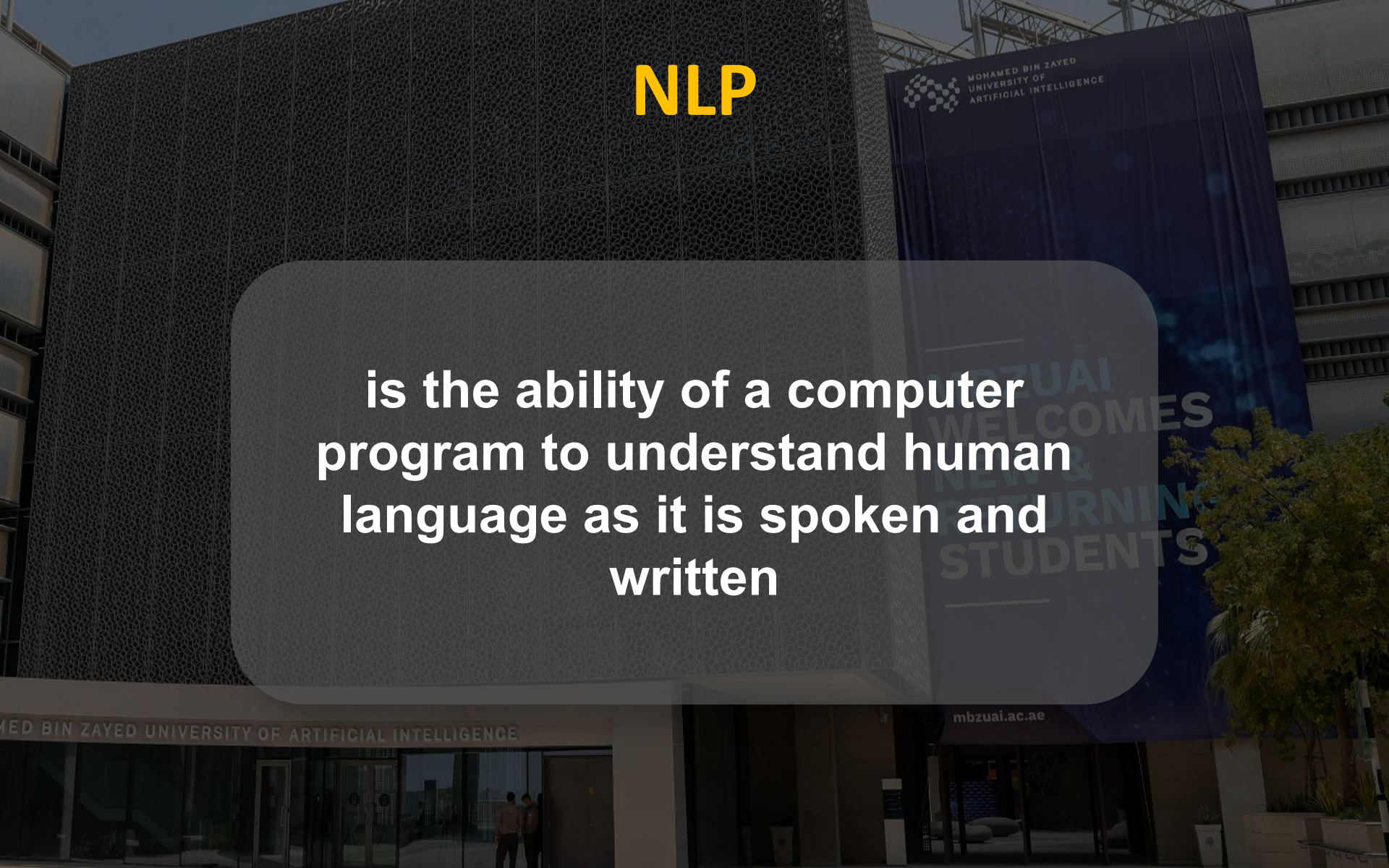
Natural Language Processing

(NLP)

With TensorFlow

A.L. Osama Al-atraqchi

NLP



is the ability of a computer program to understand human language as it is spoken and written

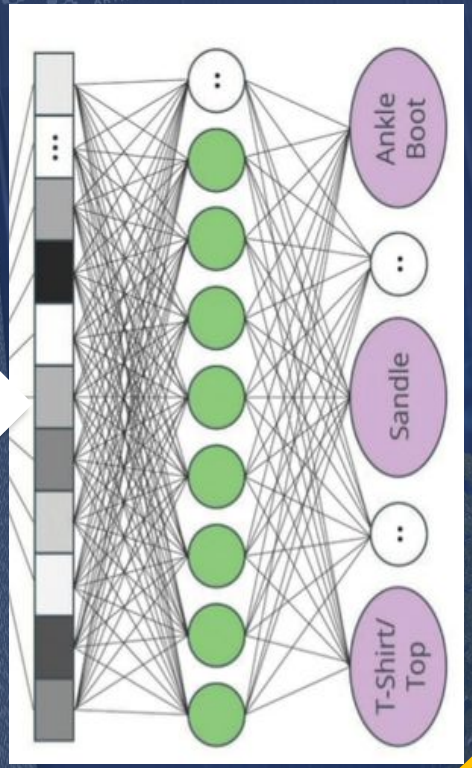
Tokenization

Sequencing

Unified the Length

Word Embedding

	0	1	2	3	4	5	6	7	8
0	0	0	10	25	40	25	10	0	0
1	0	10	25	40	55	40	25	10	0
2	10	25	40	55	70	55	40	25	10
3	25	40	55	70	85	70	55	40	25
4	40	55	70	85	100	85	70	55	40
5	25	40	55	70	85	70	55	40	25
6	10	25	40	55	70	55	40	25	10
7	0	10	25	40	55	40	25	10	0
8	0	0	10	25	40	25	10	0	0

