

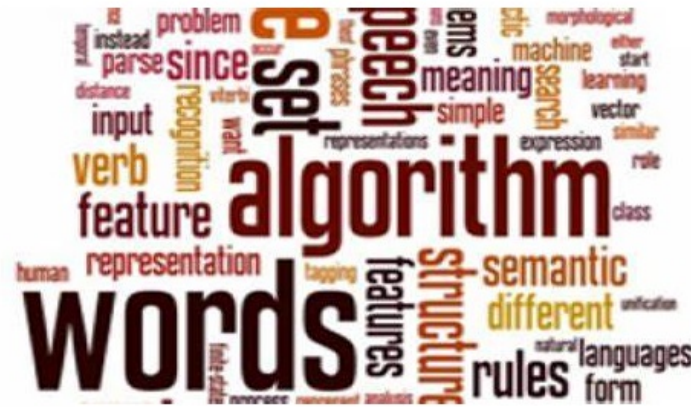
# Introduction to the course

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON



**David Cecchini**  
Data Scientist

# Text data is available online

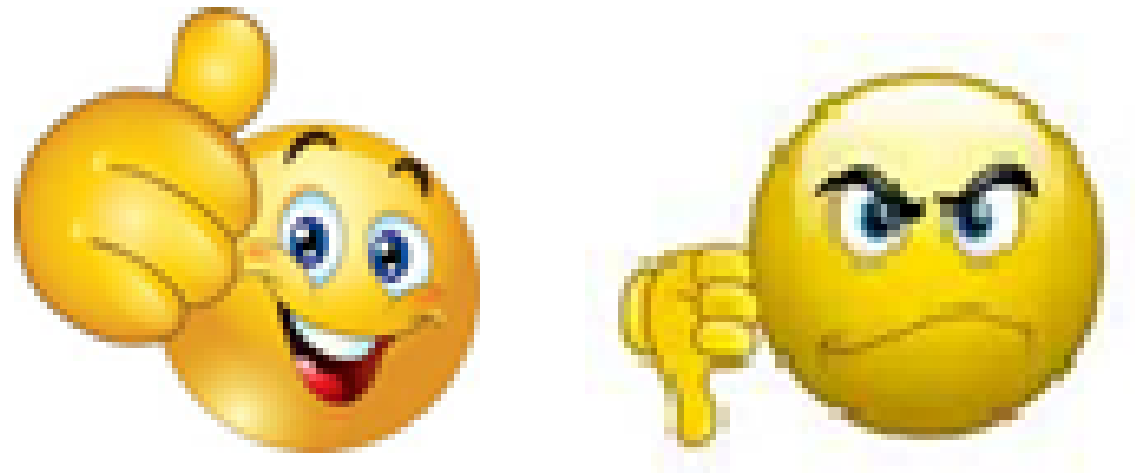


# Applications of machine learning to text data

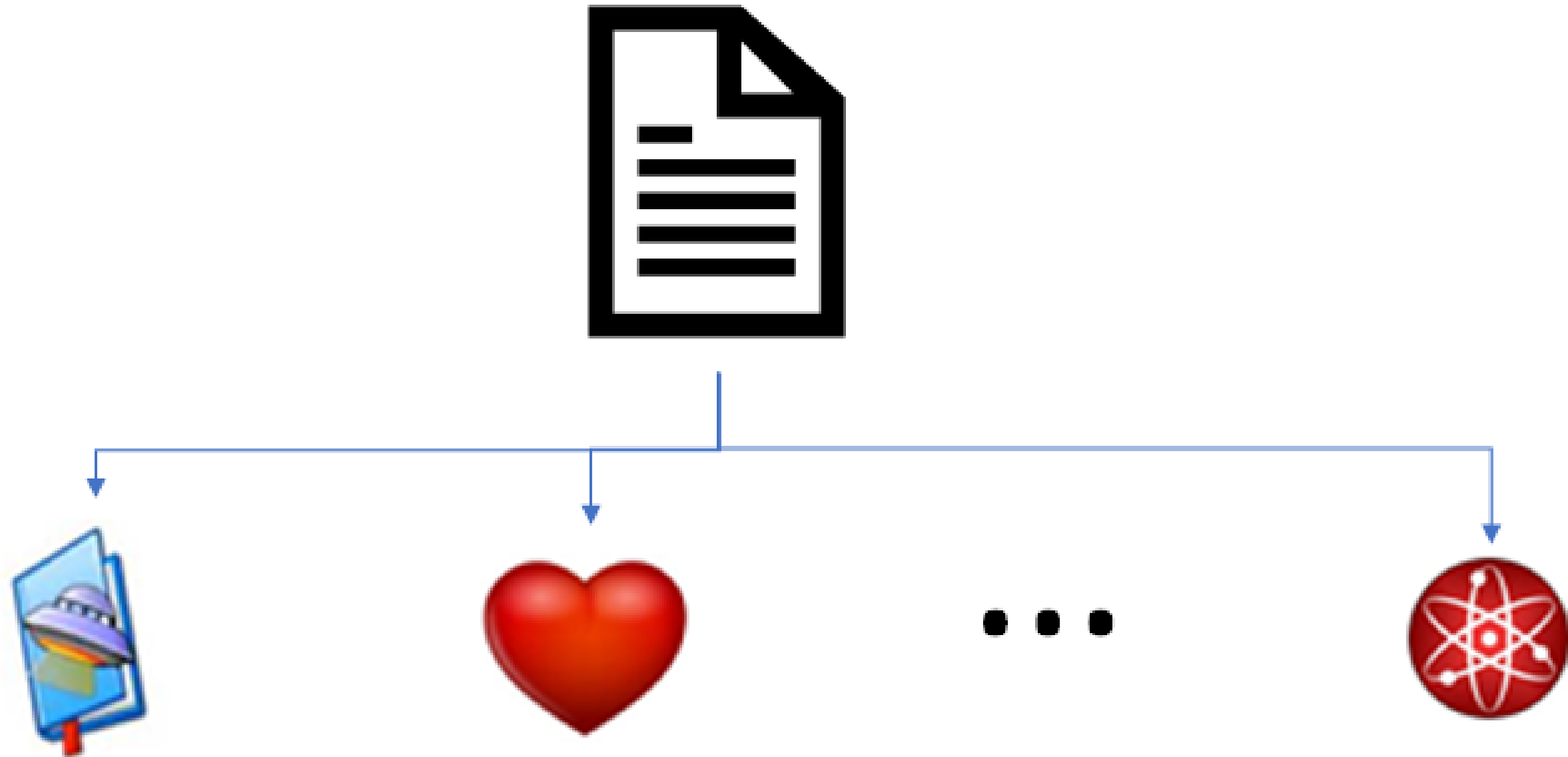
Four applications:

- Sentiment analysis
- Multi-class classification
- Text generation
- Machine neural translation

# Sentiment analysis



# Multi-class classification



# Text generation

David Amore Cecchini

to me ▼

Have you heard?

Next World Cup is going to be awesome, and you will be rooting for the Seleção, right?

Best,

Yes!

No, I haven't.

Not yet!



