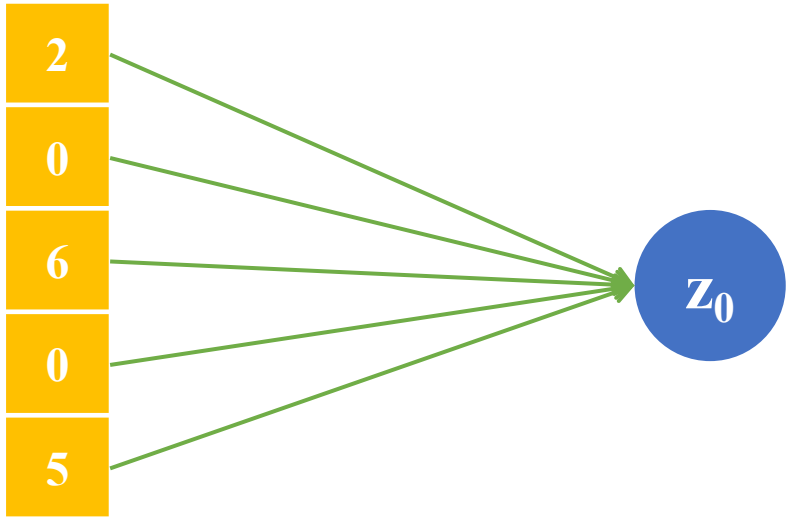


(1, 6)

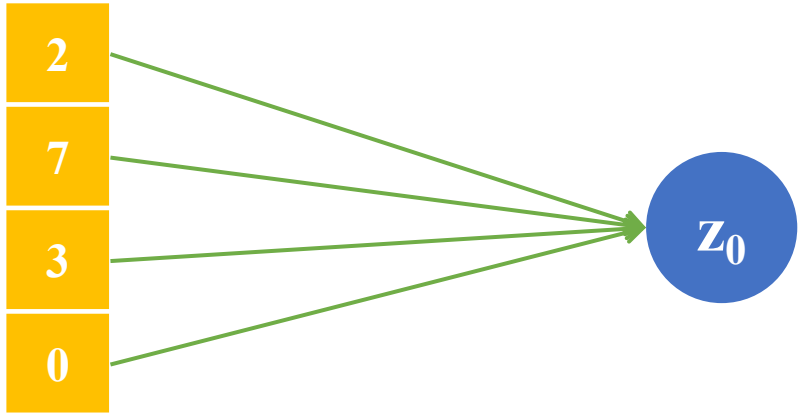
Shape W ?

Vectorization

$$Z = X.W^T + b$$



(1, 5)



(1, 4)

Shape W ?

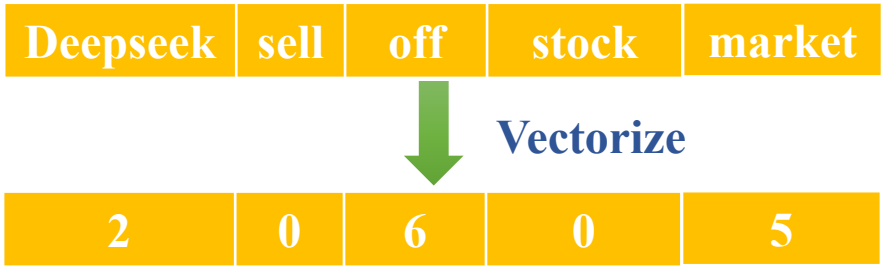
Vectorization

0	<unk>
1	<pad>
2	deepseek
3	all
4	documents
5	market
6	off
7	release

Vocabulary

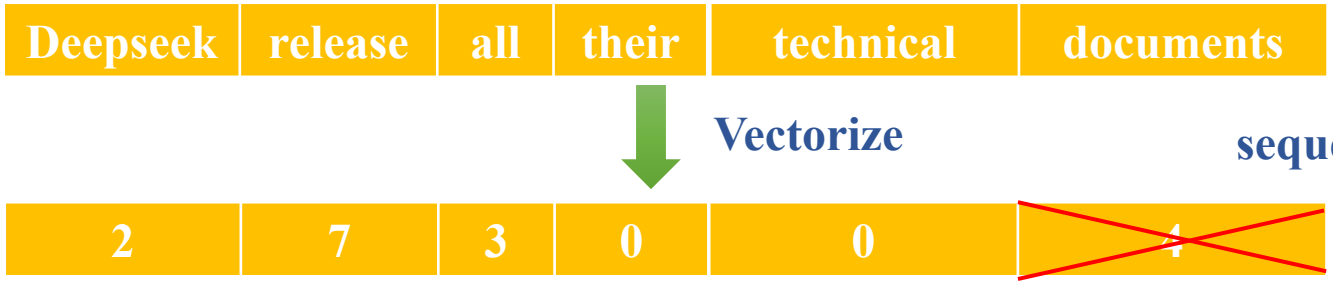
```python
# Tokenize and numericalize your samples
def vectorize(text, vocab, sequence_length):
    tokens = tokenizer(text)
    token_ids = [vocab[token] for token in tokens][:sequence_length]
    token_ids = token_ids + [vocab["<pad>"]] * (sequence_length - len(tokens))
    return torch.tensor(token_ids, dtype=torch.long)

# Vectorize the samples
corpus_ids = []
for sentence in corpus:
    corpus_ids.append(vectorize(sentence, vocab, sequence_length))
```




```python
for v in corpus_ids:
    print(v)
```

```
tensor([2, 0, 6, 0, 5])
tensor([2, 7, 3, 0, 0])
```



sequence_length = 5

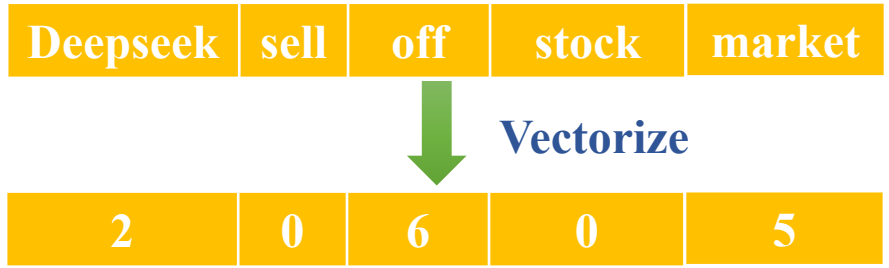
Vectorization

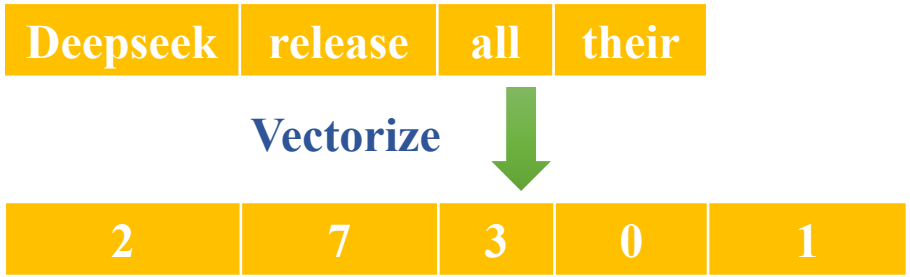
0	<unk>
1	<pad>
2	deepseek
3	all
4	documents
5	market
6	off
7	release

Vocabulary

```python
# Tokenize and numericalize your samples
def vectorize(text, vocab, sequence_length):
    tokens = tokenizer(text)
    token_ids = [vocab[token] for token in tokens][:sequence_length]
    token_ids = token_ids + [vocab["<pad>"]] * (sequence_length - len(tokens))
    return torch.tensor(token_ids, dtype=torch.long)

# Vectorize the samples
corpus_ids = []
for sentence in corpus:
    corpus_ids.append(vectorize(sentence, vocab, sequence_length))
```






```python
for v in corpus_ids:
    print(v)
```

```
tensor([2, 0, 6, 0, 5])
tensor([2, 7, 3, 0, 0])
```

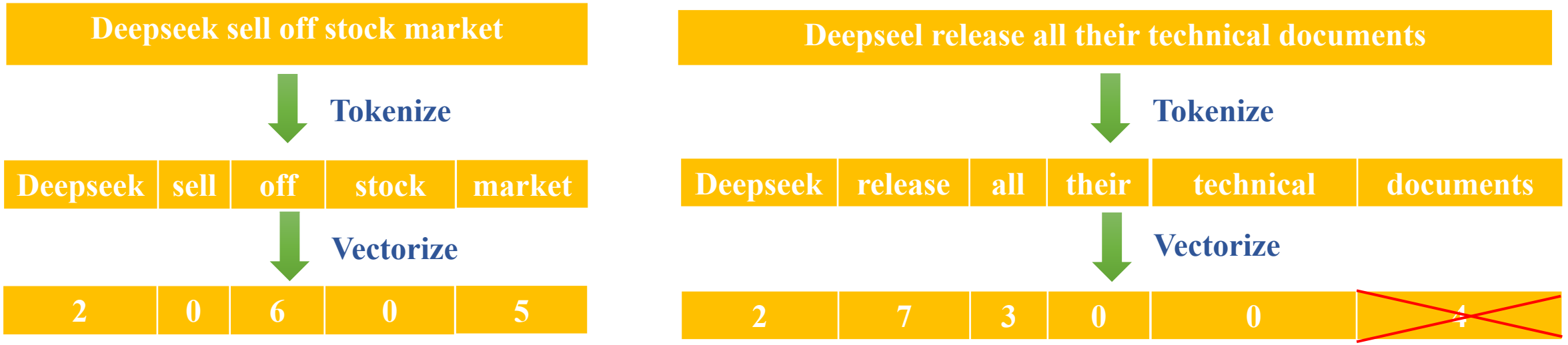
sequence_length = 5

Preprocessing

Example corpus

sample1: ‘Deepseek sell off stock market’

sample2: ‘Deepseek release all their technical documents’



Vocabulary

0	1	2	3	4	5	6	7
<unk>	<pad>	deepseek	all	documents	market	off	release

Embedding

Index	Token	
0	<unk>	[-0.1882, 0.5530]
1	<pad>	[1.7840, -0.8278]
2	deepseek	[1.0281, -1.9094]
3	all	[-1.3083, -1.9094]
4	documents	[0.2293, 1.3255]
5	market	[0.4058, -0.6624]
6	off	[0.5582, 0.0786]
7	release	[0.4309, -1.3067]

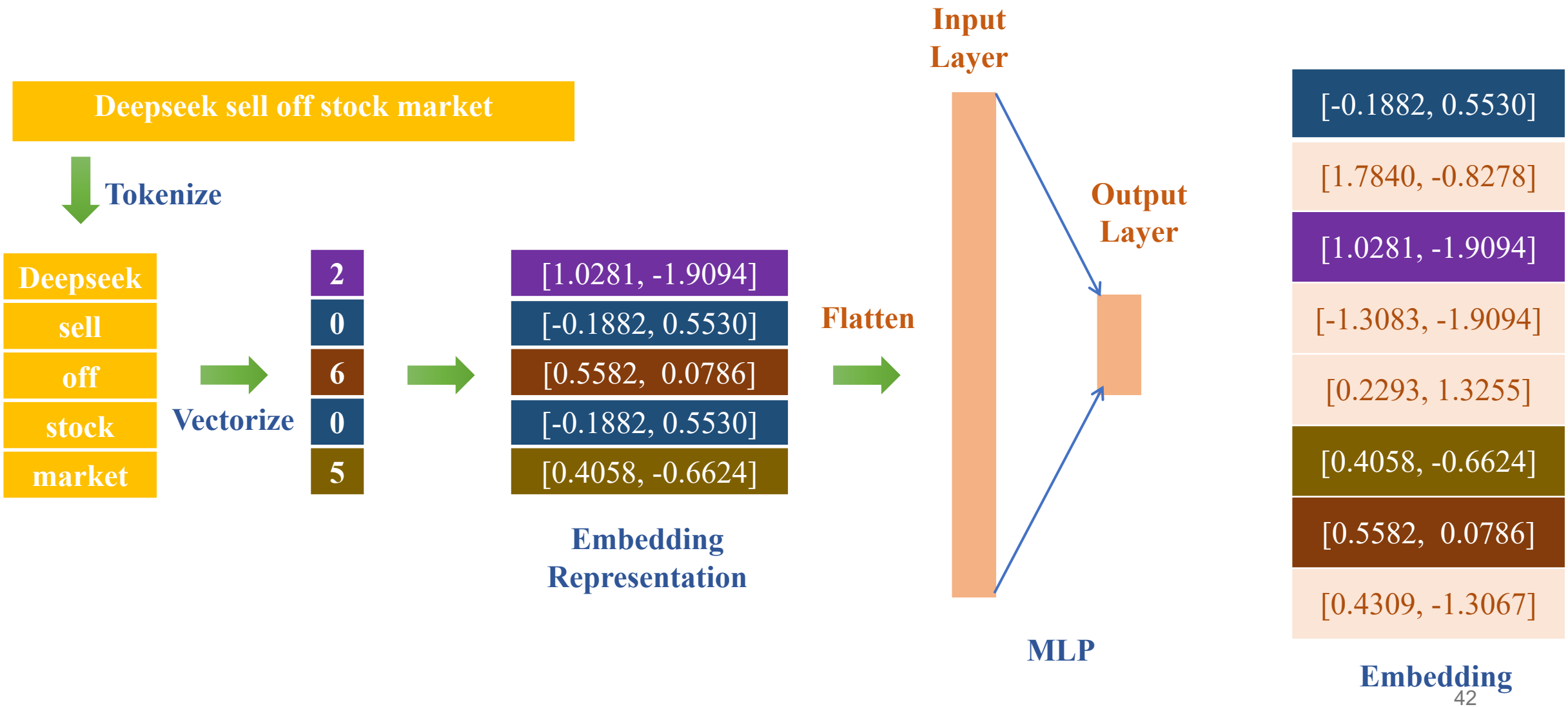
Embedding_dims = 2

```
vocab_size = 8
embedding_dim = 2
embedding = nn.Embedding(vocab_size, embedding_dim)
custom_weights = torch.tensor( [[-0.1882,  0.5530],
                                [ 1.7840, -0.8278],
                                [ 1.0281, -1.9094],
                                [-1.3083, -0.0987],
                                [ 0.2293,  1.3255],
                                [ 0.4058, -0.6624],
                                [ 0.5582,  0.0786],
                                [ 0.4309, -1.3067]]).float()
embedding.weight = nn.Parameter(custom_weights)
```