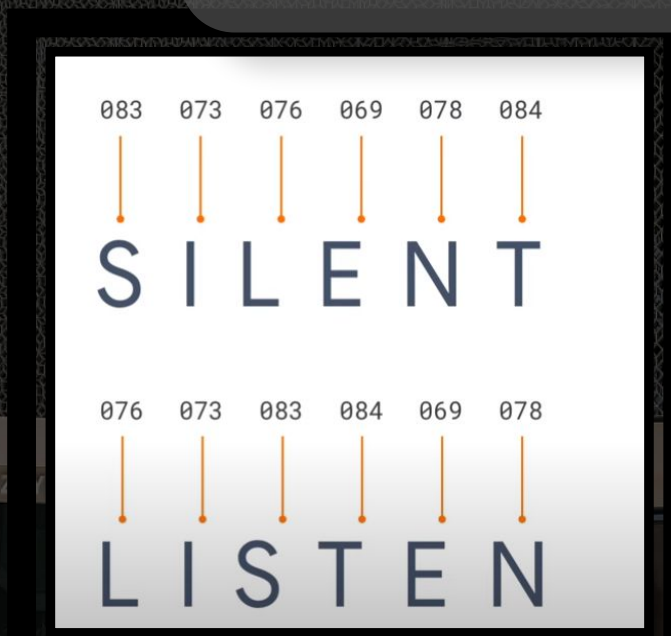


Pre-processing

Training

Tokenizatio n

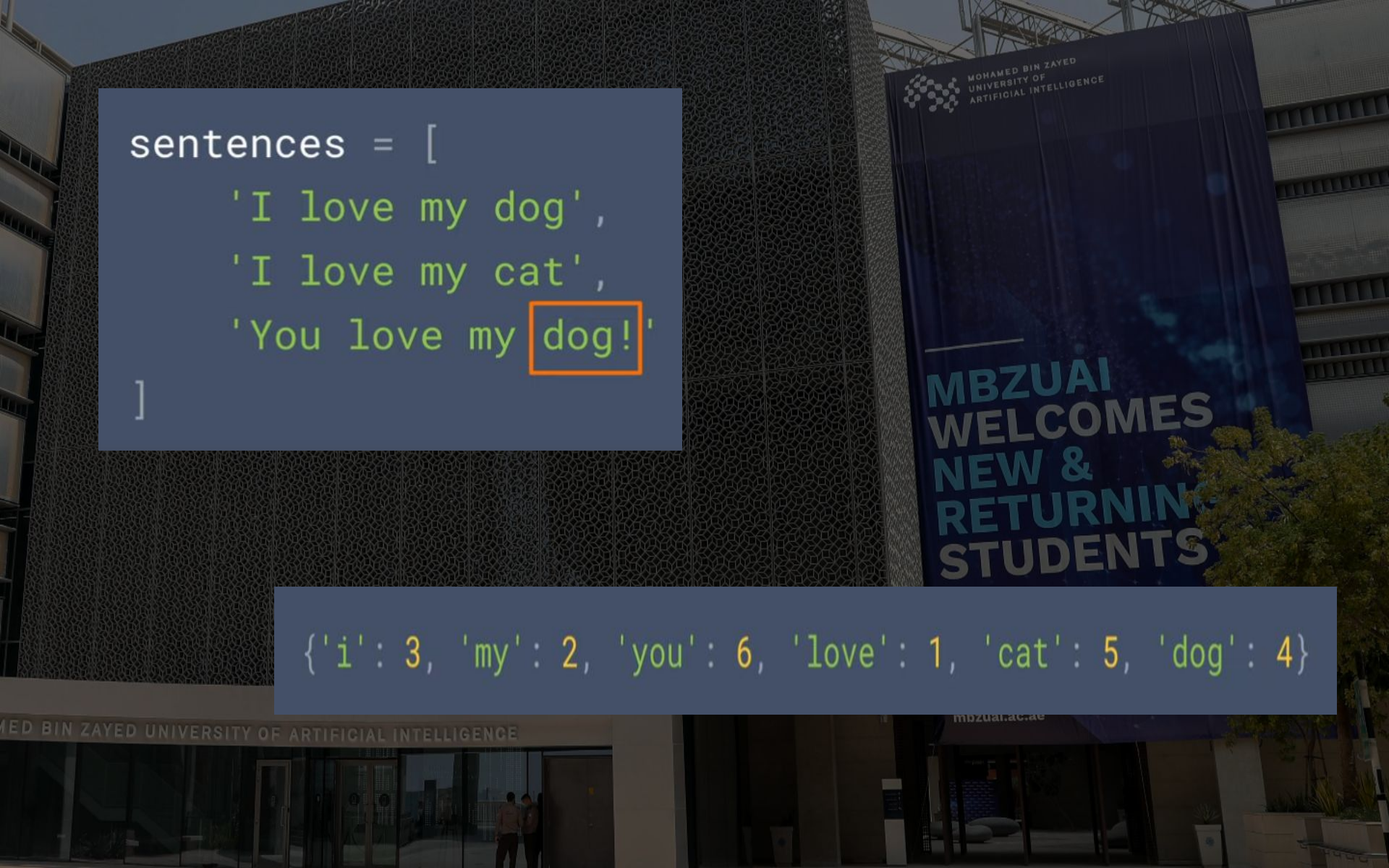
Is a process of breaking up a string into a tokens (words, numbers), in TensorFlow should be integer



