Unit 4.

Natural Language Processing with Keras

4.1. Natural Language Processing with Keras

Word Meaning

Meaning of "mouse" in Wordnet

Lemma

- mouse (N)

Senses

1. Any of the numerous small rodents …
2. A hand-operated electronic device that controls a cursor …
3. …

Polysemy

Some relationships between senses

- **Synonymy** (the same meaning)
  1. couch / sofa
  2. car / automobile
  3. big / large
  4. water / H2O
- **Similarity** (similar meanings or uses)
  1. car / bicycle
  2. cat / dog
  3. coffee / tea
- **Antonymy** (the opposite meaning)
  1. dark / light
  2. up / down
  3. hot / cold
- **Word relatedness** (different meanings used in a semantic domain or field)
  1. coffee / cup
  2. surgeon / nurse / hospital
  3. menu / food / restaurant

Word semantics and embeddings. Dan Jurafsky. https://web.stanford.edu/~jurafsky/

Word Meaning

Vector semantics

- Meaning of a word depends on its relationship with other words (or meanings)
- Meaning is a point in a multidimensional space based on distribution
- **Each word = a vector**
- Similar words are **nearby in semantic space**
- Semantic space is built automatically **by seeing wich words are nearby in text**

Word Meaning

Two words are similar in meaning if their context vectors are similar

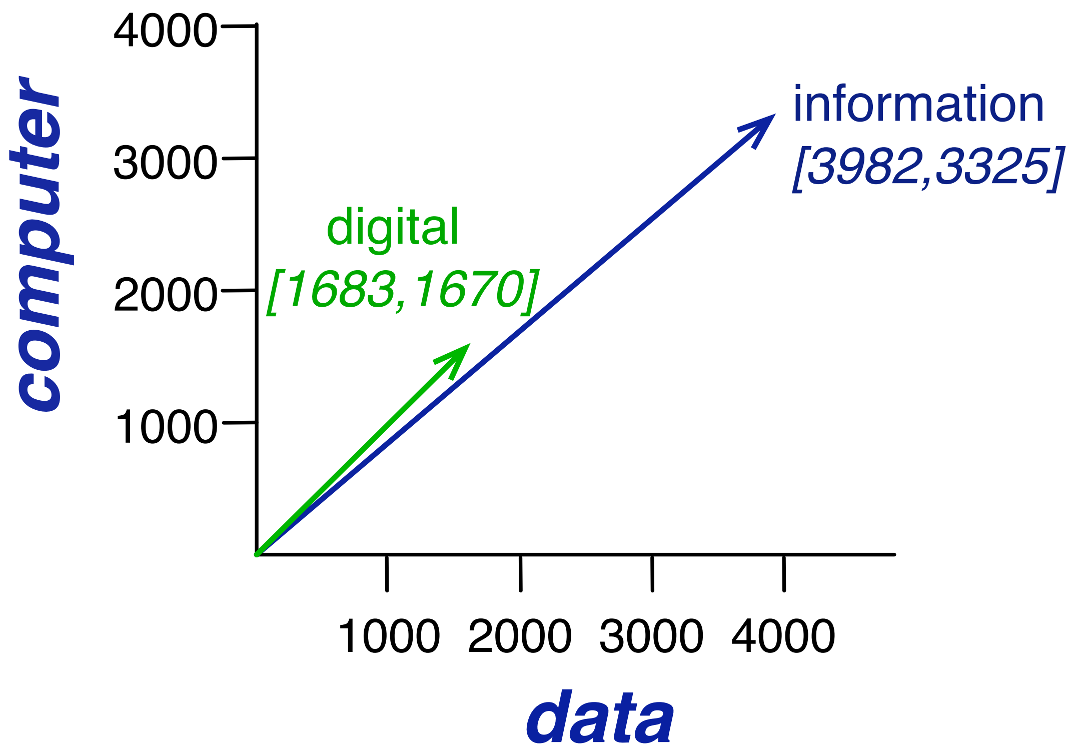
is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