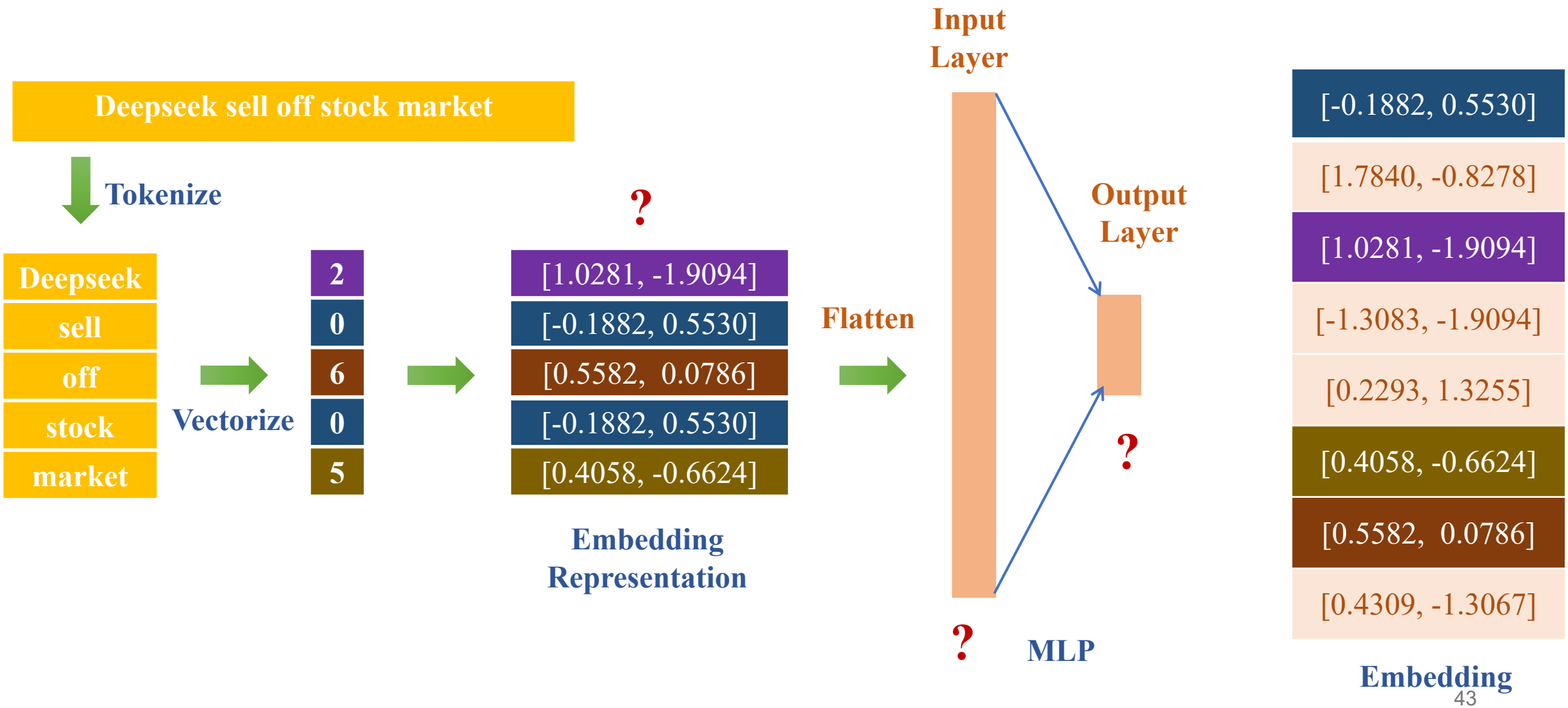
Initialization

AI VIETNAM
All-in-One Course
(TA Session)

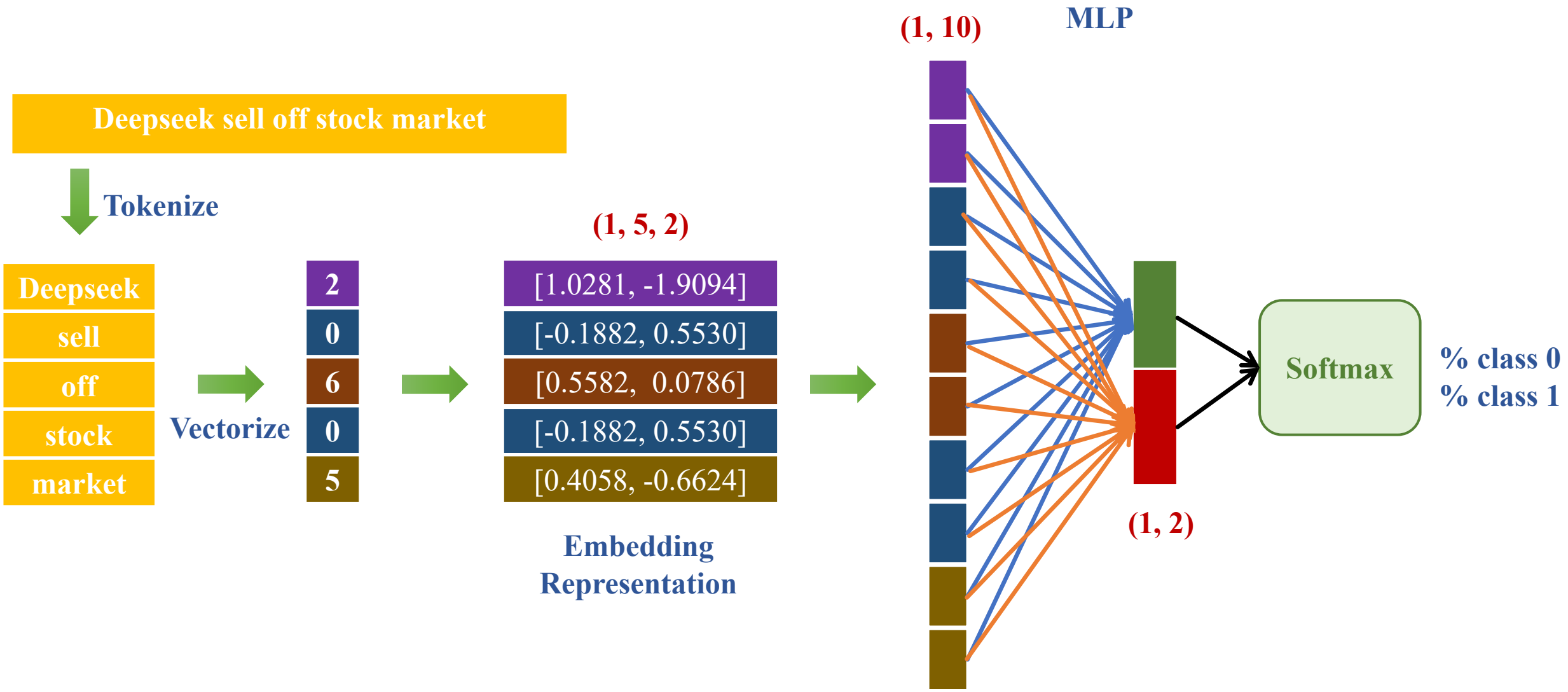
Text Classification



Text Classification



Text Classification



FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

0.1028] b_1

X

1.0281
-1.9094
-0.1882
0.5530
0.5582
0.0786
-0.1882
0.5530
0.4058
-0.6624

$XW_0 + b_0$

Z_0

$Z_0 = XW_0 + b_0 = [(1.0281 * 0.2108) + (-1.9094 * -0.0074) + (-0.1882 * 0.2760) + (0.5530 * 0.2325) + (0.5582 * -0.0518) + (0.0786 * -0.1876) + (-0.1882 * 0.0194) + (0.5530 * 0.0378) + (0.4058 * 0.0210) + (-0.6624 * 0.2982)] + -0.3049$

$= [?] + -0.3049 = ?$

FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

0.1028] b_1

X

X		W_0		
1.0281	⊙	0.2108	=	0.2167
-1.9094		-0.0074		0.0141
-0.1882		0.2760		-0.0519
0.5530		0.2325		0.1286
0.5582		-0.0518		-0.0289
0.0786		-0.1876		-0.0147
-0.1882		0.0194		-0.0037
0.5530		0.0378		0.0209
0.4058		0.0210		0.0085
-0.6624		0.2982		-0.1975

Sum = 0.0921 + **-0.3049** b_0 = Z_0

XW_0 b_0

FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

0.1028] b_1

X

1.0281
-1.9094
-0.1882
0.5530
0.5582
0.0786
-0.1882
0.5530
0.4058
-0.6624

Z_0

$Z_0 = XW_0 + b_0 = 0.9210 + -0.3049 = -0.2128$

FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

0.1028] b_1

X

1.0281
-1.9094
-0.1882
0.5530
0.5582
0.0786
-0.1882
0.5530
0.4058
-0.6624

$XW_1 + b_1$

Z_1

$Z_1 = XW_1 + b_1 = [(1.0281 * 0.0284) + (-1.9094 * 0.2968) + (-0.1882 * -0.0260) + (0.5530 * 0.1251) + (0.5582 * -0.0282) + (0.0786 * 0.0175) + (-0.1882 * -0.1817) + (0.5530 * 0.2483) + (0.4058 * 0.2338) + (-0.6624 * 0.2985)] + 0.1028$

$= [?] + 0.1028 = ?$

FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

0.1028] b_1

X

X	⊙	W_1	=
1.0281		0.0284	0.0292
-1.9094		0.2968	-0.5667
-0.1882		-0.0260	0.0049
0.5530		0.1251	0.0692
0.5582		-0.0282	-0.0157
0.0786		0.0175	0.0014
-0.1882		-0.1817	0.0342
0.5530		0.2483	0.1373
0.4058		0.2338	0.0949
-0.6624		0.2985	-0.1977

Sum = -0.4091 + 0.1028 (b_1) = Z_1

XW_1 b_1

FC Forward

W

[0.2108, -0.0074, 0.2760, 0.2325, -0.0518, -0.1876, 0.0194, 0.0378, 0.0210, 0.2982] W_0

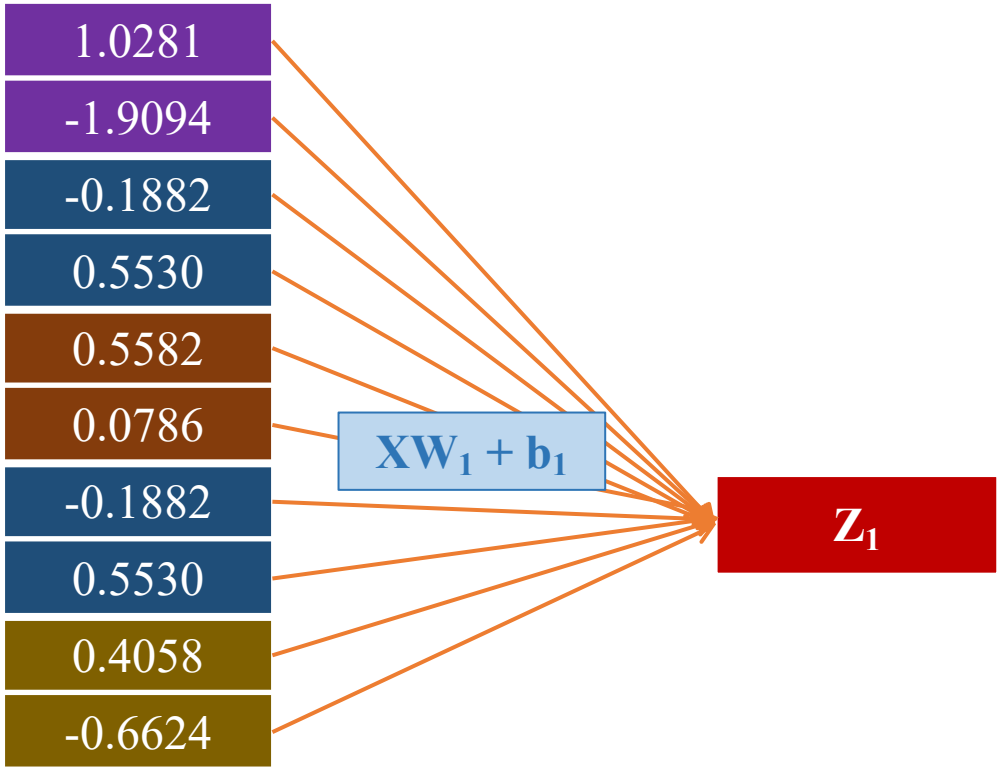
[0.0284, 0.2968, -0.0260, 0.1251, -0.0282, 0.0175, -0.1817, 0.2483, 0.2338, 0.2985] W_1

b

[-0.3049, b_0

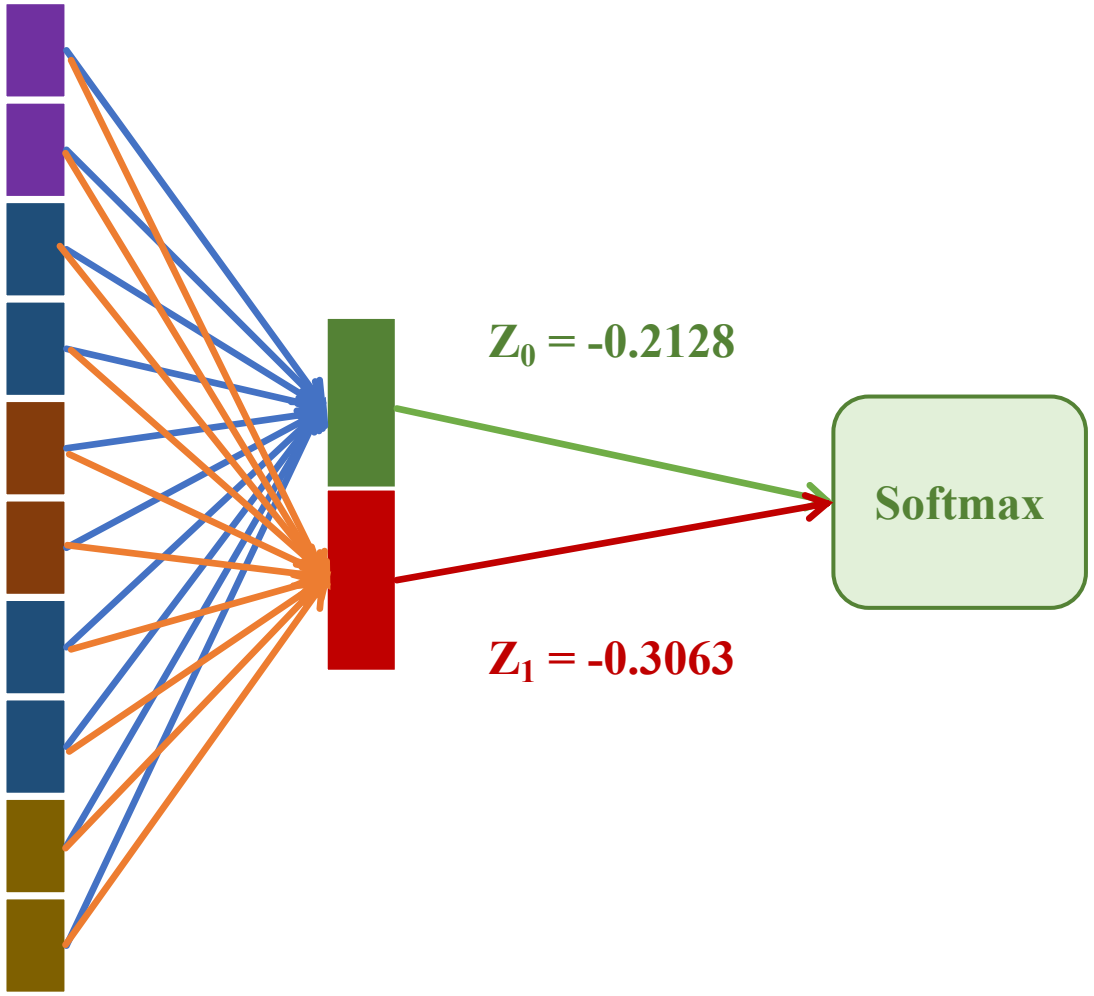
0.1028] b_1

X



$$Z_1 = XW_1 + b_1 = -0.4091 + 0.1028 = -0.3063$$

FC Forward



$$S(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}}$$

The picture can't be displayed.