TensorFlo

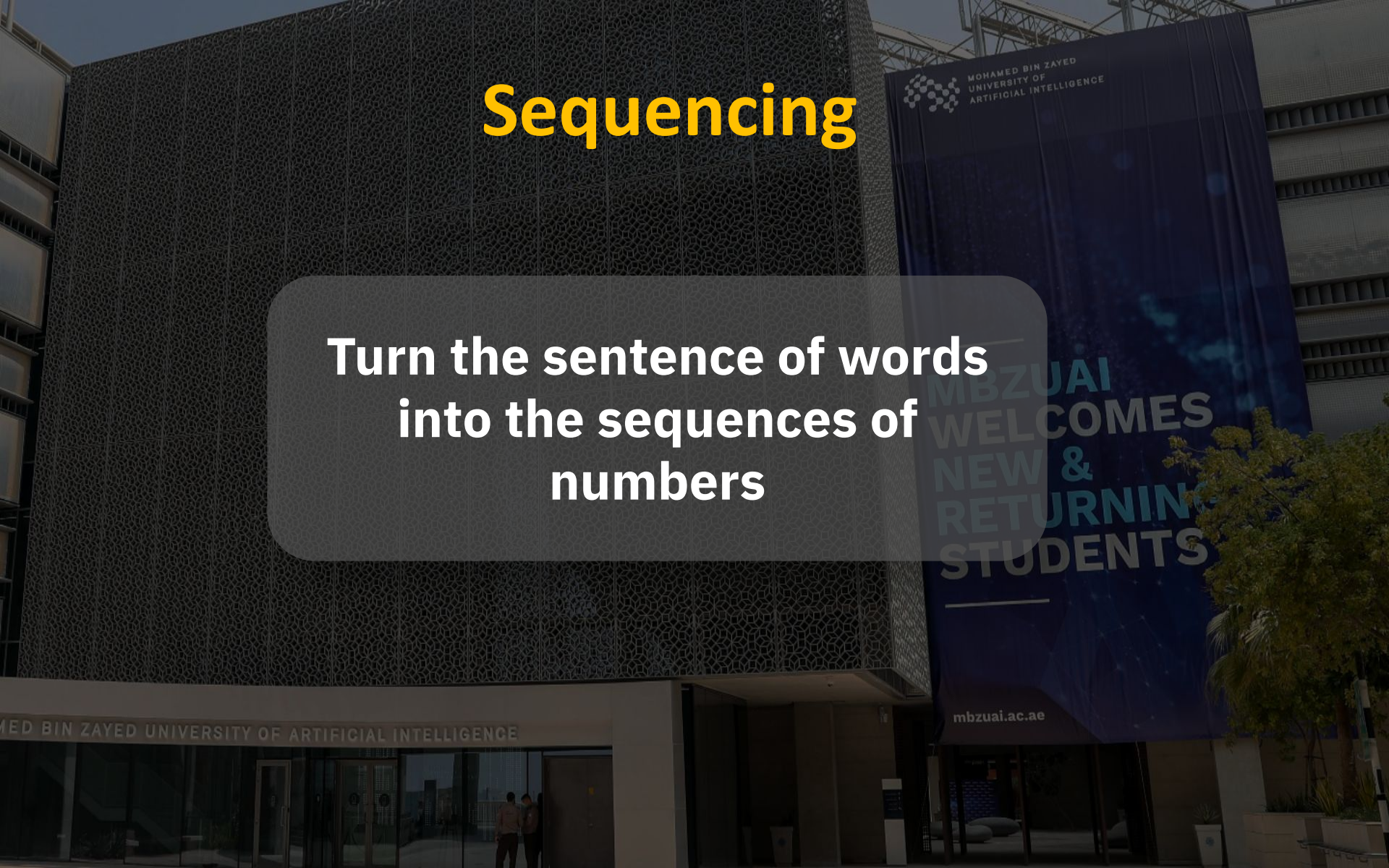
```
sentences = [
    'I love my dog',
    'I love my cat'
]

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(word_index)
```

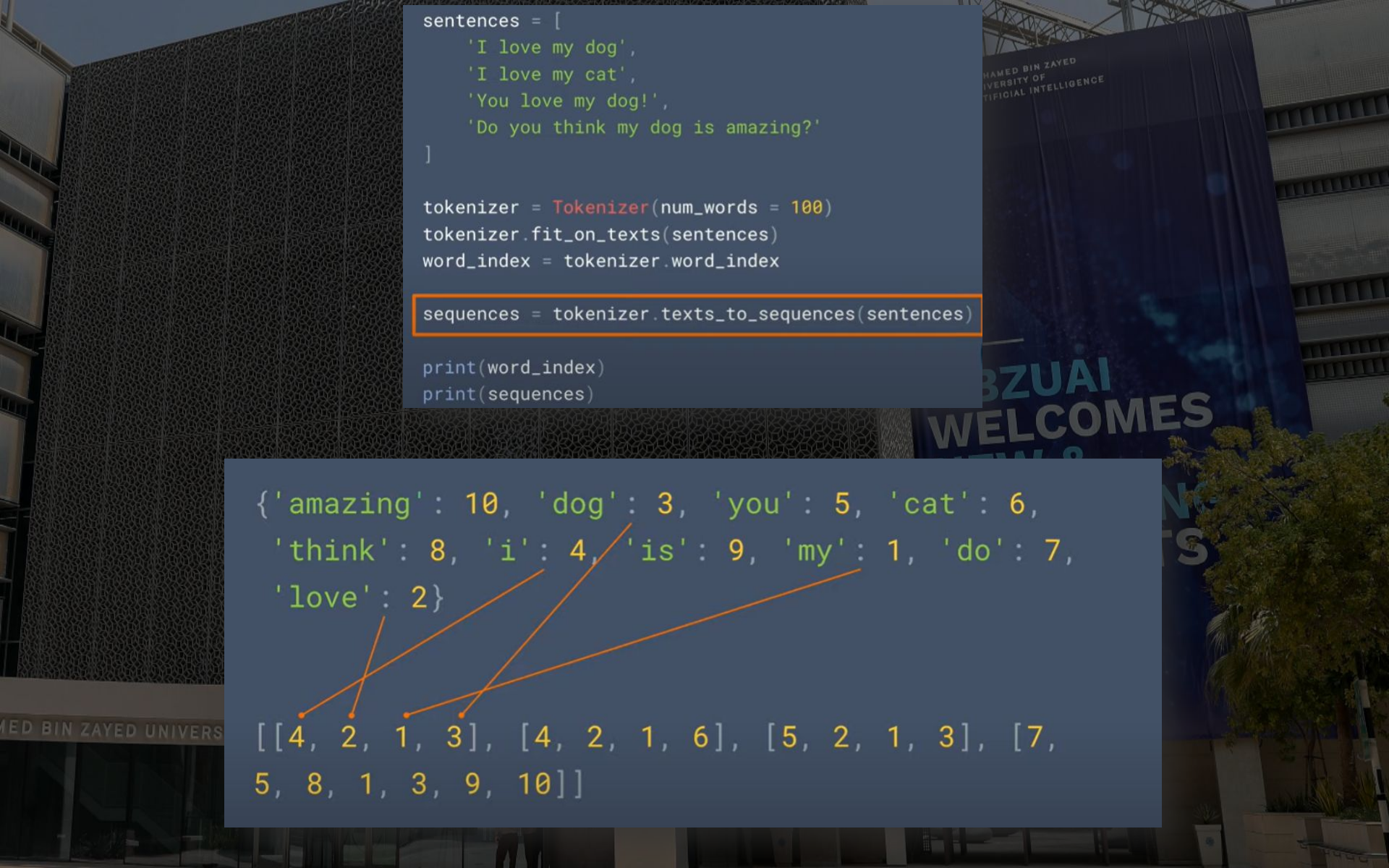
```
{'i': 1, 'my': 3, 'dog': 4, 'cat': 5, 'love': 2}
```