word-word matrix

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

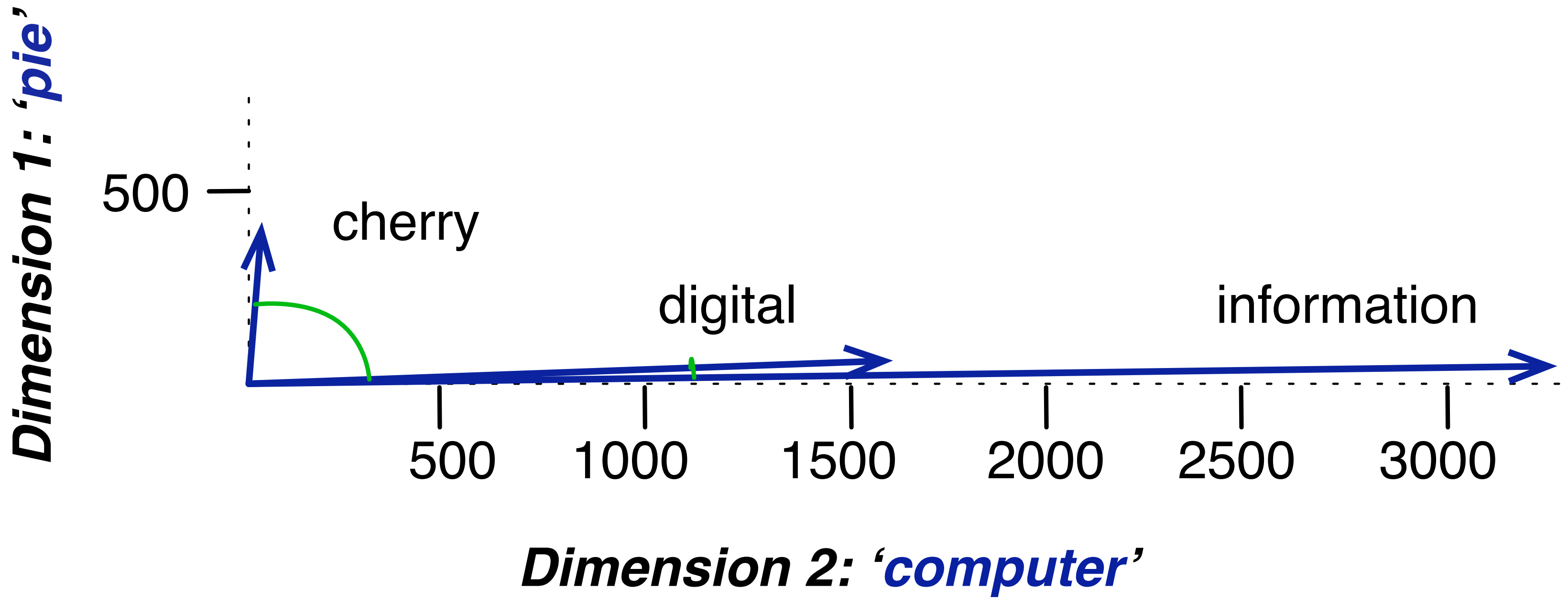
Word Meaning

- Example of projection in a bidimensional space



Word Meaning

- The most used similarity metric is **cosine**



Natural Language Processing with Keras

One-hot-encoding vs word embedding:

- We notice obvious problems with the one-hot-encoding representation.
- To improve, the “word embedding” is introduced which is a distributed representation method.

One-Hot-Encoding	Embedding
The dimension of the vector space is large. The dimension is as large as the vocabulary size. Dimension = |V| = 20,000 to 50,000	The dimension of the vector space is limited. Dimension = 50 - 1000
Vectors are sparse; they are mostly filled with 0s that carry no information.	Vectors are dense. Every vector element carries some information.
No semantic relationship among the vectors. The vectors are orthogonal to each other.	Semantic relationship among the vectors.