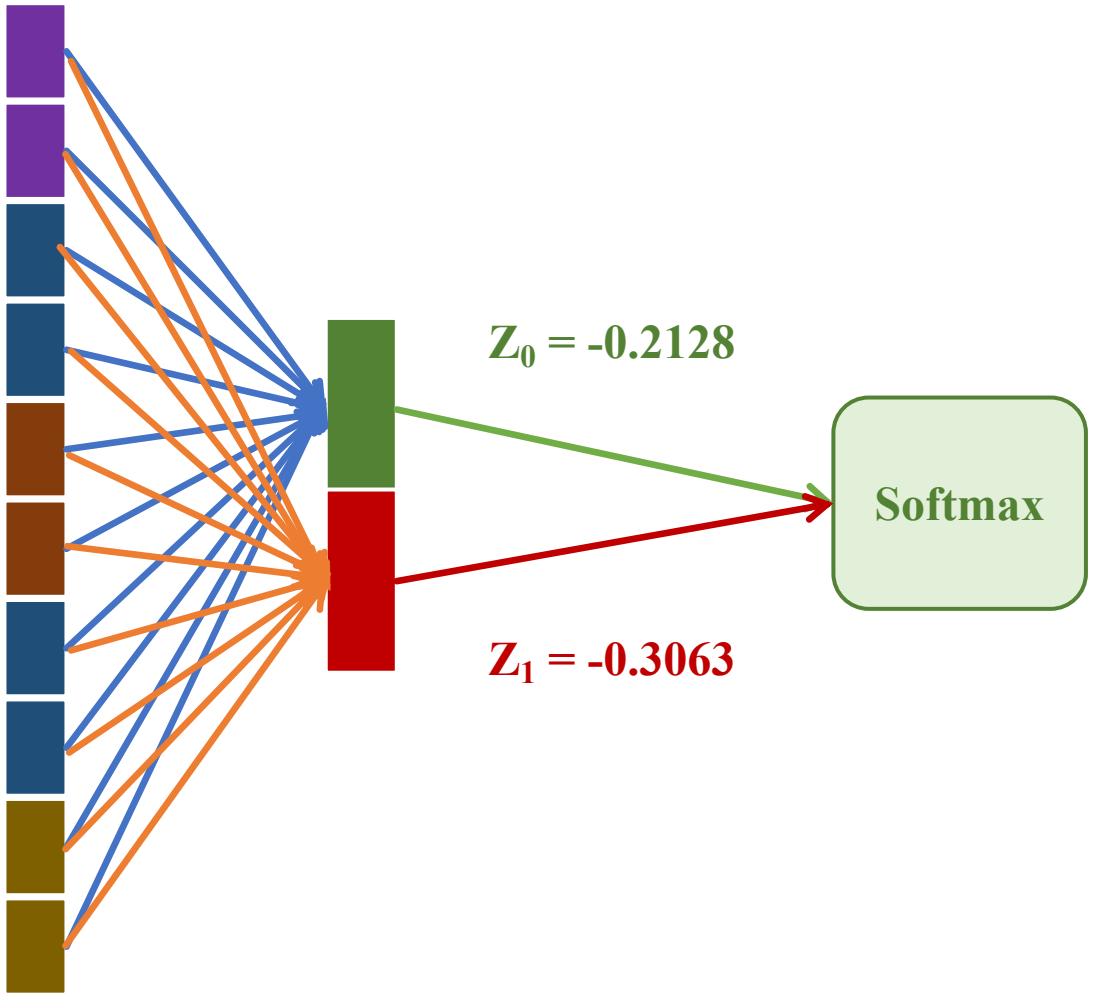
The picture can't be displayed.

The picture can't be displayed.

The picture can't be displayed.

The picture can't be displayed.

FC Forward



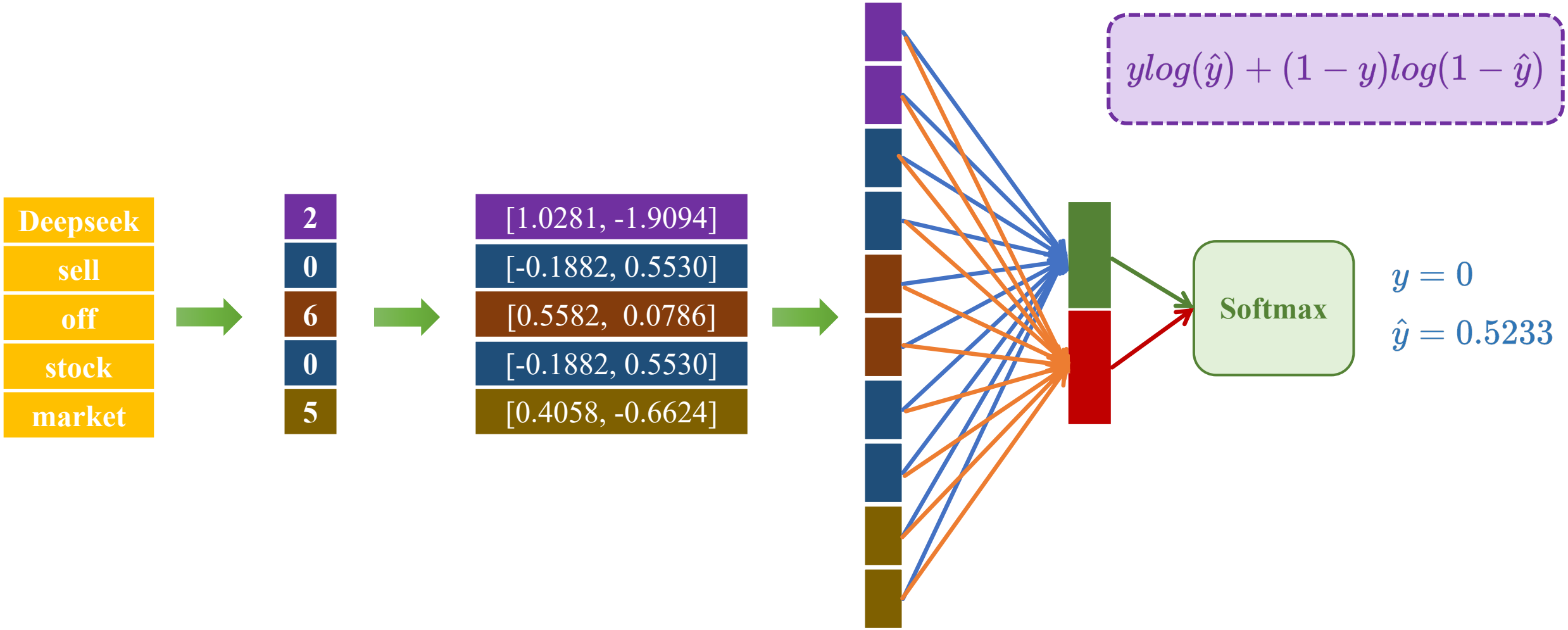
$$S(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}}$$

$$e^{Z_0} = e^{-0.2128} = 0.8083 \qquad e^{Z_1} = e^{-0.3063} = 0.7362$$

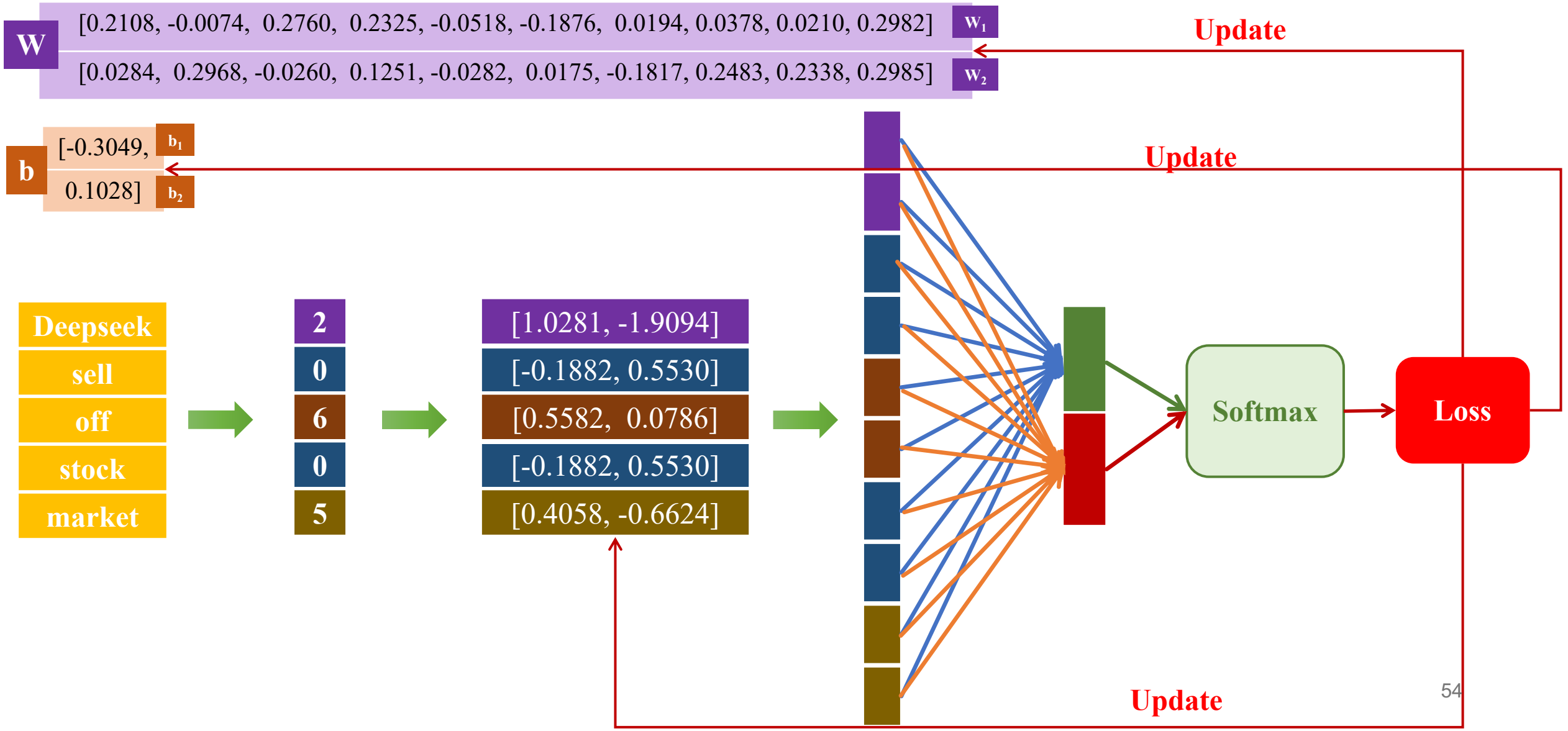
$$\sum_{j=1}^{k} e^{Z_j} = 0.8083 + 0.7362 = 1.5445$$

$$S_0 = \frac{0.8083}{1.5445} = 0.5233 \qquad S_1 = \frac{0.7361}{1.5445} = 0.4766$$

FCN Forward


Deepseek
sell
off
stock
market
2
0
6
0
5
[1.0281, -1.9094]
[-0.1882, 0.5530]
[0.5582, 0.0786]
[-0.1882, 0.5530]
[0.4058, -0.6624]
$ylog(\hat{y}) + (1-y)log(1-\hat{y})$
Softmax
$y = 0$
$\hat{y} = 0.5233$