sentences = [
'I love my dog',
'I love my cat',
'You love my dog!'
]
MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE
MBZUAI WELCOMES NEW & RETURNIN STUDENTS
{'i': 3, 'my': 2, 'you': 6, 'love': 1, 'cat': 5, 'dog': 4}
MED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Sequencing

Turn the sentence of words into the sequences of numbers



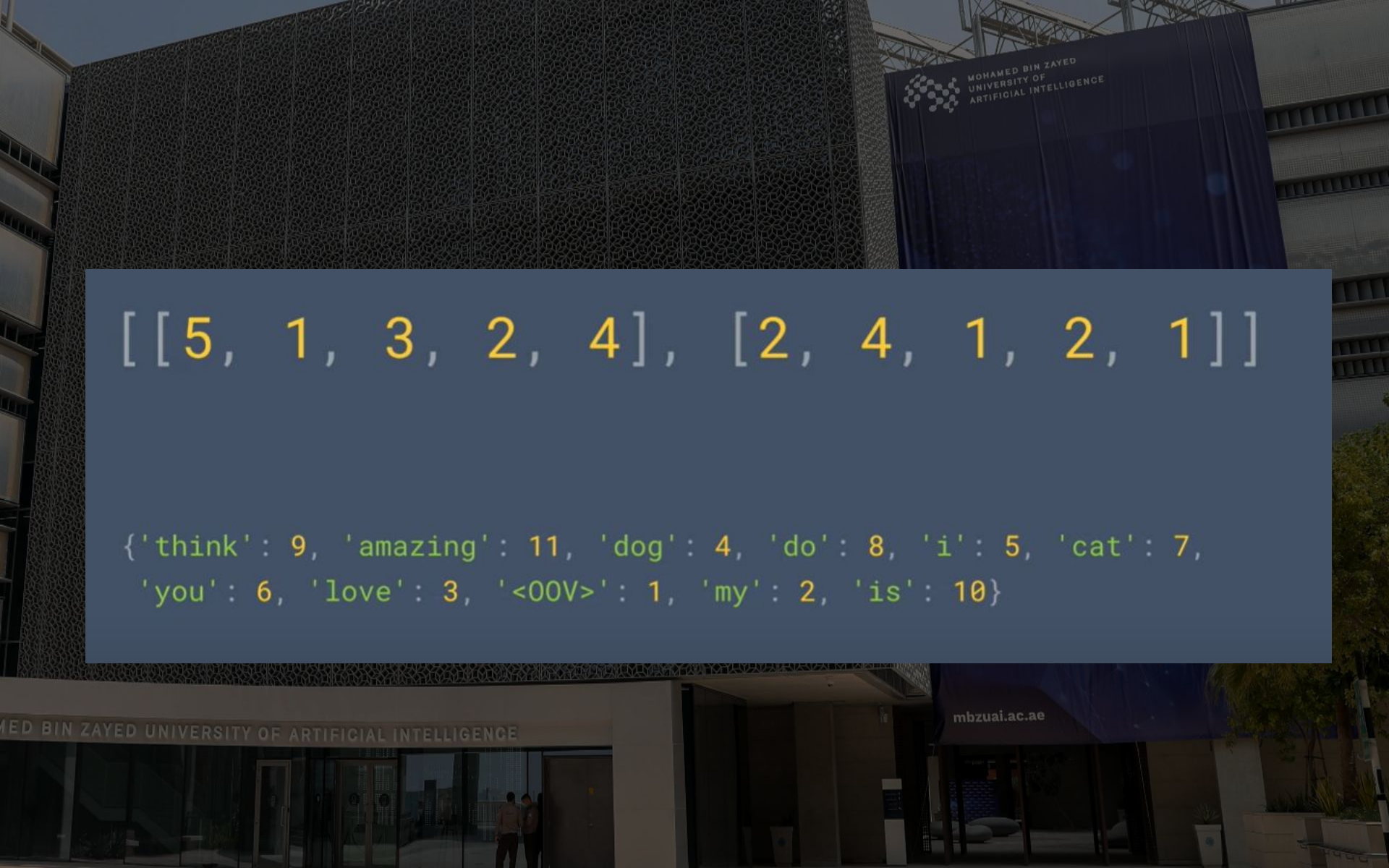

sentences = [
'I love my dog',
'I love my cat',
'You love my dog!',
'Do you think my dog is amazing?'
]
tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(sentences)
print(word_index)
print(sequences)
{'amazing': 10, 'dog': 3, 'you': 5, 'cat': 6,
'think': 8, 'i': 4, 'is': 9, 'my': 1, 'do': 7,
'love': 2}
[[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7,
5, 8, 1, 3, 9, 10]]