- There are also “paragraph embedding” and “document embedding” representations.
- We will call “dense vector” or “embedding vector” interchangeably.

One-hot-encoding vs word embedding:

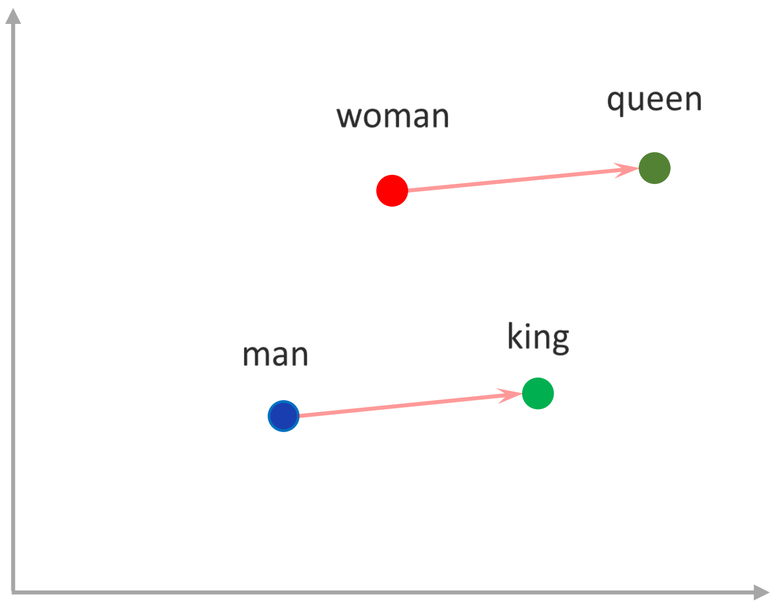
Ex Given a sentence “I eat an apple every morning”, let’s suppose that the words are indexed as:

I	: 3
Eat	: 0
An	: 2
Apple	: 1
Every	: 4
Morning	: 5

The words would have the following one-hot-encoding representations:

I	: [0 0 0 1 0 0]
Eat	: [1 0 0 0 0 0]
An	: [0 0 1 0 0 0]
Apple	: [0 1 0 0 0 0]
Every	: [0 0 0 0 1 0]
Morning	: [0 0 0 0 0 1]

Word embedding (Word2Vec):



- Among the dense vectors, relationships such as following are established:

$$queen - woman = king - man$$

man is a king as woman is a queen

woman + king – man = queen

Word embedding (Word2Vec):

- Use CBOW (Continuous Bag of Words) and/or Skip-Gram to build the embedding vectors.

  1). Build a predictive model based on the Softmax regression (multi-class logistic regression).

  2). We assume one-hot-encoded input and output vectors.

  3). Extract the embedding vectors from the trained weight matrices.

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Training Sentence	Center Word	Context Words
I eat an apple every morning	I	eat
I eat an apple every morning	eat	I, an
I eat an apple every morning	an	eat, apple
I eat an apple every morning	apple	an, every
I eat an apple every morning	every	apple, morning
I eat an apple every morning	morning	every

- We assumed a “sliding window” over the training sentence.

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex Let's suppose that the vector dimension of the one-hot-encoded words = 6.

Let's also suppose that we would like to find dense vectors of dimension = 3.

We will consider two context words (one from the left and another from the right).