Text Classification



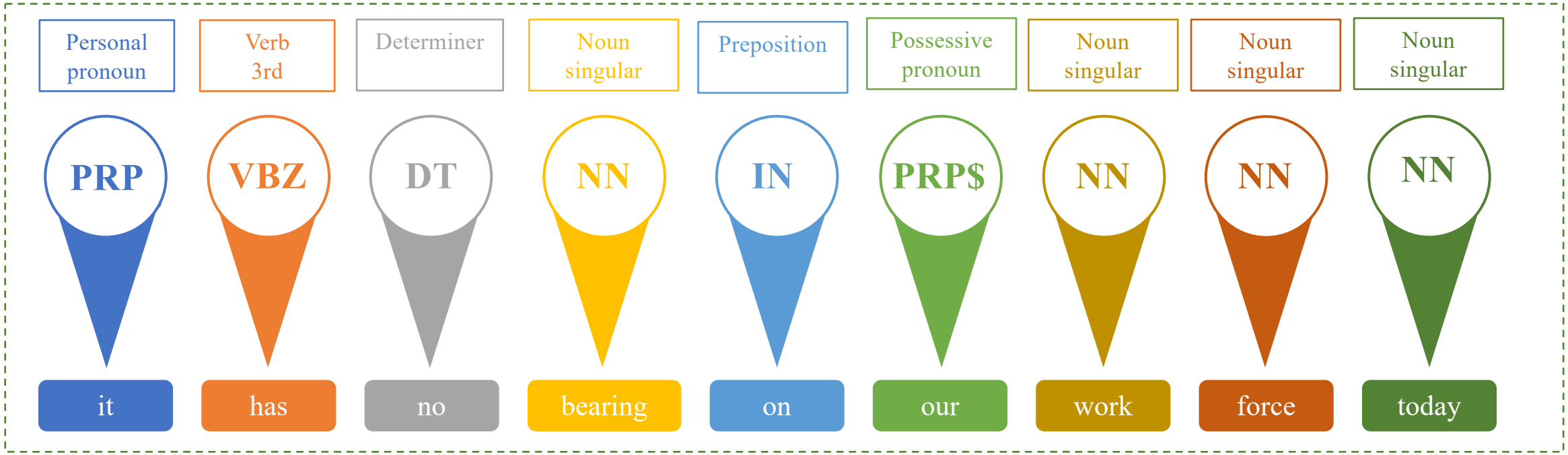
Part-of-Speech Tagging

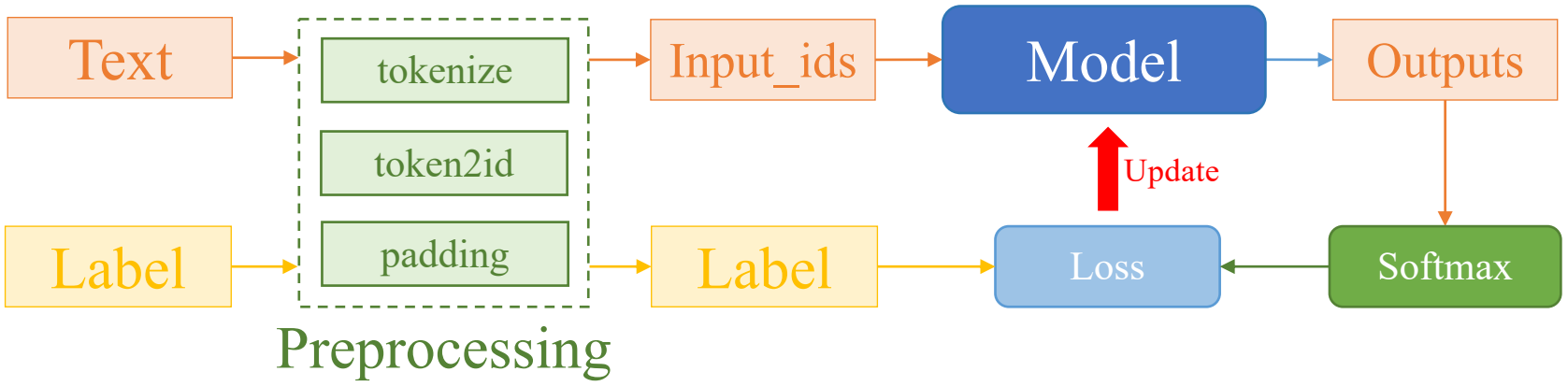
Code

Part-of-Speech Tagging

- Preprocessing
- Embedding
- POS Tagging

Part-of-Speech Tagging





Preprocessing

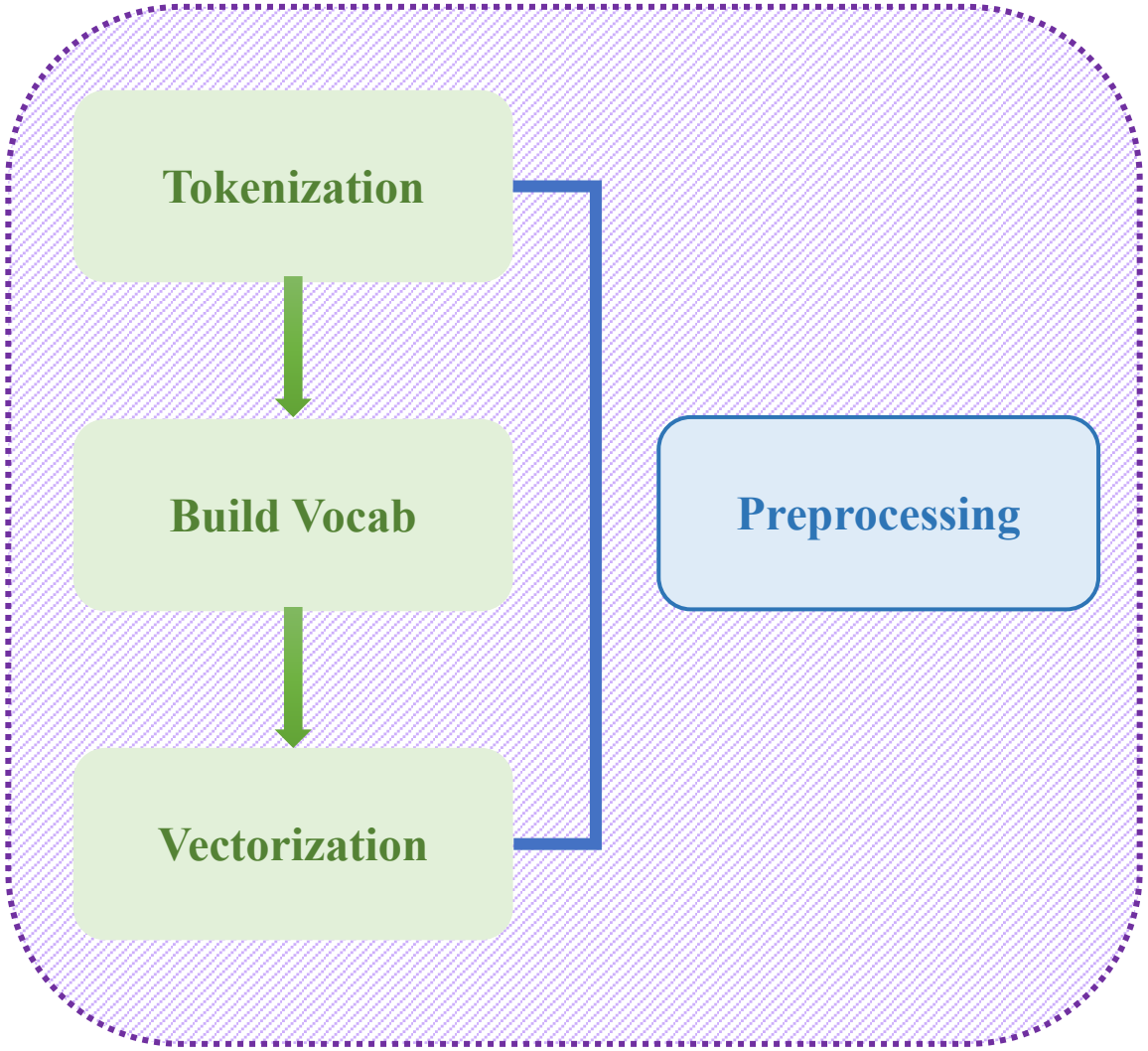
❖ Text Corpus

AI is growing fast

Models are large

```
corpus = [
    "AI is growing fast",
    "Models are large"
]
```

```
# 0: noun/pronoun - 1: verb - others - 2
labels = [[0, 1, 1, 2],
          [0, 1, 2]]
```

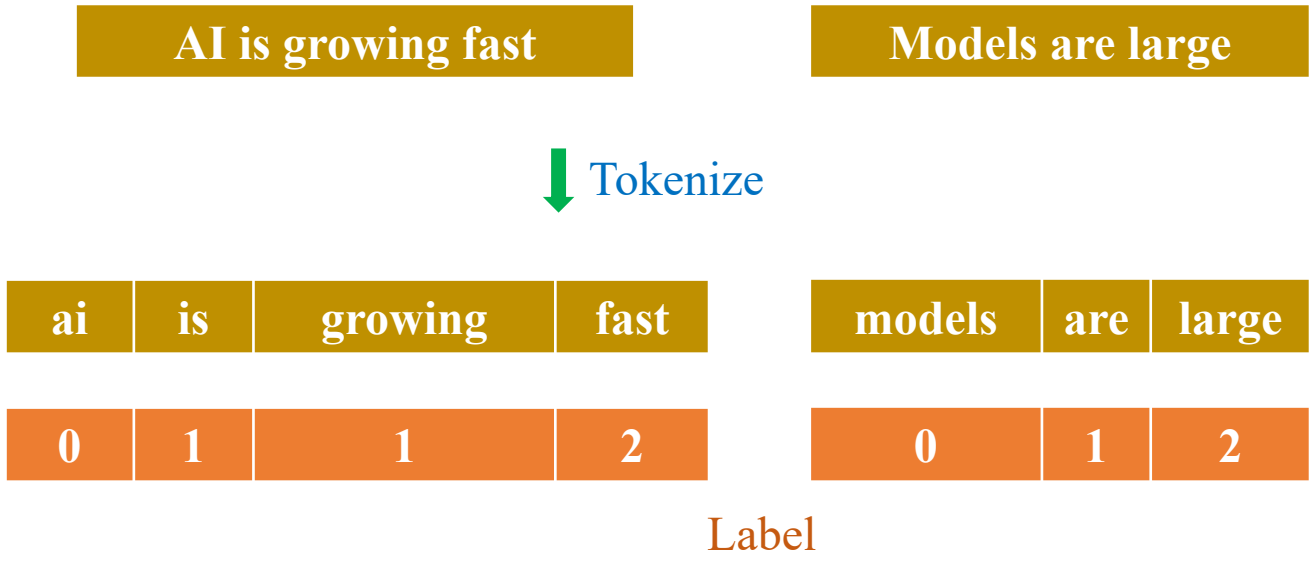


Tokenization

Example corpus

sample1: 'AI is growing fast'

sample2: 'Models are large'




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

sample1 = corpus[0]
sample2 = corpus[1]

# Define tokenizer
tokenizer = get_tokenizer('basic_english')

# Tokenize sample 1
sample1_tokens = tokenizer(sample1)
print(sample1_tokens)

# Tokenize sample 2
sample2_tokens = tokenizer(sample2)
print(sample2_tokens)
```

```
['ai', 'is', 'growing', 'fast']
['models', 'are', 'large']
```

Building Vocab

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

# Define tokenizer
tokenizer = get_tokenizer('basic_english')

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

# Create vocabulary
vocab = build_vocab_from_iterator(yield_tokens(corpus),
                                  max_tokens=vocab_size,
                                  specials=["<unk>", "<pad>"])
vocab.set_default_index(vocab["<unk>"])
```

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

```
{'is': 6,
 'growing': 5,
 'models': 8,
 'are': 3,
 'ai': 2,
 'fast': 4,
 '<pad>': 1,
 'large': 7,
 '<unk>': 0}
```

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary

Vectorization

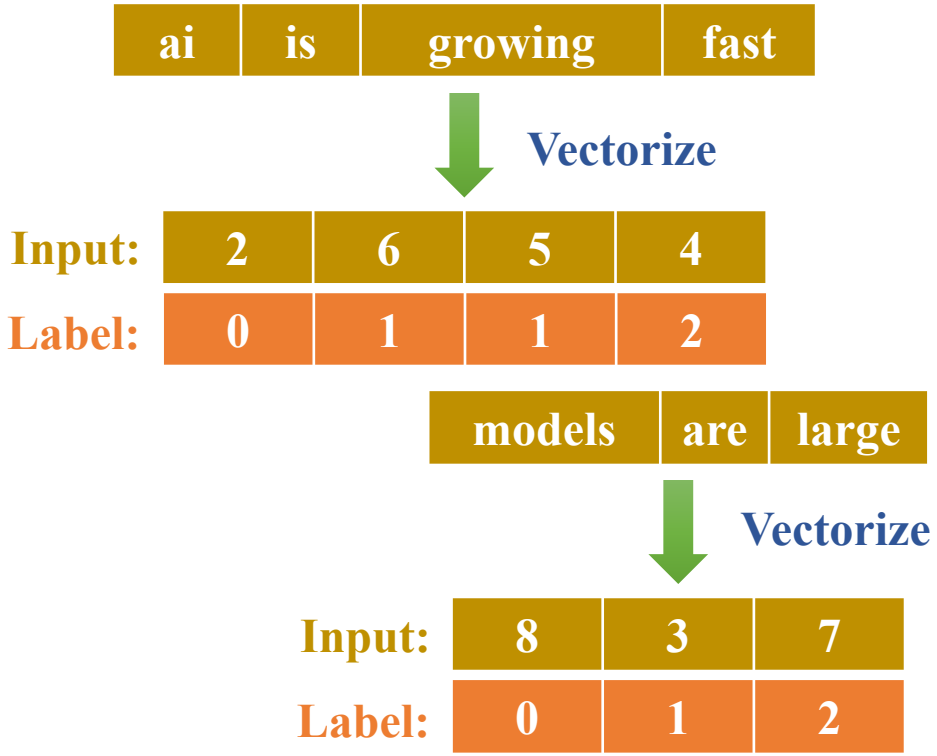
0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary

```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 6, 5, 4]
[8, 3, 7]
```



Vectorization

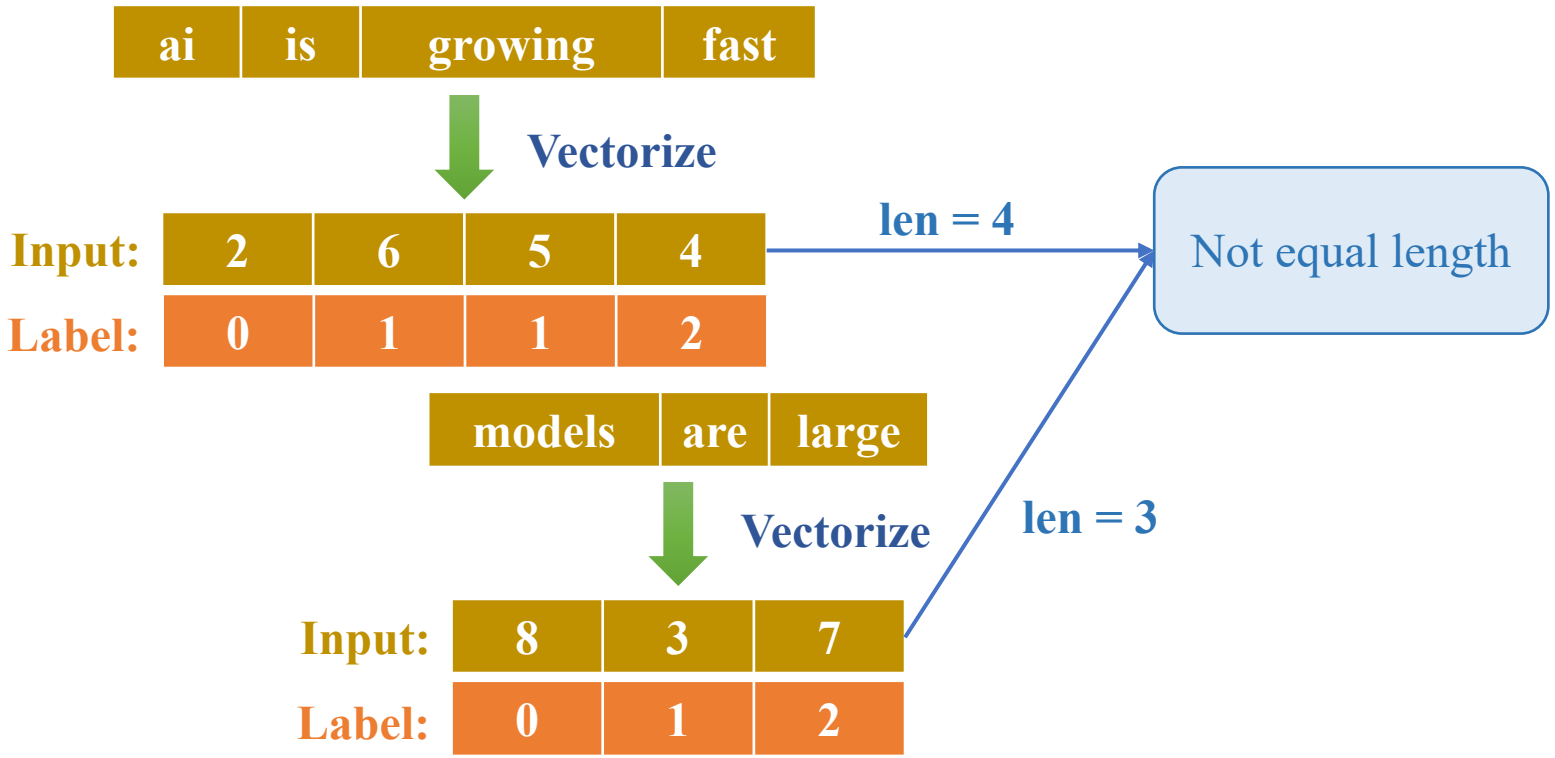
0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary

```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 6, 5, 4]
[8, 3, 7]
```



Vectorization

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

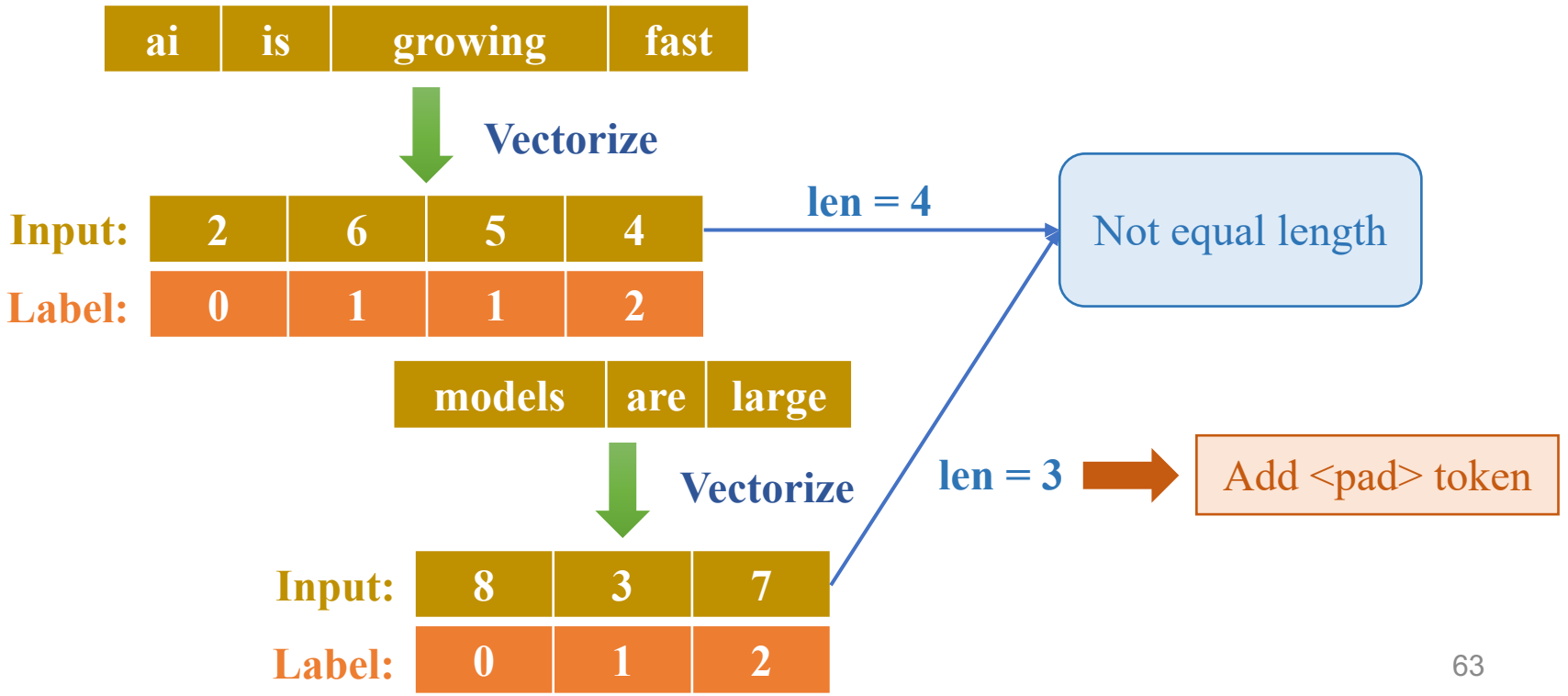
Vocabulary

```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 6, 5, 4]
[8, 3, 7]
```