Out of Vocabulary

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

```
[[4, 2, 1, 3], [1, 3, 1]]

{'think': 8, 'amazing': 10, 'my': 1, 'love': 2, 'dog': 3, 'is': 9,
'you': 5, 'do': 7, 'cat': 6, 'i': 4}
```

```
tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
```




```
[[5, 1, 3, 2, 4], [2, 4, 1, 2, 1]]
```

```
{'think': 9, 'amazing': 11, 'dog': 4, 'do': 8, 'i': 5, 'cat': 7,
 'you': 6, 'love': 3, '<OOV>': 1, 'my': 2, 'is': 10}
```

Padding

Input image (28x28 = 784 pixels)

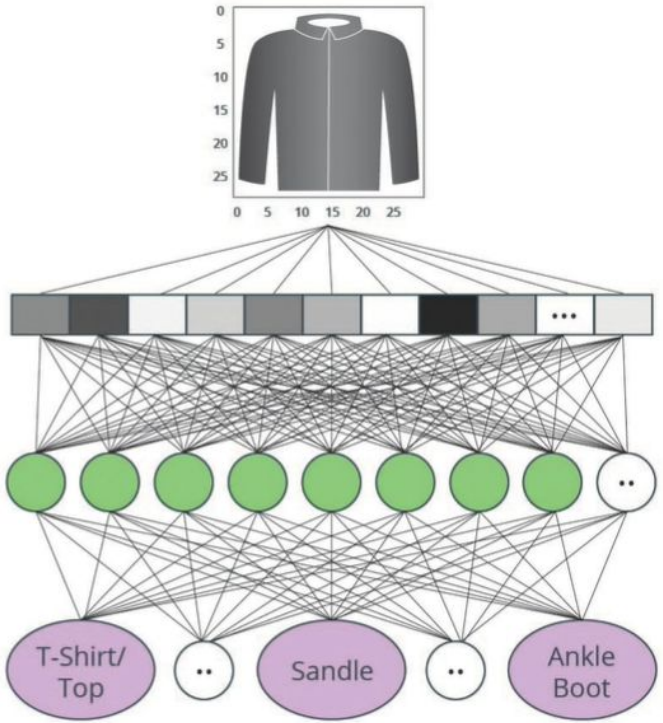
```
tf.keras.layers.Flatten (input_shape=(28, 28, 1))
```

Dense layer (128 units)

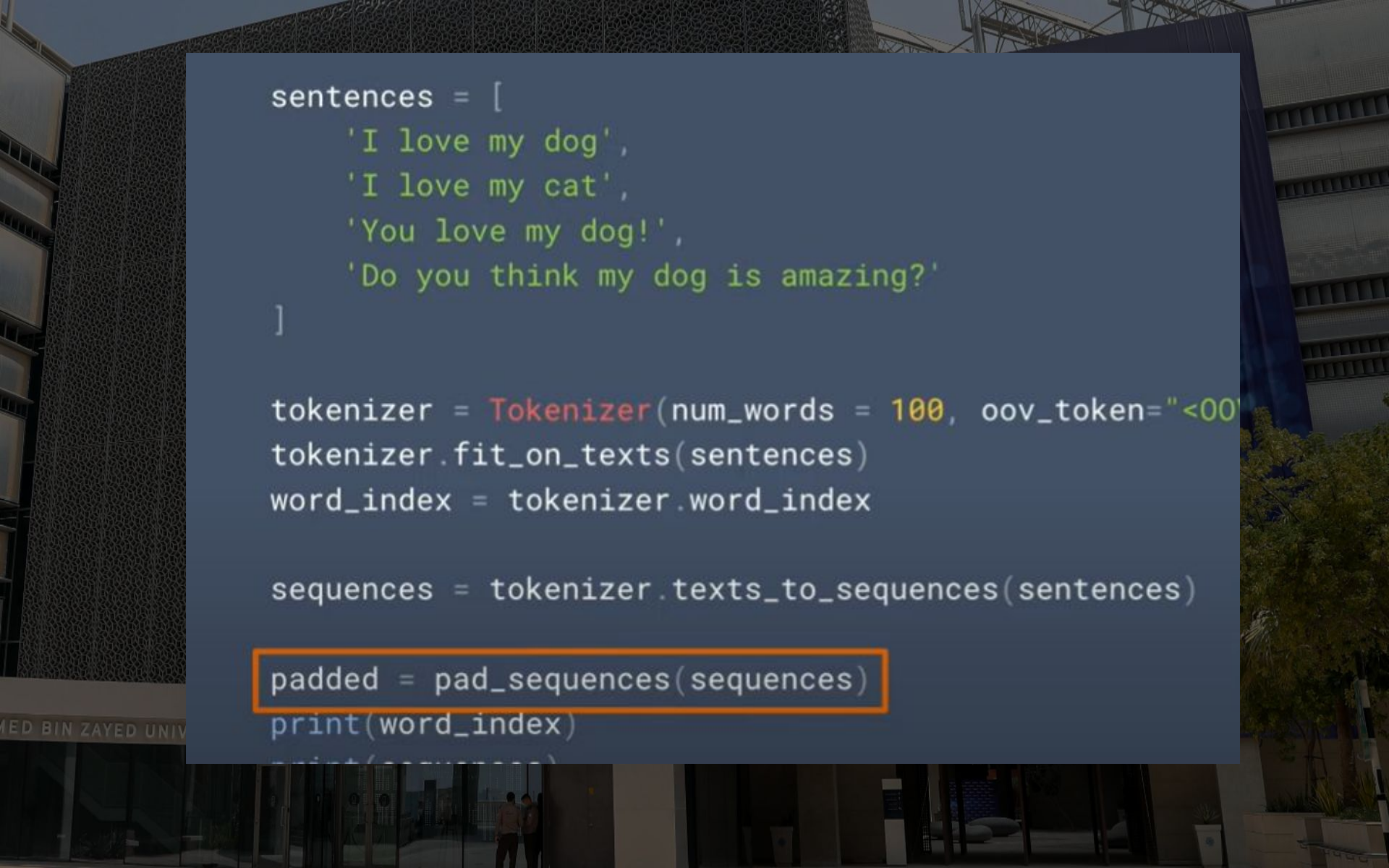
```
tf.keras.layers.Dense(128, activation=tf.nn.relu)
```