So, we have a situation as the following:

I eat an apple every morning

I eat an ___?___ every morning

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex (continues from the previous page)

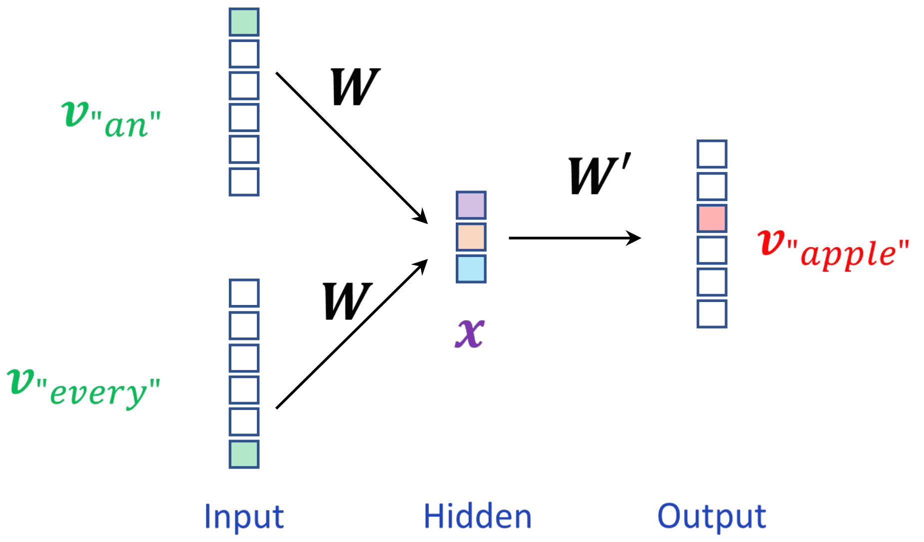
Then, we build a Softmax regression model.

For the vector inputs of “an” and “every”,

we would like to train the weights $\boldsymbol{W}$

and $\boldsymbol{W}'$ such that the predicted output is

the vector “apple”.



One-hot-encoded words : $v_{"an"}$ =[1 0 0 0 0 0] , $v_{"every"}$ =[0 0 0 0 0 1] , $v_{"apple"}$ =[0 0 1 0 0 0]

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex (continues from the previous page)

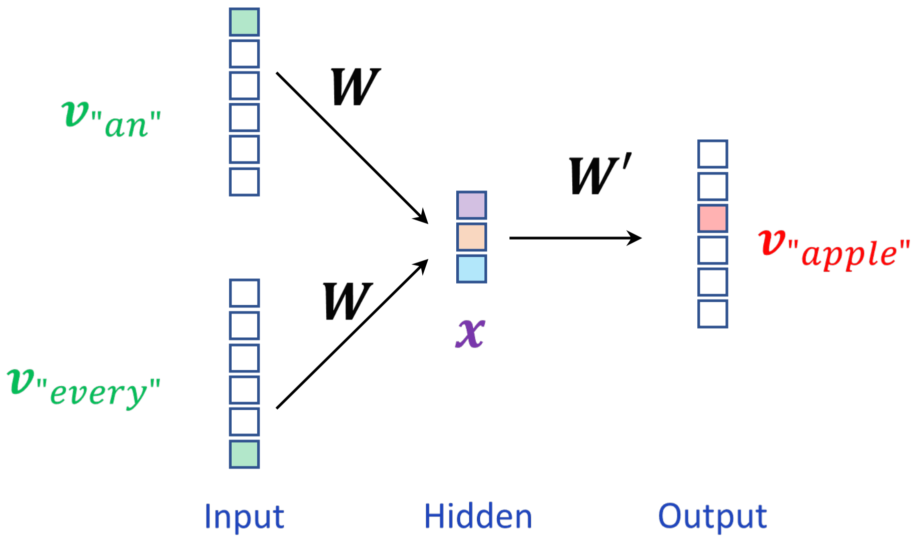
We have the following sizes:

Size of the matrix $\boldsymbol{W}$=3×6

Size of the matrix $\boldsymbol{W}'$=6×3

Dimension of the vector $\boldsymbol{x}$= 3.

Dimension of the input and output = 6.



Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex (continues from the previous page)

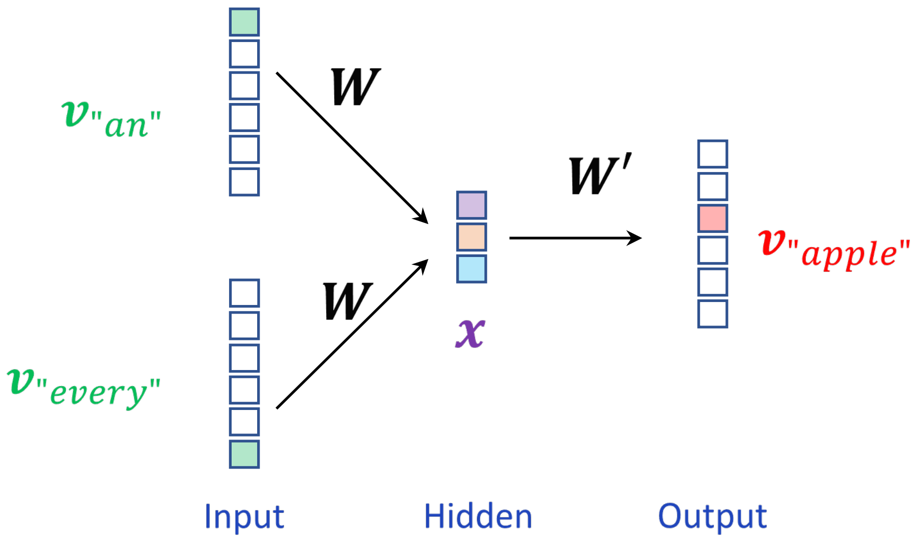
We propagate forward from the input layer

to the hidden layer (a single node):

$\boldsymbol{x}_{\text{"an"}} = \boldsymbol{W} \cdot \boldsymbol{v}_{\text{"an"}}$

$\boldsymbol{x}_{\text{"every"}} = \boldsymbol{W} \cdot \boldsymbol{v}_{\text{"every"}}$

$\boldsymbol{x} = \frac{\boldsymbol{x}_{\text{"an"}} + \boldsymbol{x}_{\text{"every"}}}{2}$ ⇐ Average for the hidden node.



Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex (continues from the previous page)

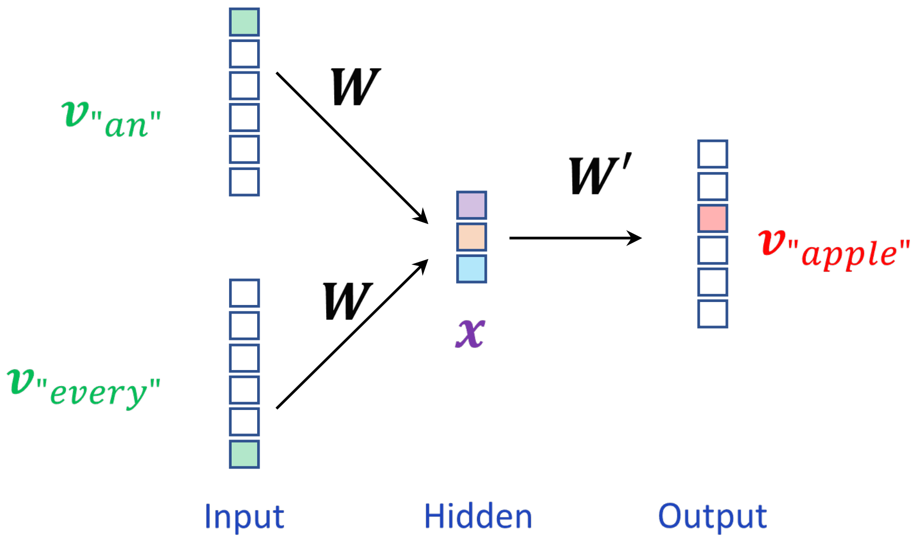
We propagate forward to the output layer:

$\acute{\boldsymbol{v}} = \boldsymbol{W}' \cdot \boldsymbol{x}$

We should train the weights $\boldsymbol{W}$ and $\boldsymbol{W}'$

argmax$(\acute{\boldsymbol{v}})$ and argmax$(\boldsymbol{v}_{"apple"})$

is minimized.



Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex (continues from the previous page)

Now, let's interpret the result.

a). When we propagate from the input layer to the hidden layer (by matrix multiplication),

the one-hot-encoded input vectors $\boldsymbol{v}_{"an"}$ and $\boldsymbol{v}_{"every"}$ are picking the columns 0 and 5

of $\boldsymbol{W}$ and projecting them to the hidden layer with the target dimension = 3.

b). So, the dense vectors for "an", "every" are the columns 0 and 5 of the trained $\boldsymbol{W}$.

c). Analogously, we can extract dense vectors from the rows of the trained $\boldsymbol{W}'$.

Word embedding (Word2Vec):

- Skip-Gram: Using a center word, predict the (missing) context words.

I eat an apple every morning

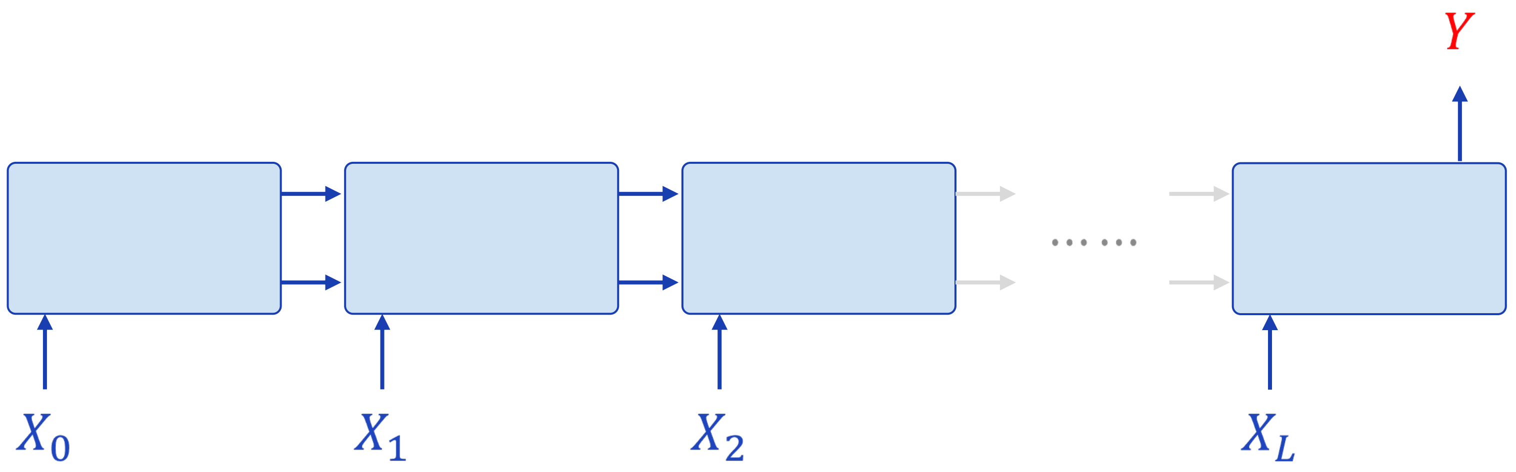
↓

I eat ?__ apple __?__ morning

- Similar to the CBOW, here also we train a Softmax regression to predict the missing words.
- We extract the dense vectors (embedding vectors) from the trained weight matrices.

LSTM network for document classification:

- "Sequence in and Vector out" model.
- Embedding representation of the words.



LSTM network for document classification: a code example.