sequence_length = 4



Vectorization

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

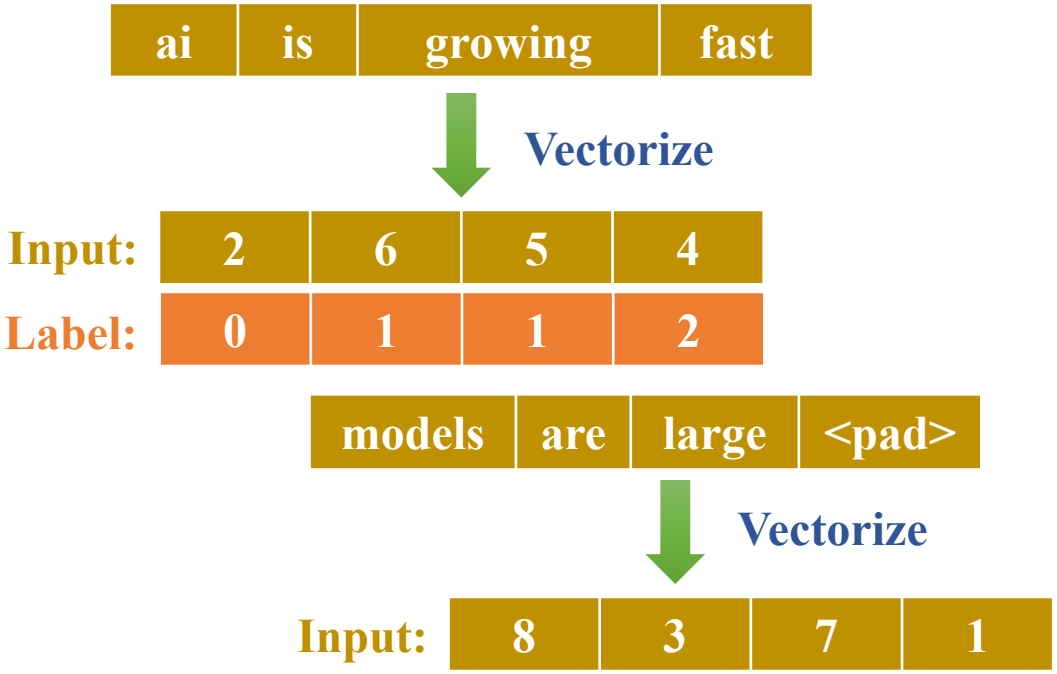
Vocabulary

```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 6, 5, 4]
[8, 3, 7]
```

sequence_length = 4



Vectorization

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

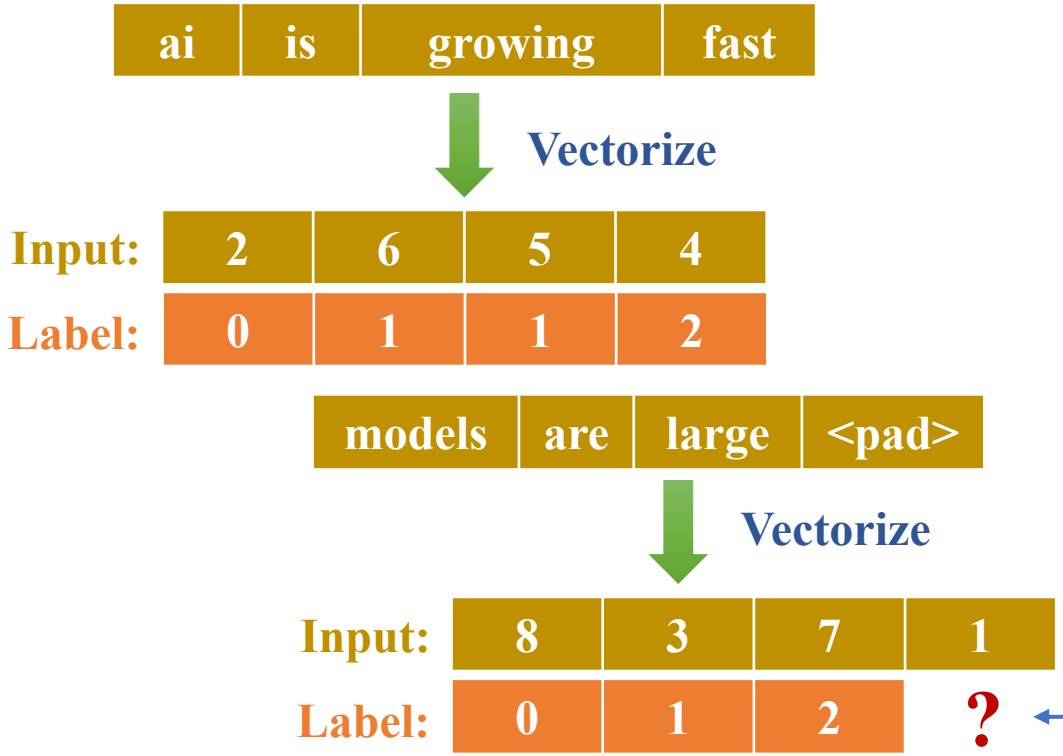
Vocabulary

```
sample1_ids = [vocab[token] for token in sample1_tokens]
sample2_ids = [vocab[token] for token in sample2_tokens]

print(sample1_ids)
print(sample2_ids)
```

```
[2, 6, 5, 4]
[8, 3, 7]
```

sequence_length = 4

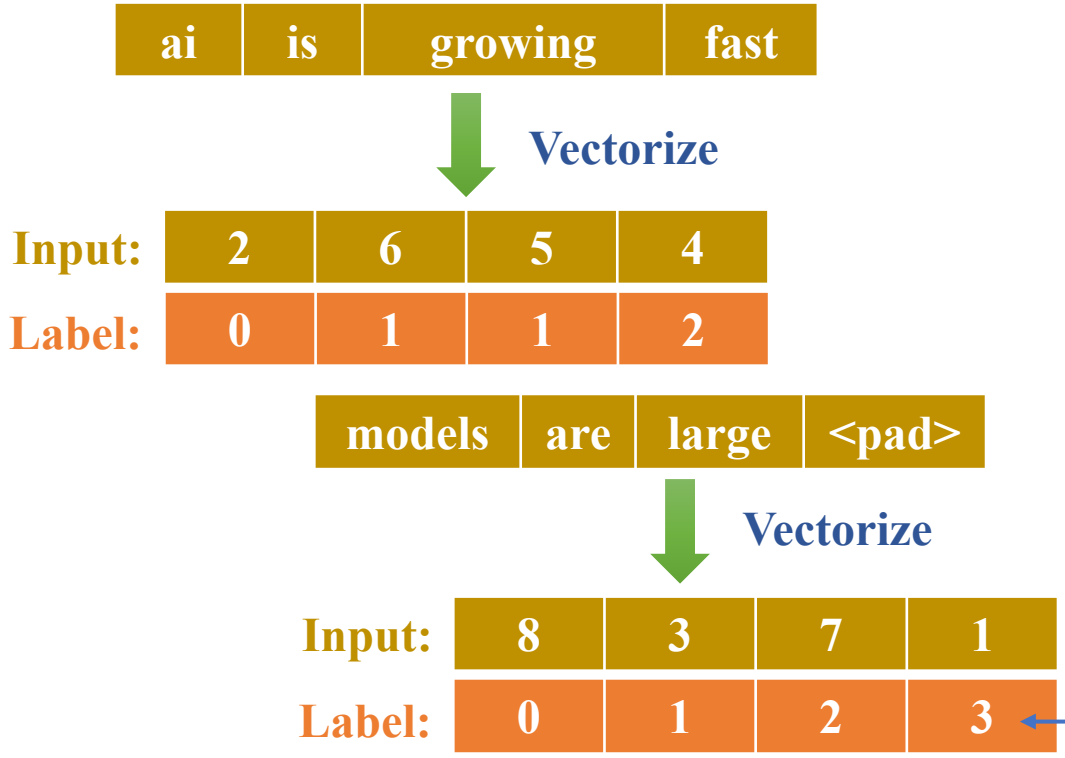


Index	Label
0	(Pro)noun
1	Verb
2	Others

Vectorization

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary



sequence_length = 4

Index	Label
0	(Pro)noun
1	Verb
2	Others
3	<pad>

```python
# Tokenize and numericalize your samples
def vectorize(text, vocab, sequence_length, sequence_label):
    tokens = tokenizer(text)

    token_ids = [vocab[token] for token in tokens]
    token_ids = token_ids + [vocab["<pad>"]] * (sequence_length - len(tokens))
    sequence_label = sequence_label + [3] * (sequence_length - len(tokens))

    return torch.tensor(token_ids, dtype=torch.long), torch.tensor(sequence_label, dtype=torch.long)

# Vectorize the samples
sentence_vecs = []
label_vecs = []
for sentence, labels in zip(corpus, labels):
    sentence_vec, labels_vec = vectorize(sentence, vocab, sequence_length, labels)
    sentence_vecs.append(sentence_vec)
    label_vecs.append(labels_vec)
```

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary

ai	is	growing	fast

sequence_length = 4

Vectorize

Input:	2	6	5	4
Label:	0	1	1	2

models	are	large	<pad>

Vectorize

Input:	8	3	7	1
Label:	0	1	2	3

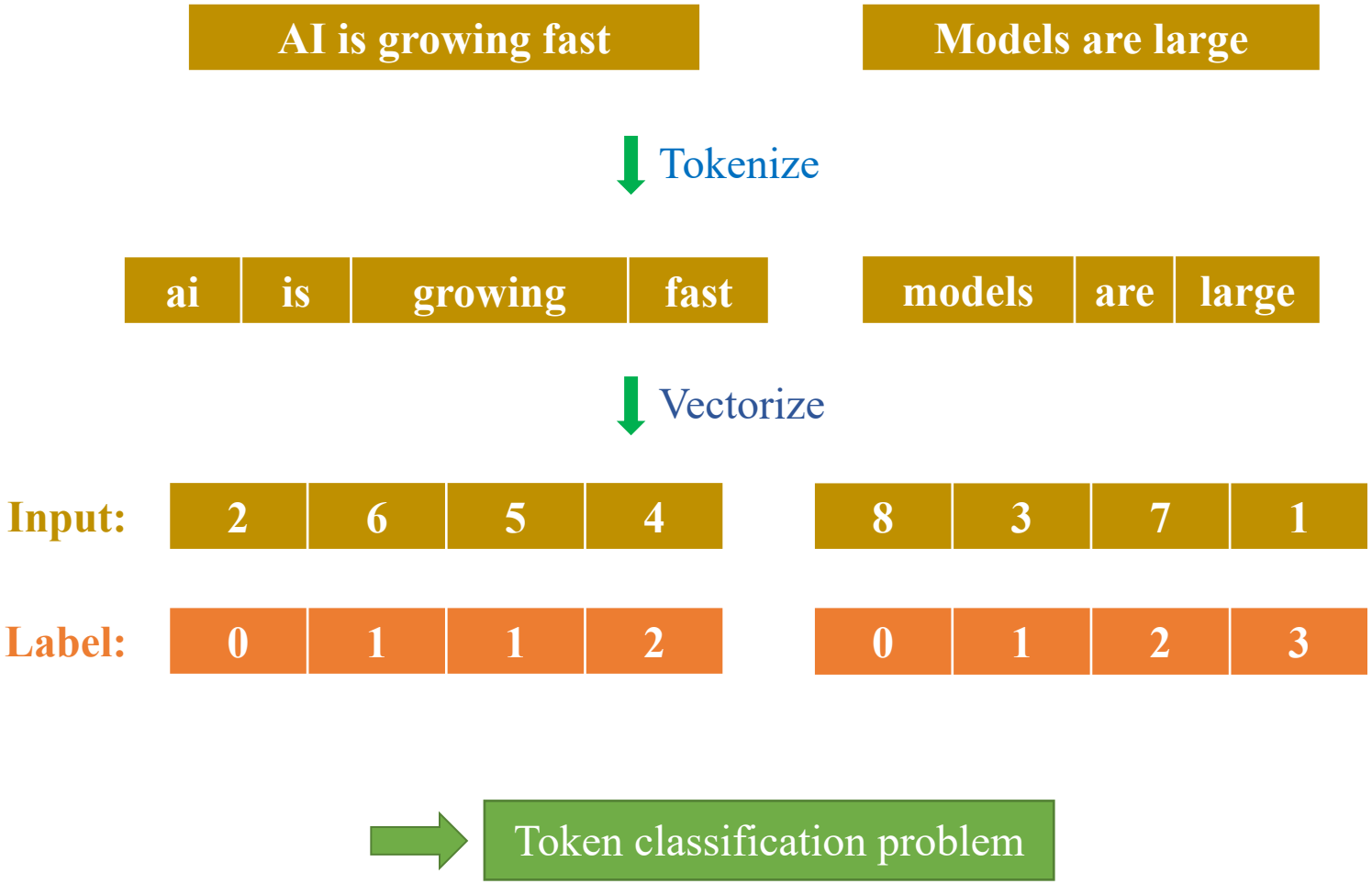
Index	Label
0	(Pro)noun
1	Verb
2	Others
3	<pad>

AI VIETNAM
All-in-One Course
(TA Session)

Vectorization

0	<unk>
1	<pad>
2	ai
3	are
4	fast
5	growing
6	is
7	large
8	models

Vocabulary

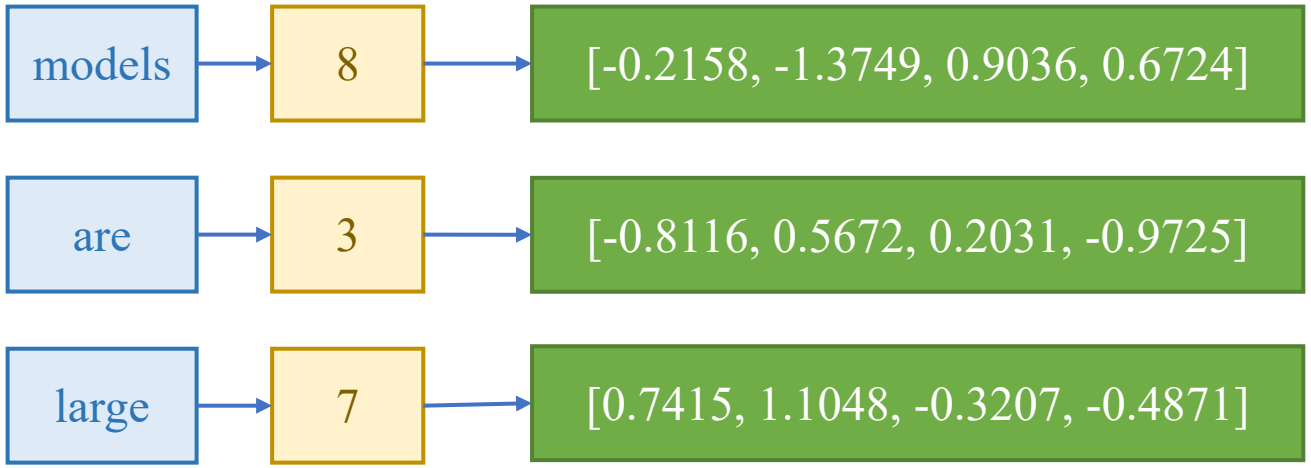


Part-of-Speech Tagging

❖ Embedding

index	vocab	Embedding vector
0	<unk>	[0.7321, -1.2503, 0.8154, -0.4987]
1	<pad>	[-0.2165, 1.7823, -0.6892, 0.9476]
2	ai	[1.1043, 0.3289, -1.5284, 0.6520]
3	are	[-0.8116, 0.5672, 0.2031, -0.9725]
4	fast	[0.4528, -0.1476, 1.2930, 0.7842]
5	growing	[-1.3459, 0.6791, -0.8853, 1.1207]
6	is	[0.9823, -0.5504, 0.4389, -1.2396]
7	large	[0.7415, 1.1048, -0.3207, -0.4871]
8	models	[-0.2158, -1.3749, 0.9036, 0.6724]