# Neural machine translation

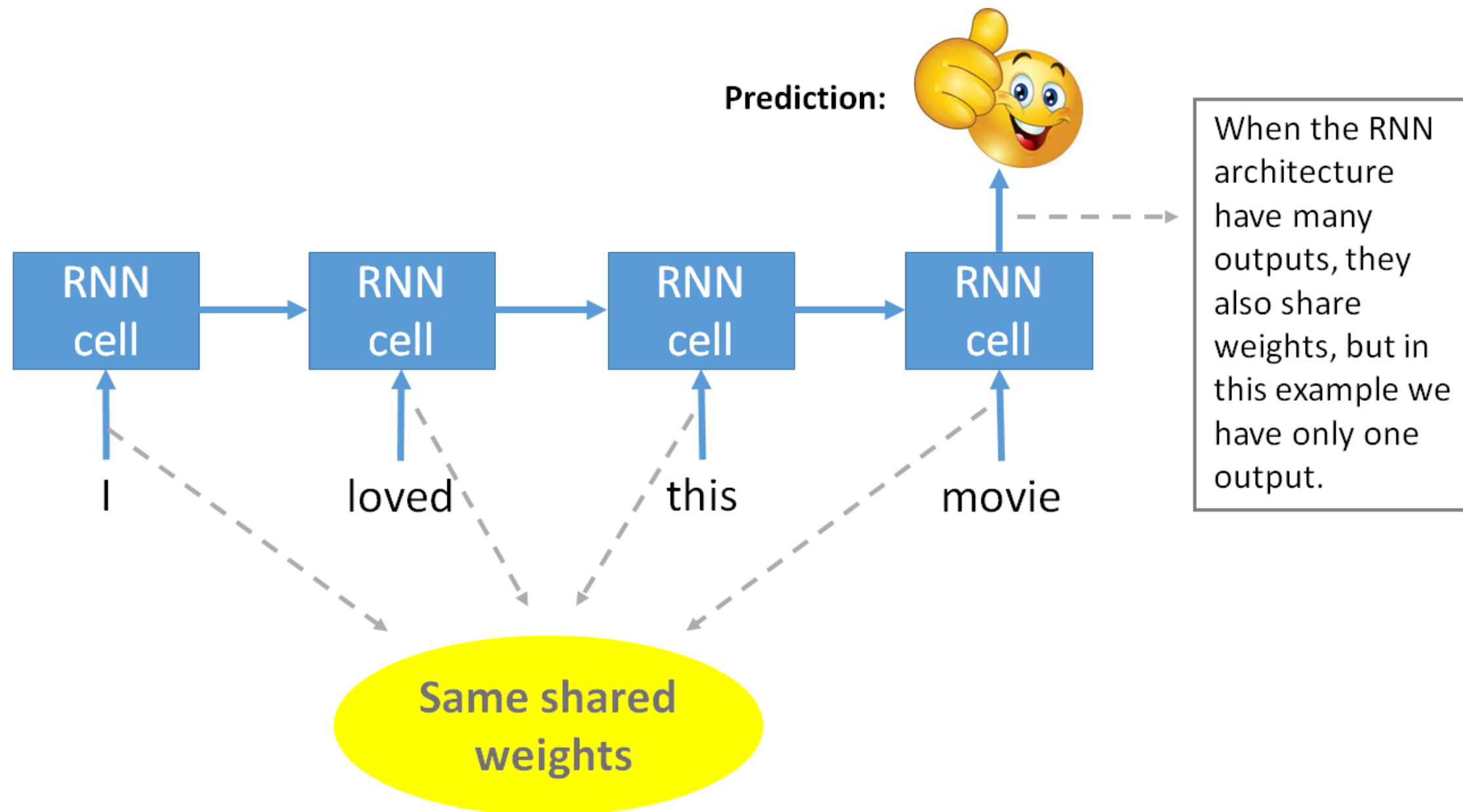
PT

Vamos jogar futebol esse domingo?

EN

Let's play soccer this Sunday?