Output (10 units)

```
tf.keras.layers.Dense(10, activation=tf.nn.softmax)
```



Probability of each class
Sum of all values == 1 (100%)


sentences = [
'I love my dog',
'I love my cat',
'You love my dog!',
'Do you think my dog is amazing?'
]
tokenizer = Tokenizer(num_words = 100, oov_token="<OO
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences)
print(word_index)

```
padded = pad_sequences(sequences, padding='post', maxlen=5)
```

```
{'do': 8, 'you': 6, 'love': 3, 'i': 5, 'amazing': 11, 'my': 2,
'is': 10, 'think': 9, 'dog': 4, '<OOV>': 1, 'cat': 7}

[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]


[[ 0  0  0  5  3  2  4]
 [ 0  0  0  5  3  2  7]
 [ 0  0  0  6  3  2  4]
 [ 8  6  9  2  4 10 11]]
```

Embedding

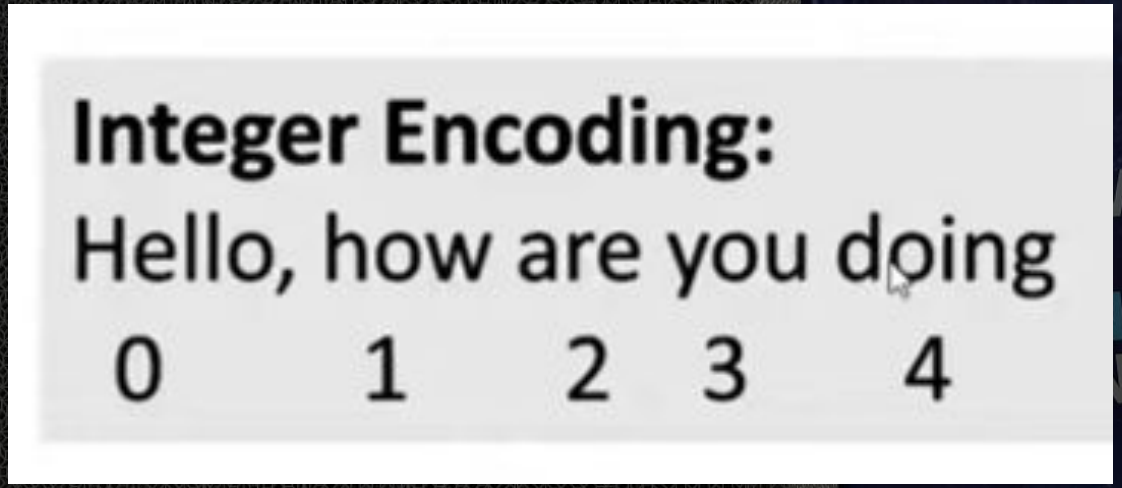
- Word2Vec.
- GloVe.
- Deep Learning

Convert the words to the vectors (Array of Numbers), and the goal is to make the similar words have a similar encoding after training.

One – hot encoded

	Hello	How	Are	You	Doing
Hello	1	0	0	0	0
How	0	1	0	0	0
Are	0	0	1	0	0
You	0	0	0	1	0
doing	0	0	0	0	1