From Word2Vec models or **learnable** parameters



Part-of-Speech Tagging

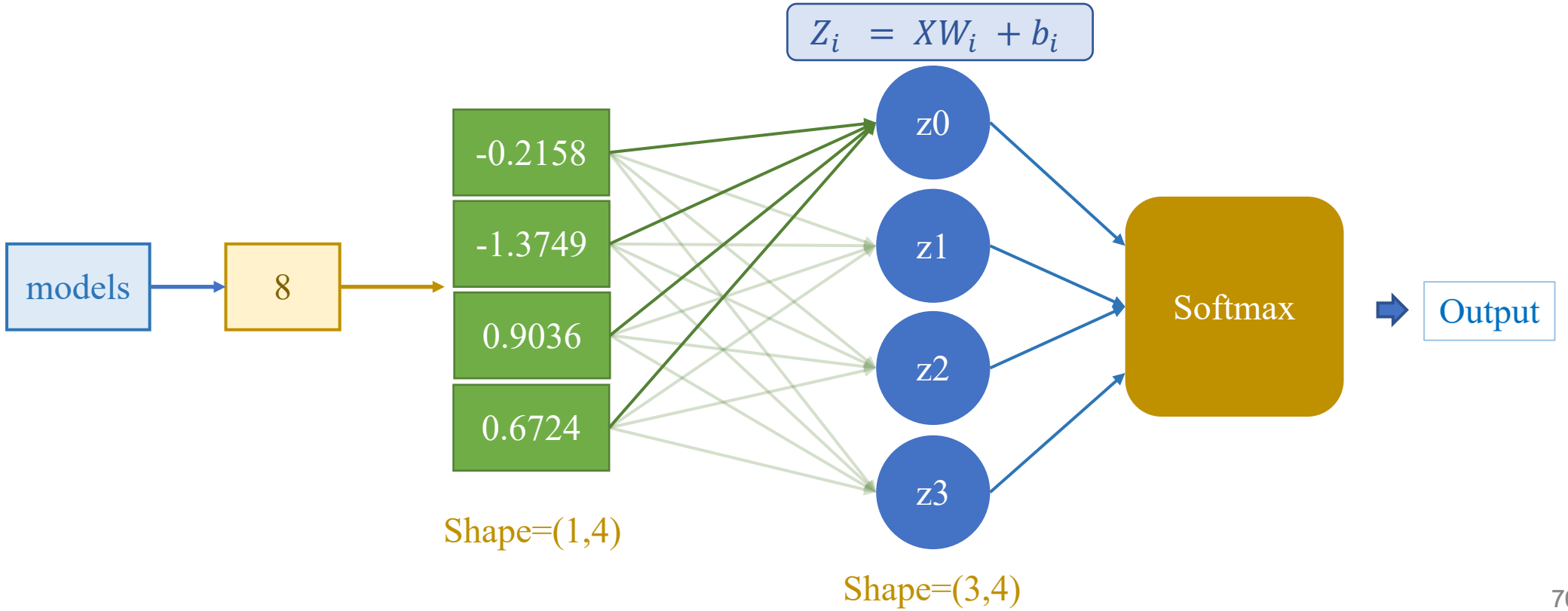
❖ **A word input**

W

[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]
b_0 b_1 b_2 b_3



Part-of-Speech Tagging

❖ **A word input**

$$Z_0 = XW_0 + b_0$$

W

[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]

b_0 b_1 b_2 b_3

X

-0.2158
-1.3749
0.9036
0.6724

x

W_0

[0.3748, -0.1062, 0.5347, 0.0573]