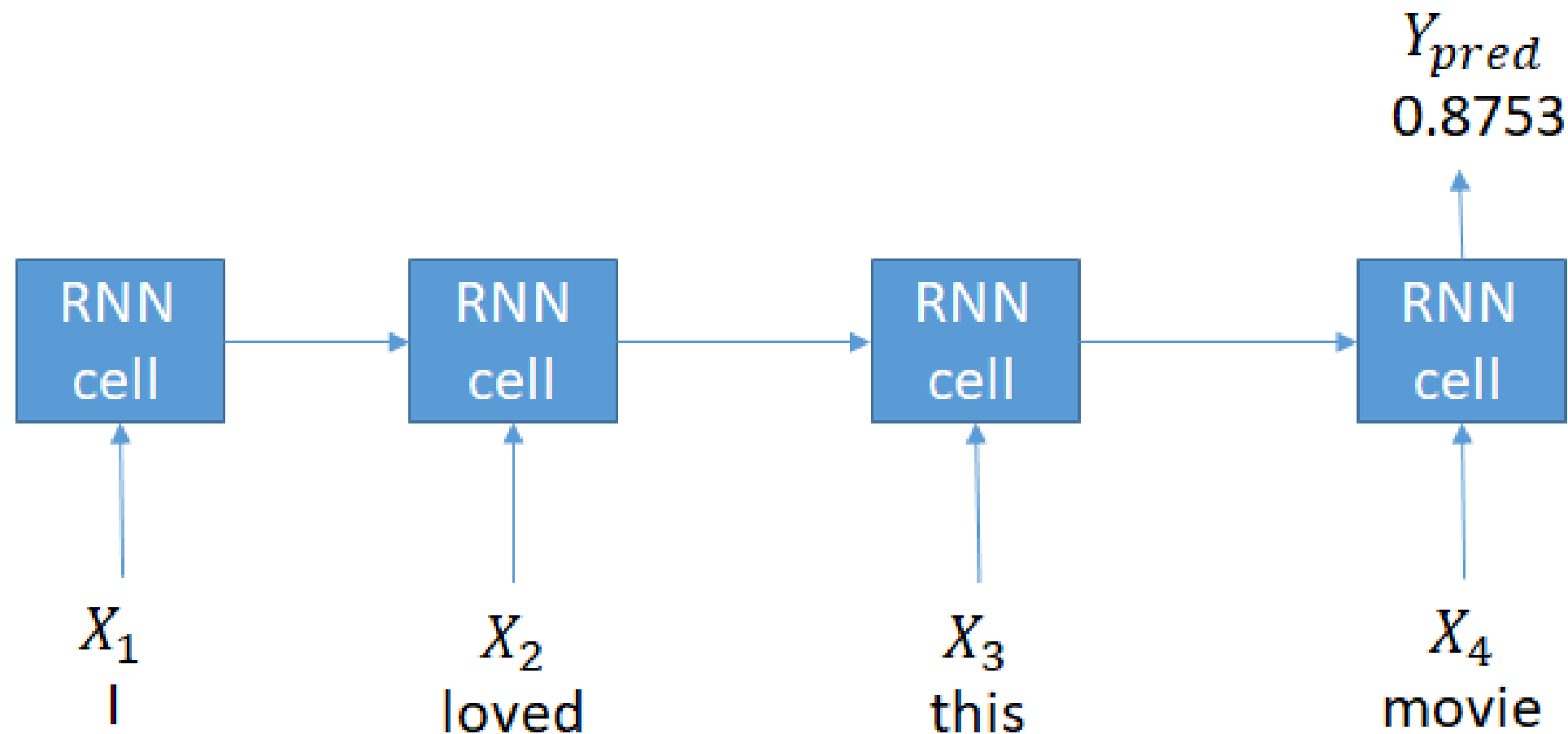
# Recurrent Neural Networks





# Sequence to sequence models

Many to one: classification



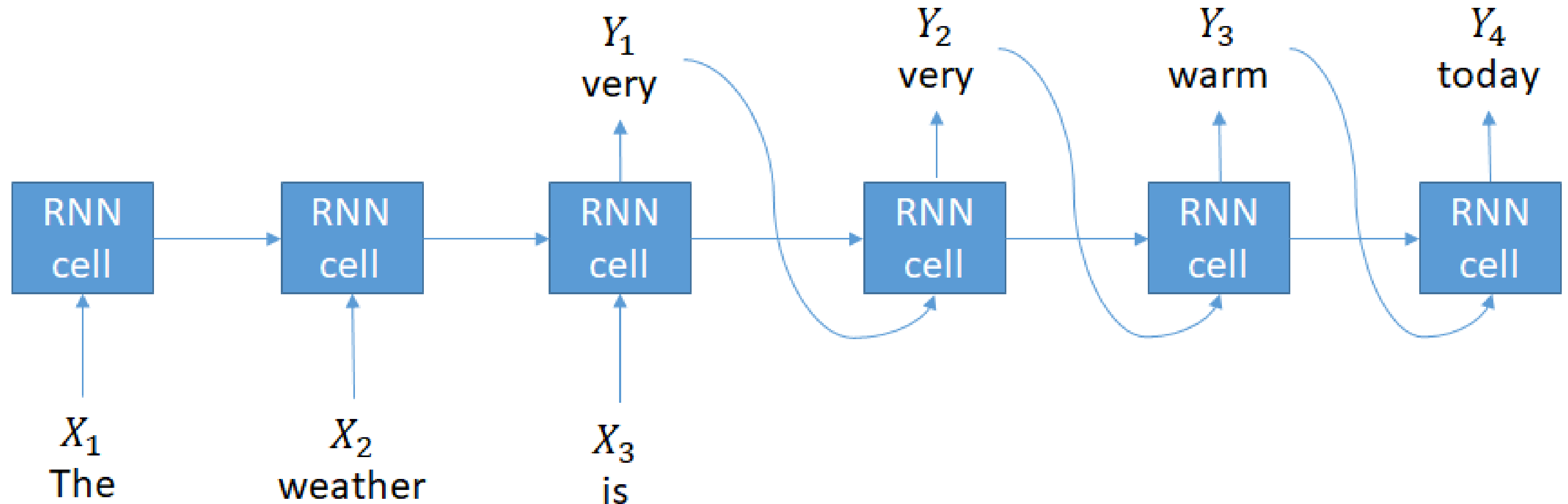
## Decision rule:

Set prediction to "positive" if  $Y_{pred} > 0.5$ , otherwise set to "negative".

\* $Y_{pred}$  is the probability of the sentence to belong to class "positive"