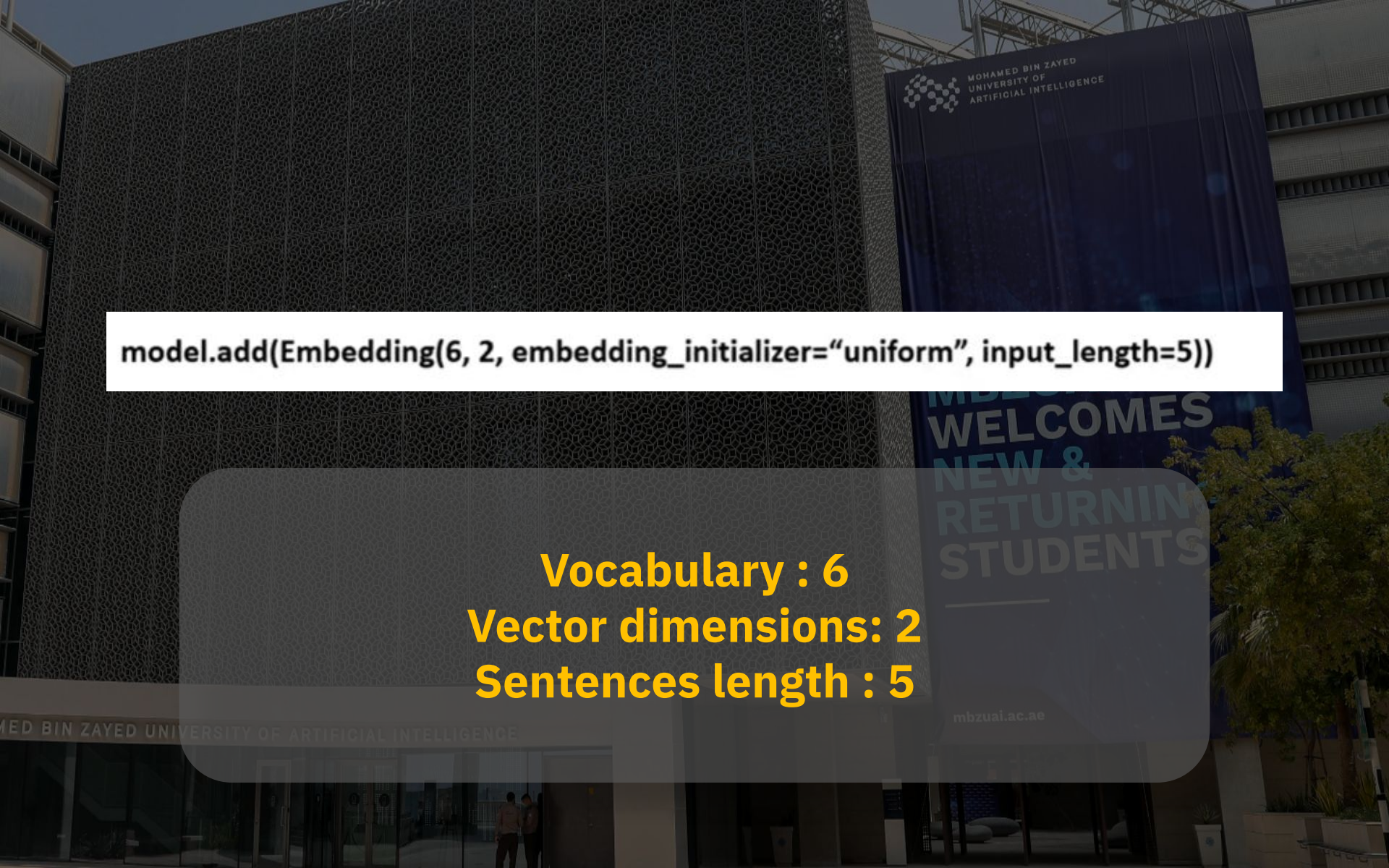
Integer Encoding



Embedding

Word	Integer	Embedding				
Hello	0	0.0286222	0.024707	0.0179271	0.00447939	0.049296
How	1	-0.00931804	-0.0488322	0.032062	-0.0299356	0.0370122
Are	2	-0.0335814	-0.00357039	0.0499876	0.0277698	0.000798009
You	3	-0.041718	0.00189773	0.039211	-0.0209171	0.0254348
Doing	4	-0.0363536	0.00302433	-0.0344366	0.0360162	0.0187734

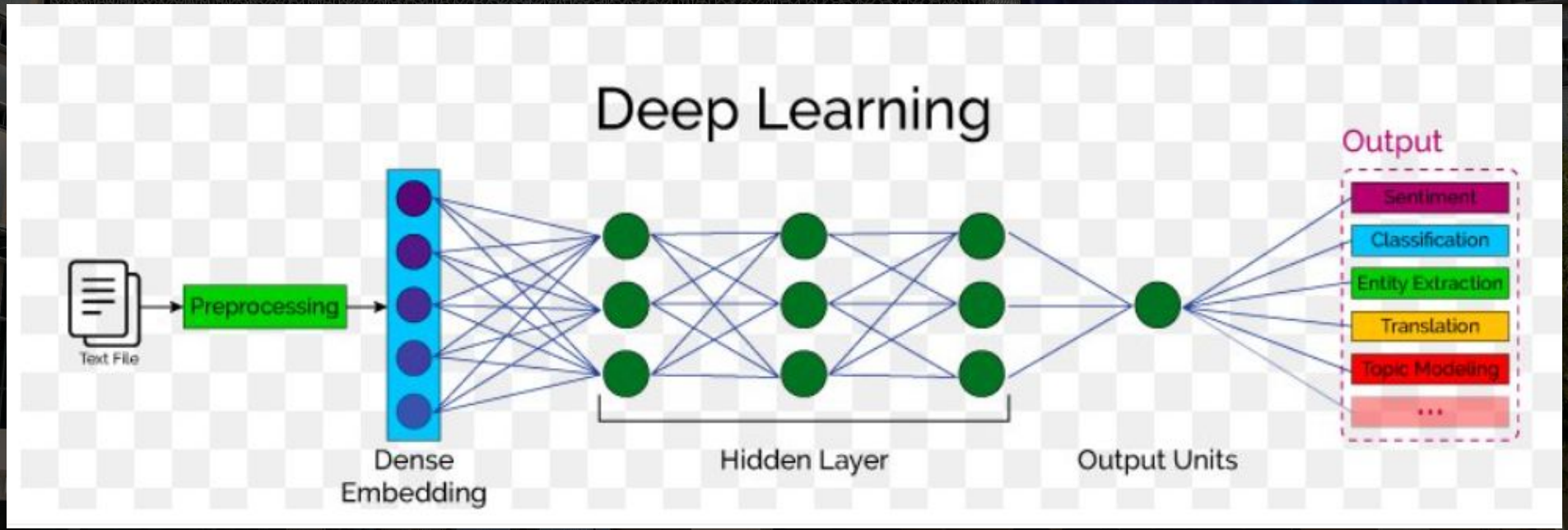
Example




```
model.add(Embedding(6, 2, embedding_initializer="uniform", input_length=5))
```