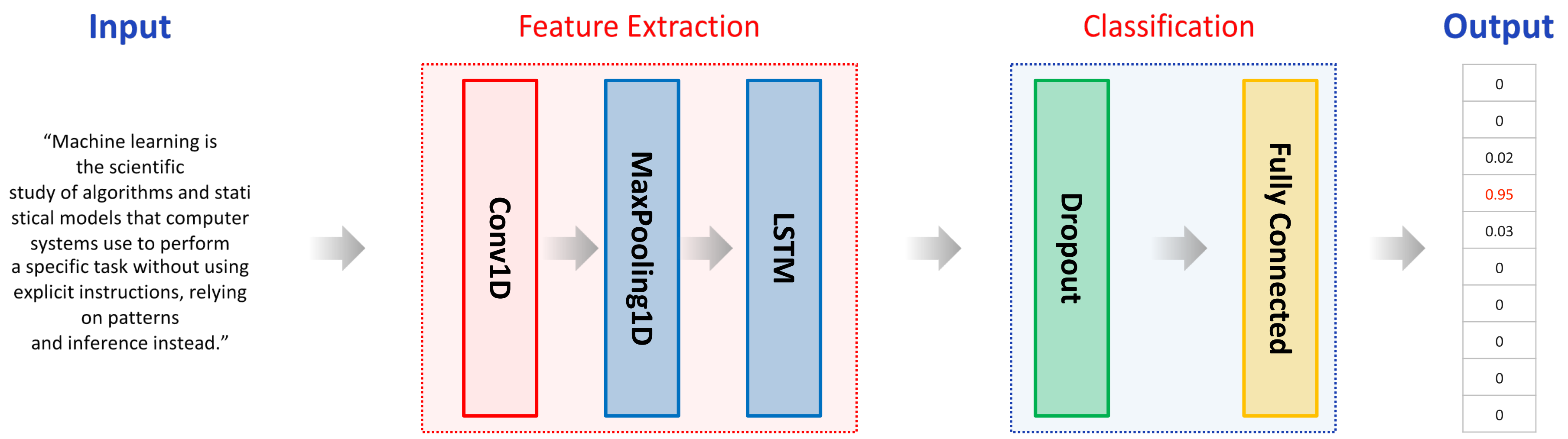
```
# Import the necessary classes.
from keras.models import Sequential                    # We will use the Sequential API.
from keras.layers import Dense, LSTM, Embedding
```

```
# Build a model by adding the layers.
my_model = Sequential()
my_model.add(Embedding(n_words,n_input))
my_model.add(LSTM(units=n_neurons, return_sequences=False, input_shape=(None, n_input), activation='tanh'))
my_model.add(Dense(1, activation='sigmoid'))
```

- In LSTM(), we should set *return_sequences=False* for a “Sequence in and Vector out” model.

Deep learning model for document classification:

- 1D convolution + 1D max pooling + LSTM for the feature extraction.
- Localized sequence patterns picked up by the 1D convolution.



Deep learning model for document classification: a code example.

```
# Import the necessary classes.
from keras.models import Sequential                    # We will use the Sequential API.
from keras.layers import Dense, LSTM, Embedding, Conv1D, MaxPool1D, Dropout
```

```
# Build a model by adding the layers.
my_model = Sequential()
my_model.add(Embedding(n_words, n_input))              # Embedding layer.
my_model.add(Conv1D(filters=n_filters, kernel_size = k_size, strides=stride_size,padding='valid',activation='relu'))
my_model.add(MaxPool1D(pool_size = 2))
my_model.add(LSTM(units=n_neurons, return_sequences=False, input_shape=(None, n_input), activation='tanh'))
my_model.add(Dropout(rate=hold_prob))
my_model.add(Dense(1, activation='sigmoid'))
```

Coding Exercise #0514

Follow practice steps on 'ex_0514.ipynb' file

Coding Exercise #0515

Follow practice steps on 'ex_0515.ipynb' file

Coding Exercise #0516

Follow practice steps on 'ex_0516.ipynb' file

Coding Exercise #0517

Follow practice steps on 'ex_0517.ipynb' file