+

b_0

0.1987

−0.0809
0.146
0.4832
0.0385

Sum = 0.5868 + 0.1987 = Z_0

XW_0 b_0 = 0.7855

Part-of-Speech Tagging

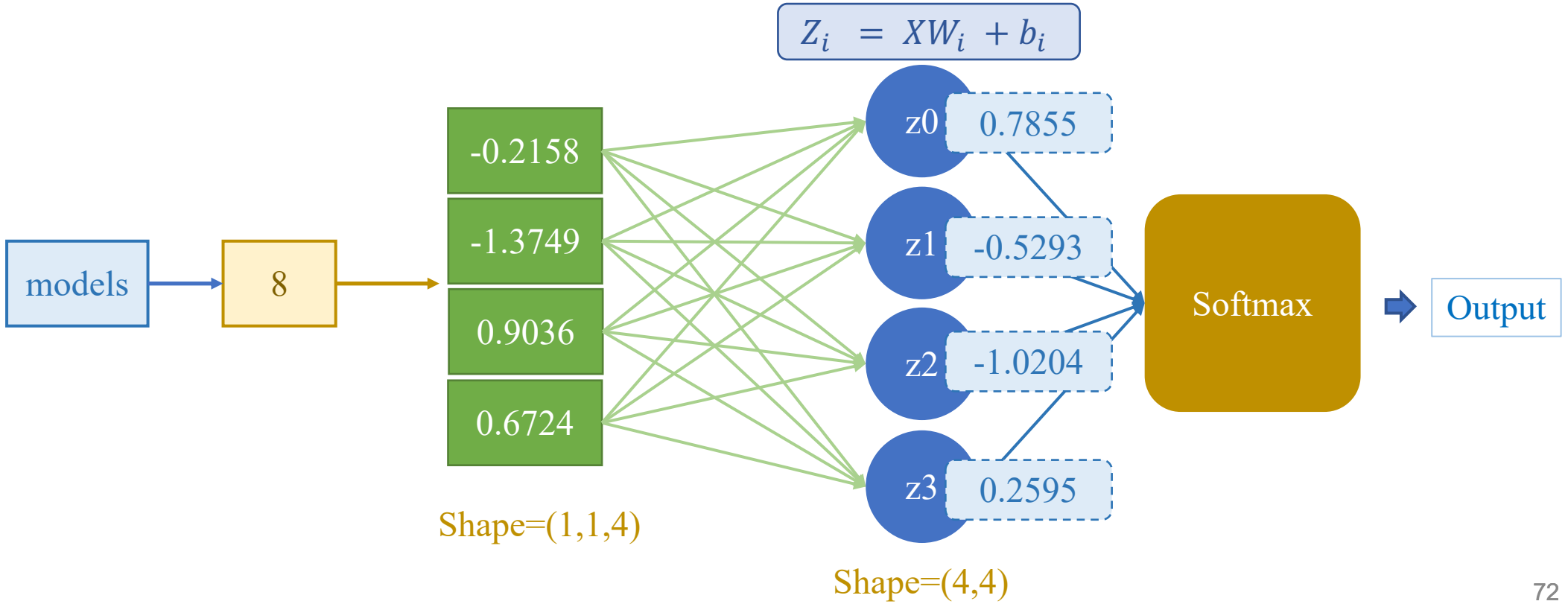
❖ **A word input**

W

[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

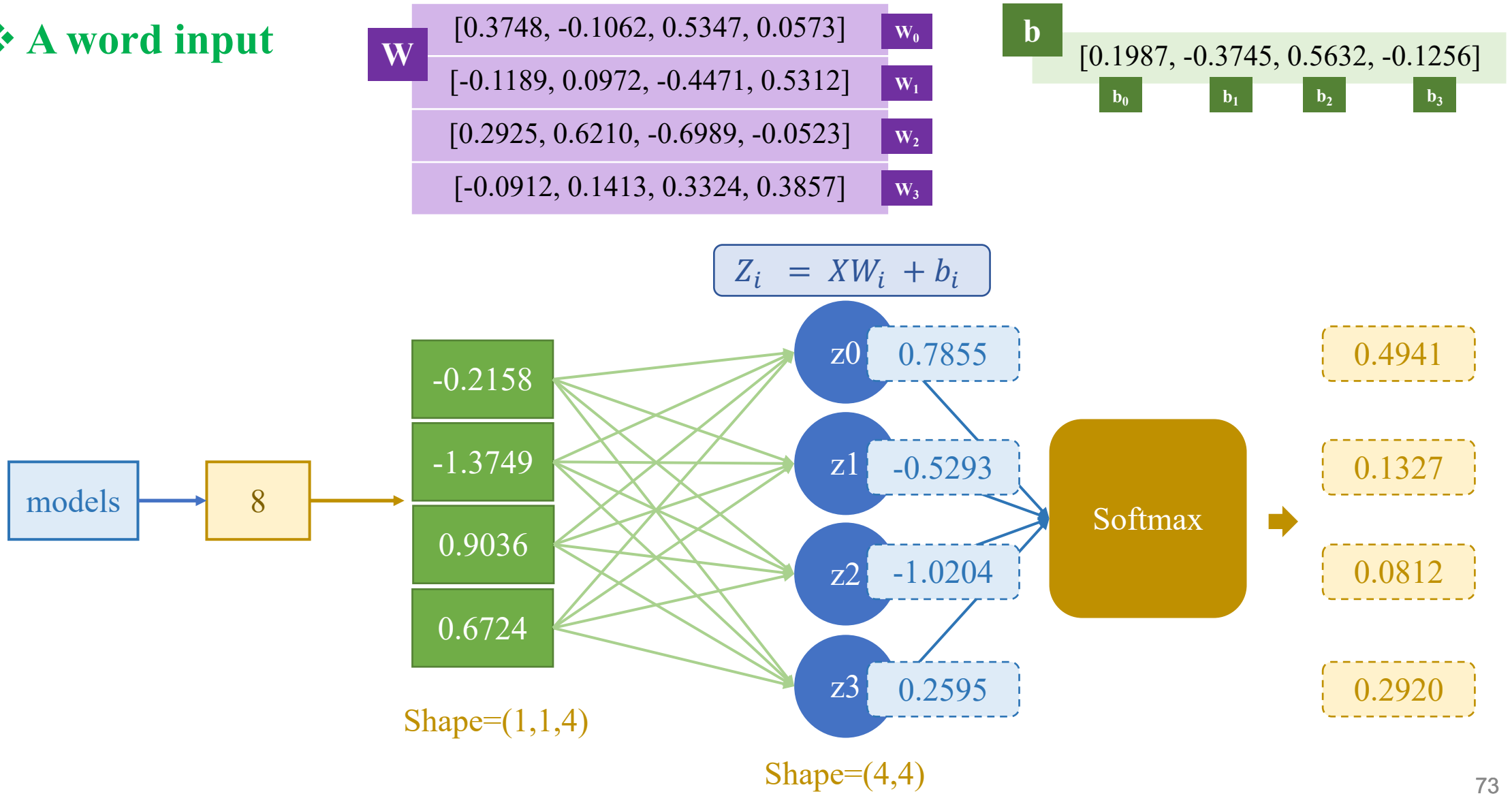
b

[0.1987, -0.3745, 0.5632, -0.1256]
b_0 b_1 b_2 b_3



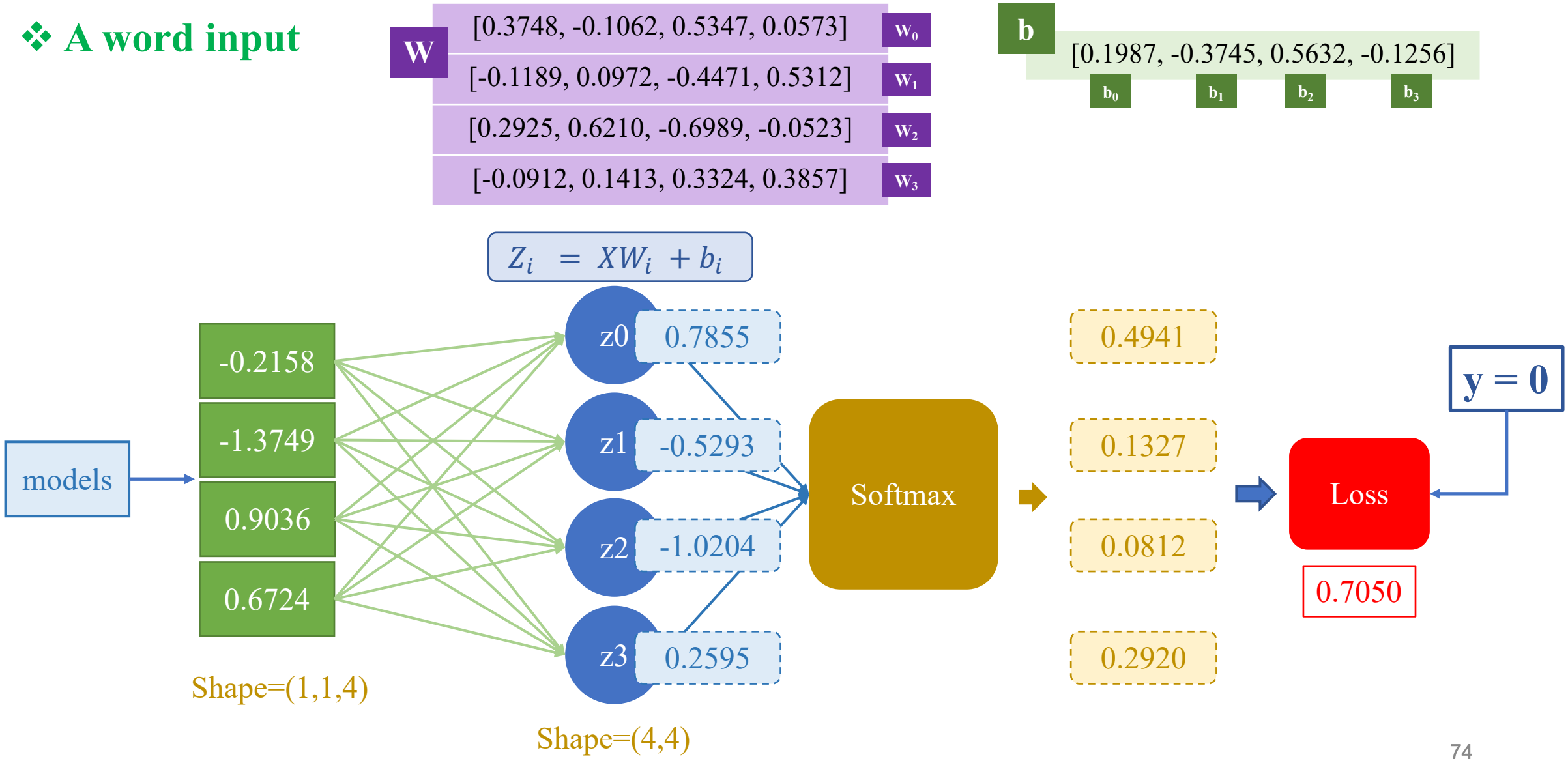
Part-of-Speech Tagging

❖ **A word input**



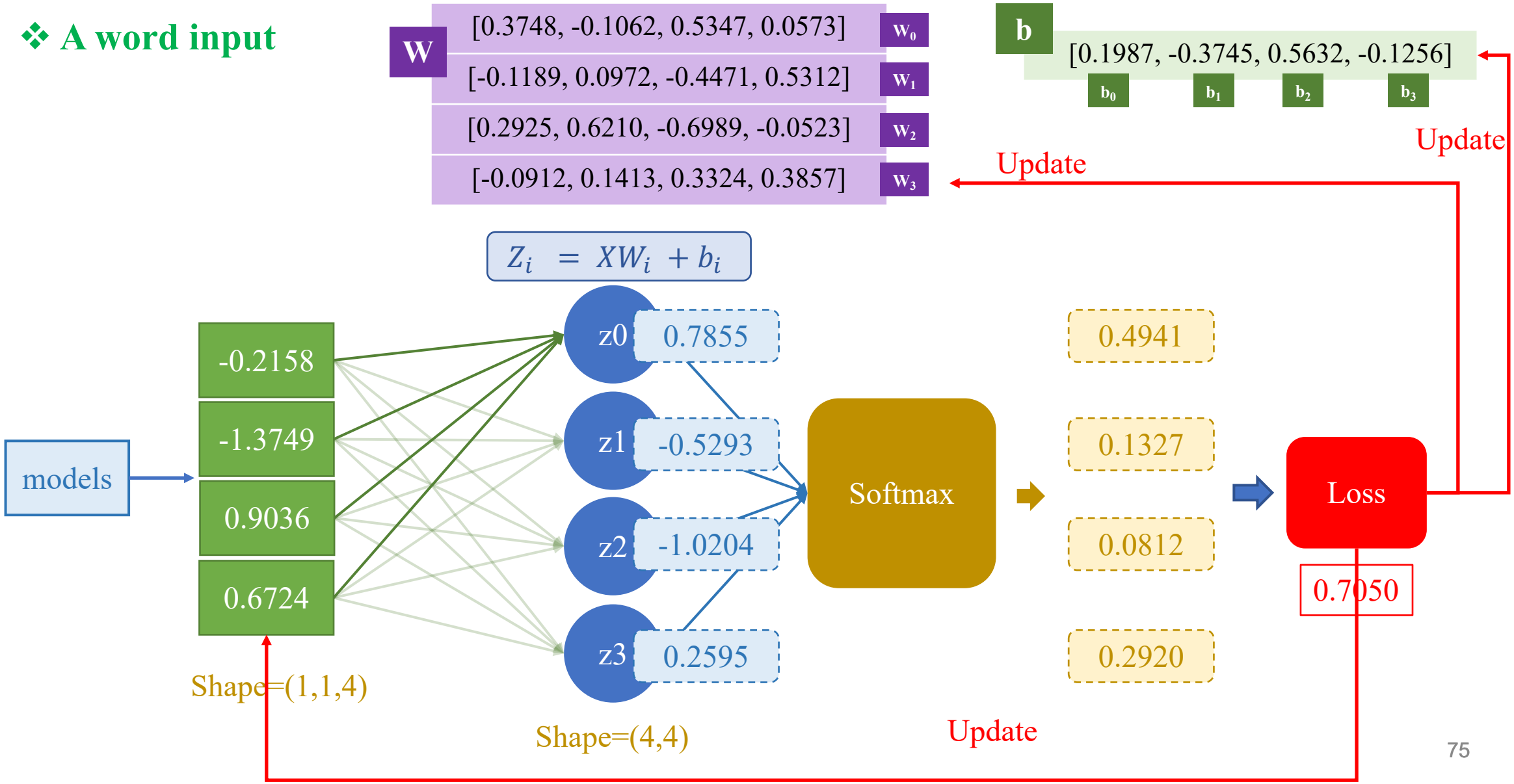
Part-of-Speech Tagging

❖ **A word input**



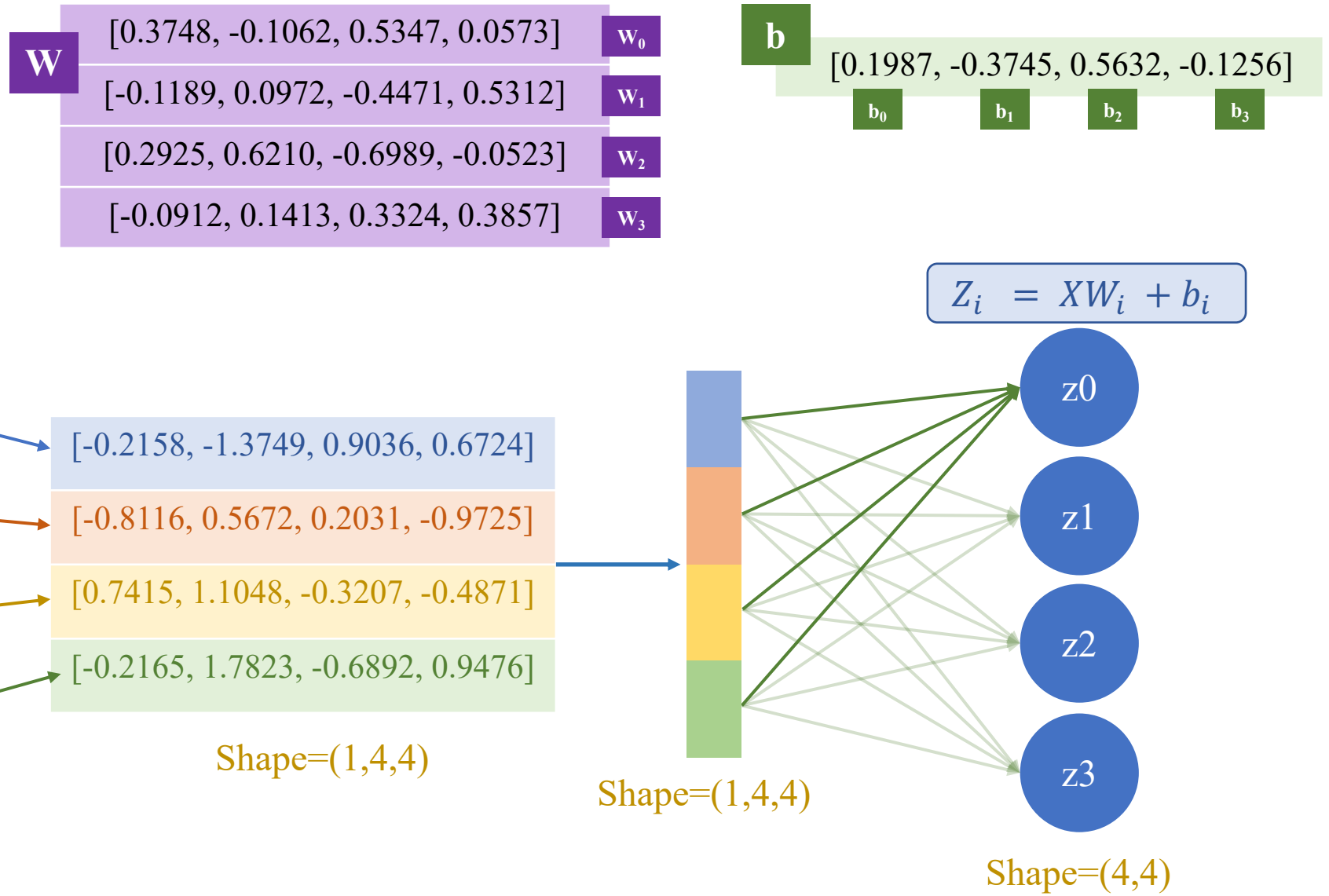
Part-of-Speech Tagging

❖ **A word input**



Part-of-Speech Tagging

❖ A word input



Part-of-Speech Tagging

❖ **A word input**

W

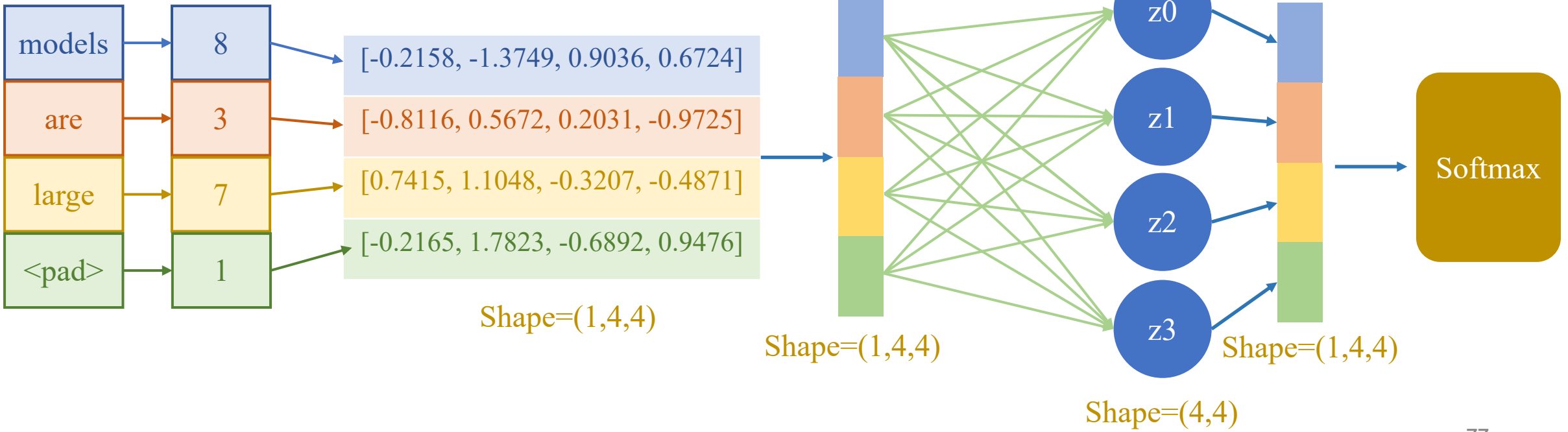
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]

b_0 b_1 b_2 b_3

$$Z_i = XW_i + b_i$$



Part-of-Speech Tagging

❖ **A word input**

W

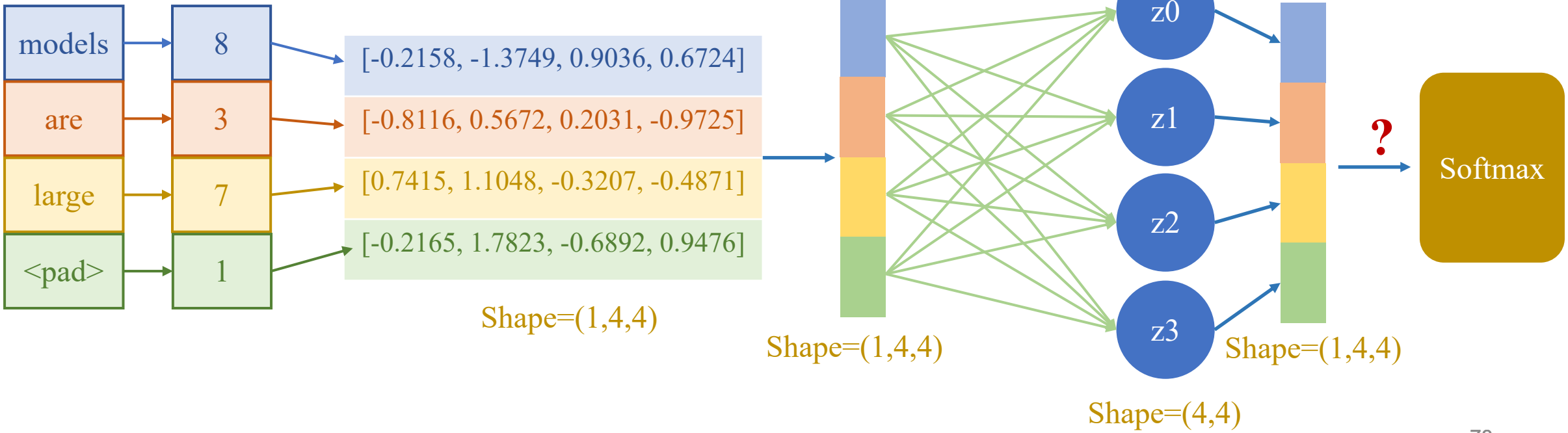
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]

b_0 b_1 b_2 b_3

$$Z_i = XW_i + b_i$$



Part-of-Speech Tagging

❖ A word input

nn.Linear

W

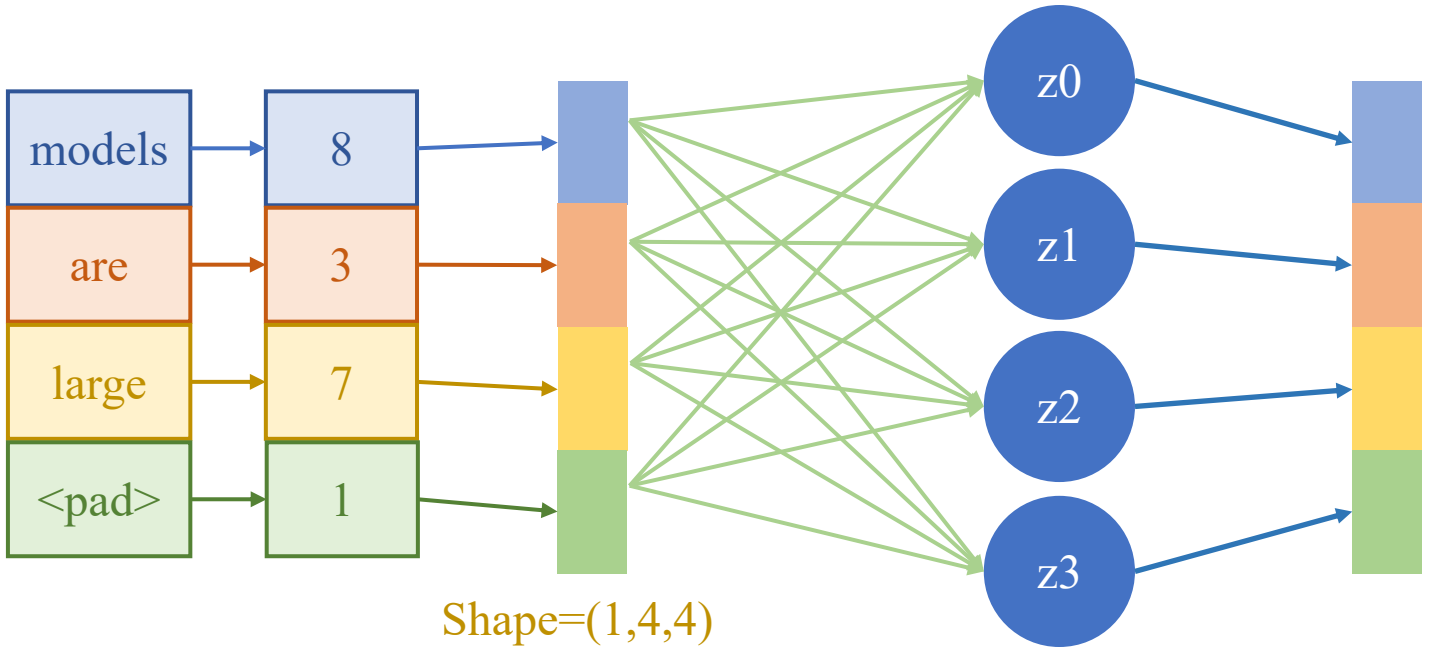
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]

b_0 b_1 b_2 b_3

$$Z_i = XW_i + b_i$$



Shape=(1,4,4)
(bs, seq_len, emb)