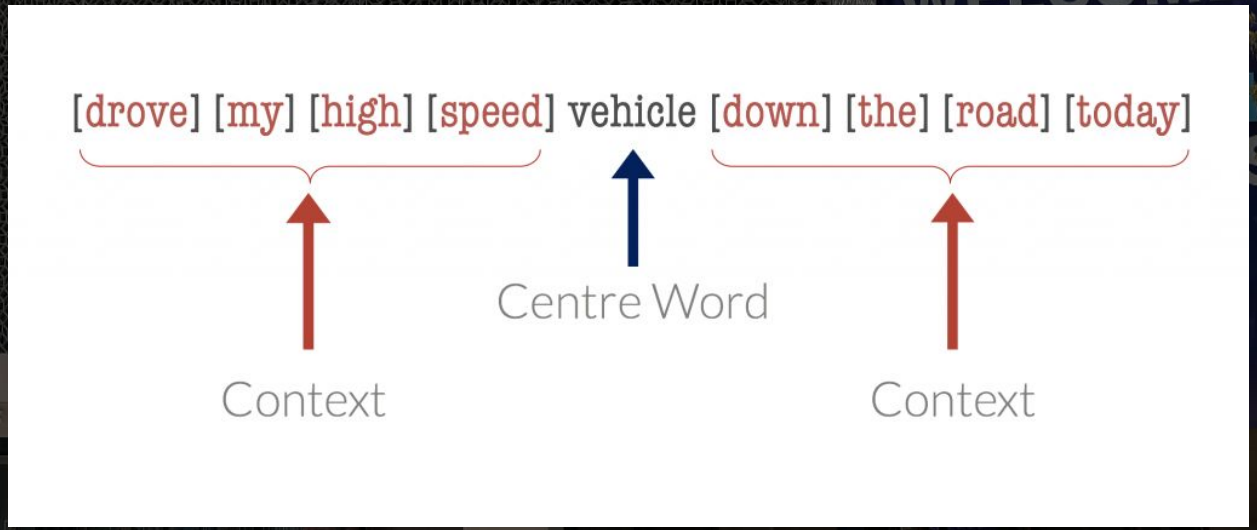
Vocabulary : 6
Vector dimensions: 2
Sentences length : 5

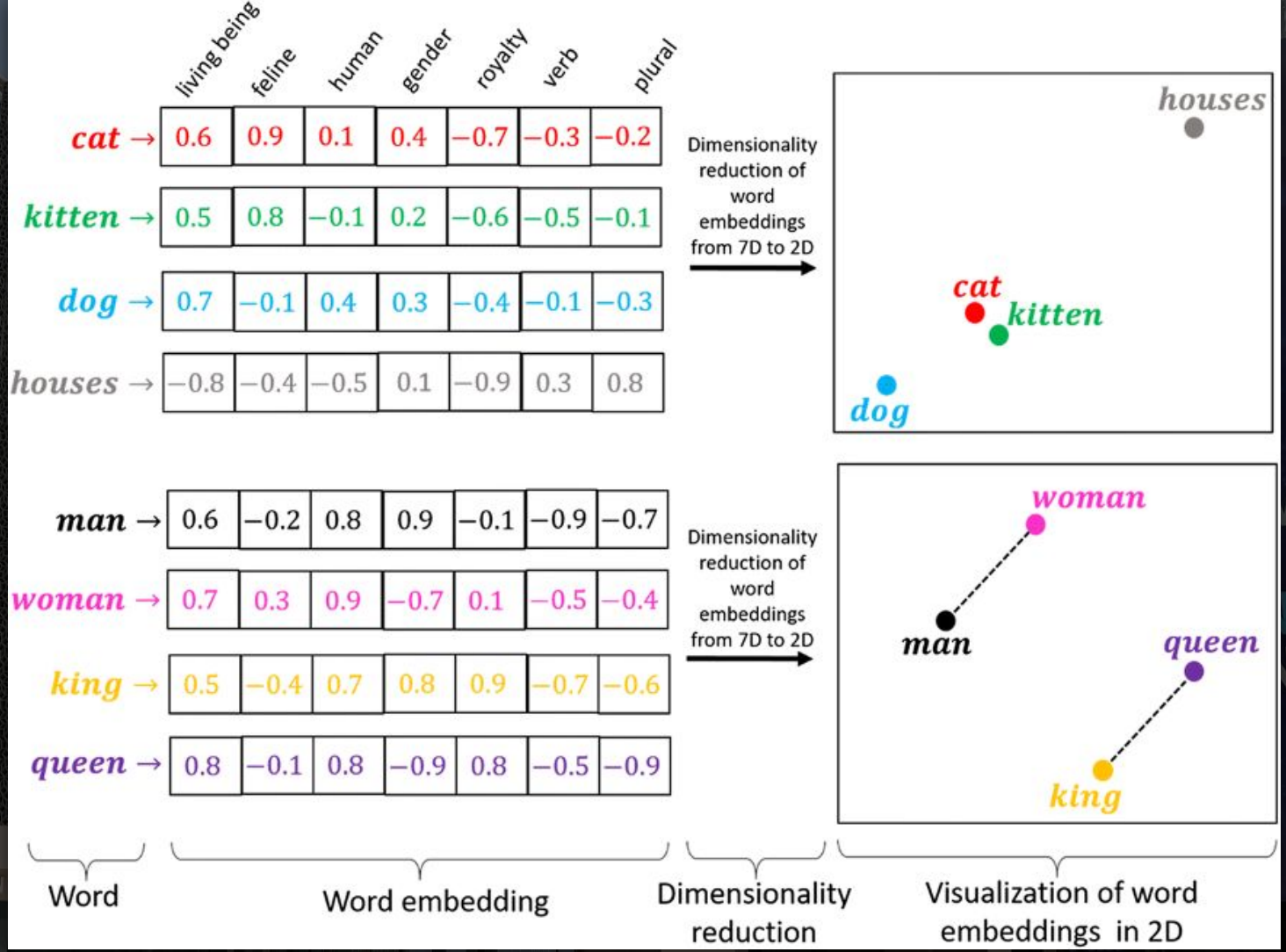
The values of vectors are a trainable parameters, initialized randomly and then learned by the model during training, in the same way a model learns weights for a dense layer



Word Embedding Training

Is trying to predict the context words from the center word or predict center word from the context word




living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Practical Example

- ✔ Dataset (CSV file)
- ✔ The Code