Shape=(4,4)

[0.7855, -0.1129, 0.1599, -0.3859]
[-0.5293, -0.8303, -0.4706, 0.6360]
[-1.0204, 0.5870, 1.7158, 2.0388]
[0.2595, -0.2790, -0.3316, 0.2824]

Shape=(1,4,4)
(bs, seq_len, classes)

Part-of-Speech Tagging

❖ **A word input**

W

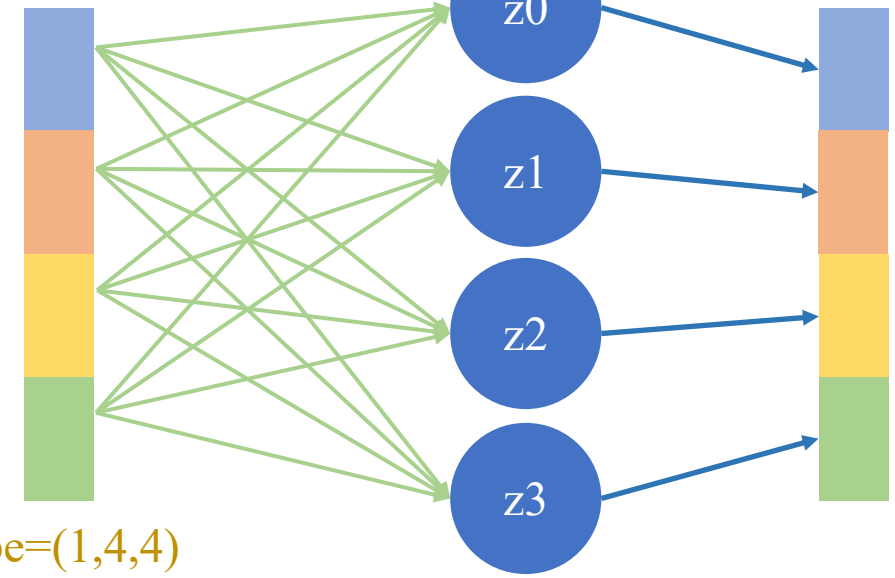
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]
b_0 b_1 b_2 b_3

Input shape: (N, C, d)
Target shape: (N, d)

$$Z_i = XW_i + b_i$$



Shape=(1,4,4)
(bs, seq_len, emb)

Shape=(4,4)

[0.7855, -0.1129, 0.1599, -0.3859]
[-0.5293, -0.8303, -0.4706, 0.6360]
[-1.0204, 0.5870, 1.7158, 2.0388]
[0.2595, -0.2790, -0.3316, 0.2824]

Shape=(1,4,4)
(bs, seq_len, classes)

Pytorch

nn.CrossEntropyLoss

permute

Shape=(1,4,4)
(bs, classes, seq_len)

Part-of-Speech Tagging

❖ **A word input**

W
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b
[0.1987, -0.3745, 0.5632, -0.1256]
b_0 b_1 b_2 b_3

Input shape: (N, C, d)
Target shape: (N, d)

$Z_i = XW_i +$

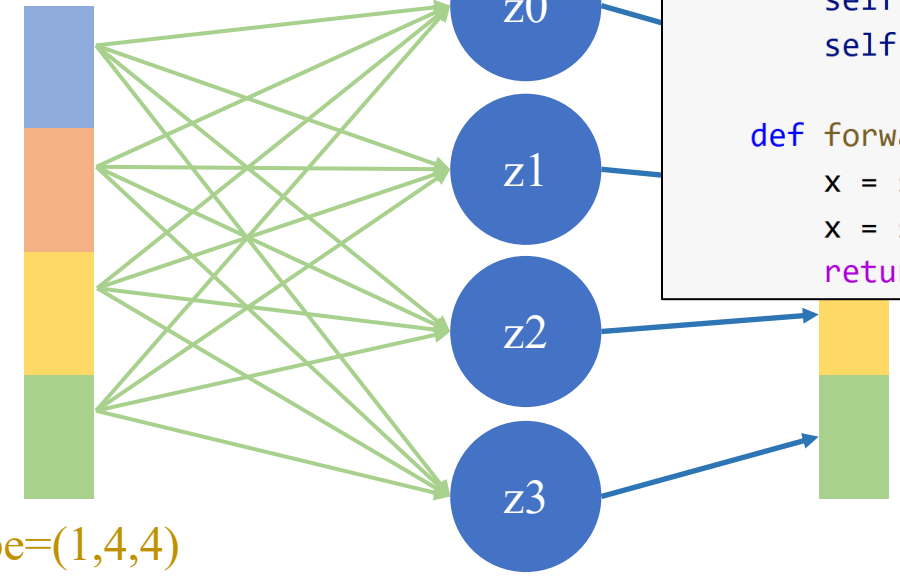
```
class POS_Model(nn.Module):
    def __init__(self, vocab_size, num_classes):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, 4)
        self.fc = nn.Linear(4, num_classes)

    def forward(self, x):
        x = self.embedding(x)
        x = self.fc(x)
        return x.permute(0, 2, 1)
```

Pytorch

nn.CrossEntropyLoss

permute



[0.2595, -0.2790, -0.3316, 0.2824]

Shape=(1,4,4)
(bs, seq_len, emb)

Shape=(4,4)

Shape=(1,4,4)
(bs, seq_len, classes)

Shape=(1,4,4)
(bs, classes, seq_len)

Part-of-Speech Tagging

❖ **A word input**

W

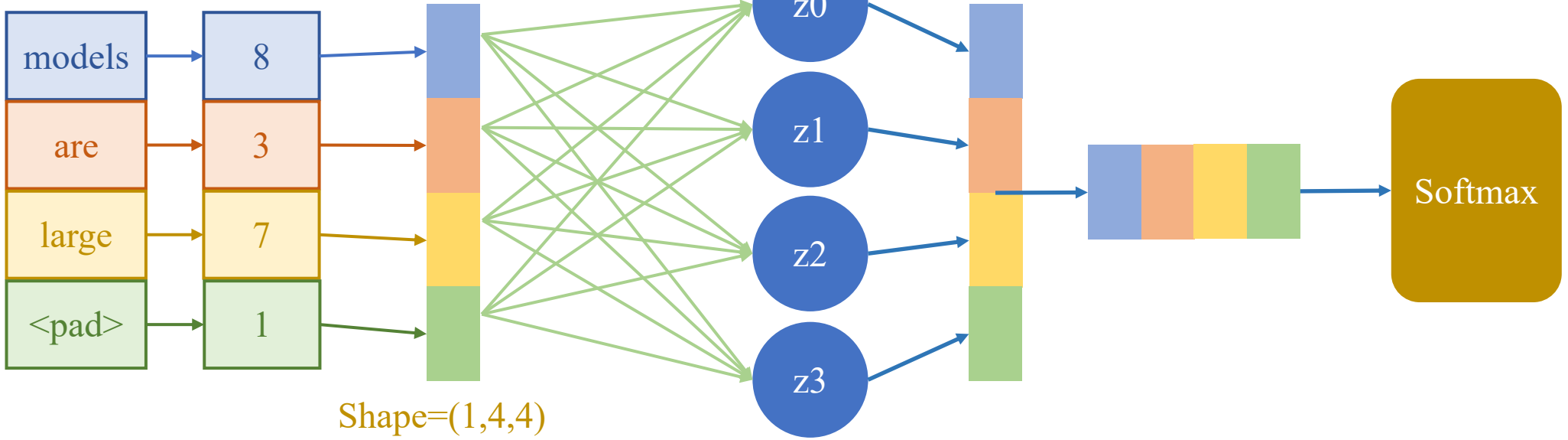
[0.3748, -0.1062, 0.5347, 0.0573] w_0
[-0.1189, 0.0972, -0.4471, 0.5312] w_1
[0.2925, 0.6210, -0.6989, -0.0523] w_2
[-0.0912, 0.1413, 0.3324, 0.3857] w_3

b

[0.1987, -0.3745, 0.5632, -0.1256]

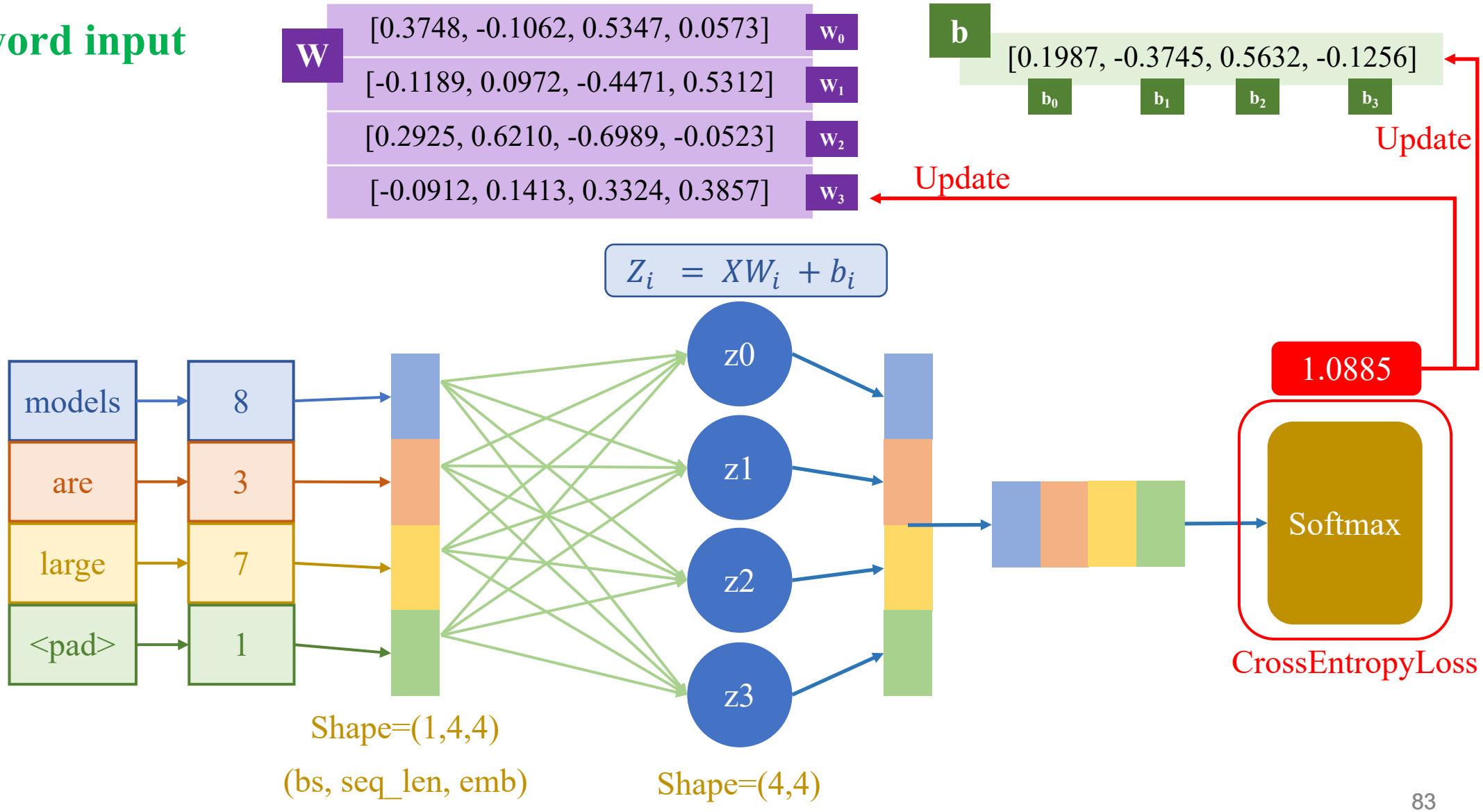
b_0 b_1 b_2 b_3

$$Z_i = XW_i + b_i$$

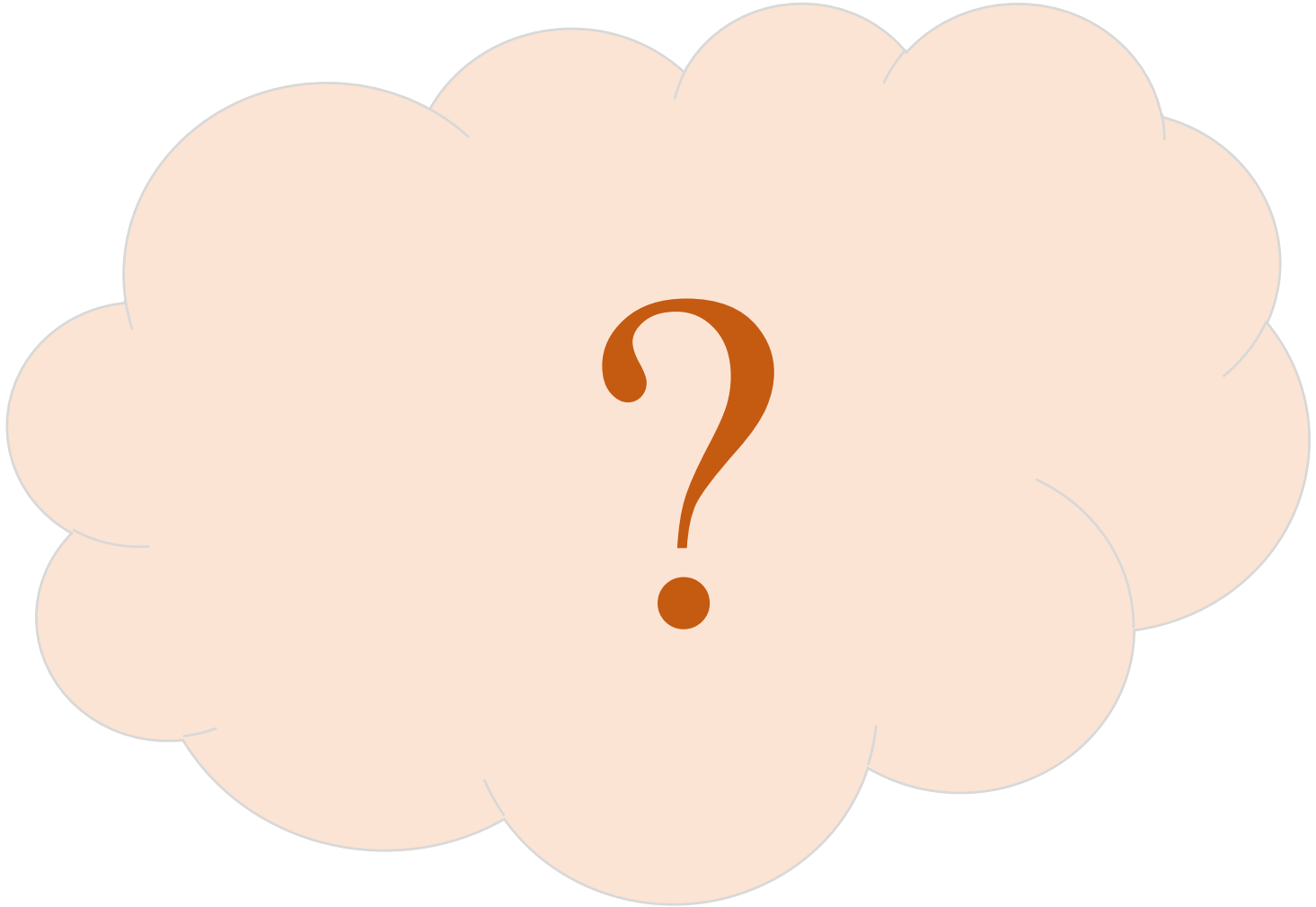


Part-of-Speech Tagging

❖ **A word input**



Question




Thank you!