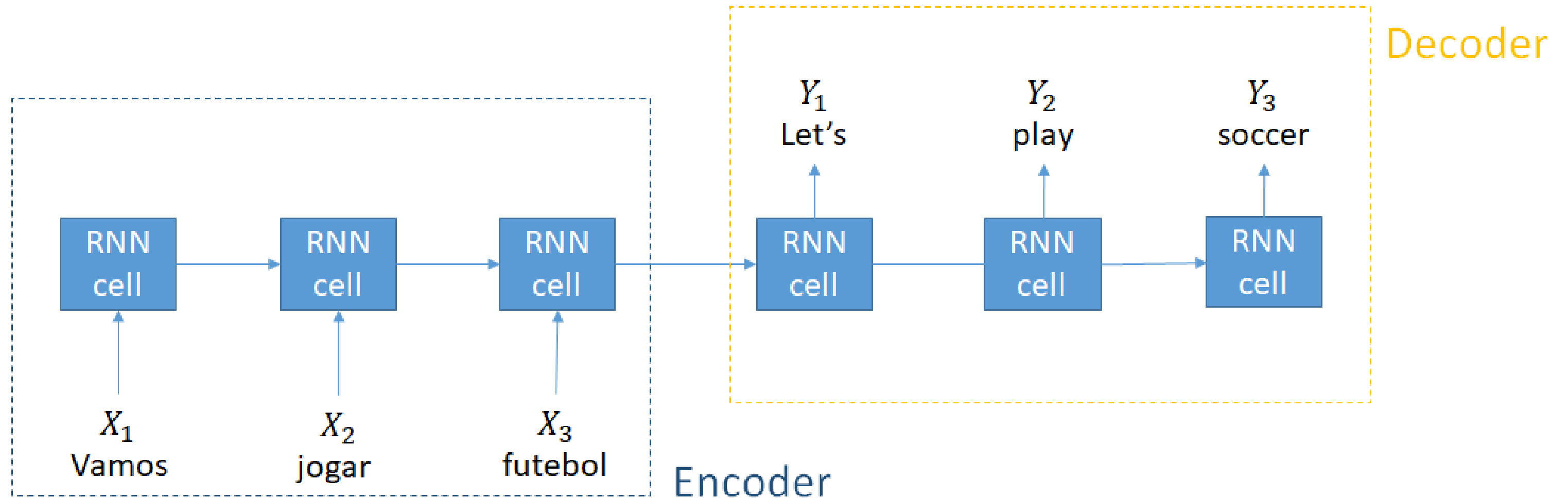
# Sequence to sequence models

Many to many: text generation



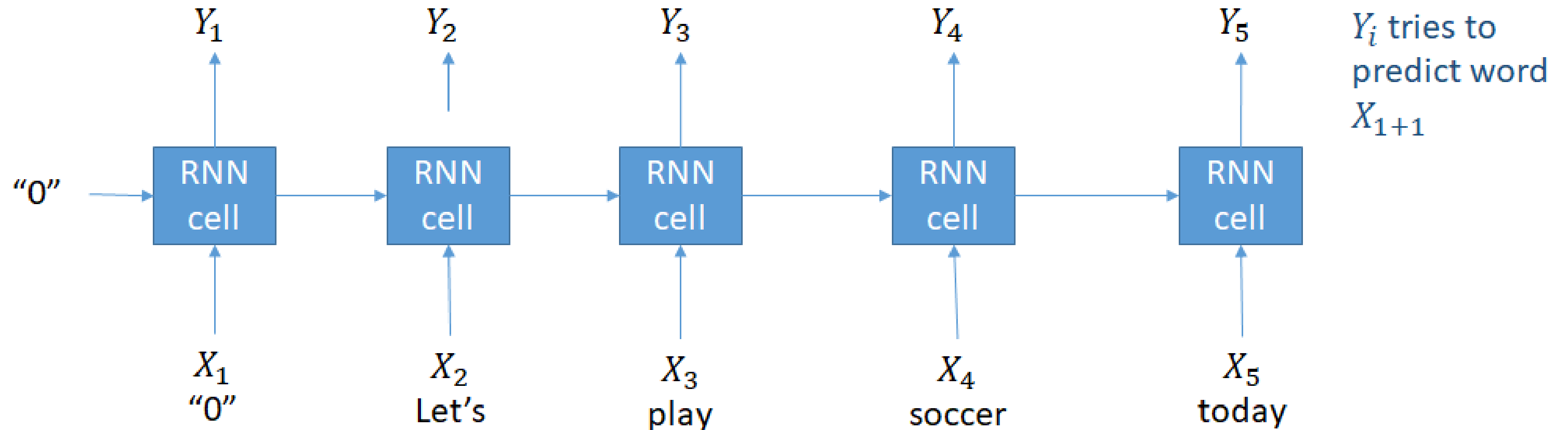
# Sequence to sequence models

Many to many: neural machine translation



# Sequence to sequence models

Many to many: language model



# Let's practice!

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON

# Introduction to language models

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON



**David Cecchini**  
Data Scientist

# Sentence probability

Many available models

- Probability of "I loved this movie".

- Unigram

- $P(\text{sentence}) = P(\text{I})P(\text{loved})P(\text{this})P(\text{movie})$

- N-gram

- N = 2 (bigram):

- $P(\text{sentence}) = P(\text{I})P(\text{loved}|\text{I})P(\text{this}|\text{loved})P(\text{movie}|\text{this})$

- N = 3 (trigram):

- $P(\text{sentence}) = P(\text{I})P(\text{loved}|\text{I})P(\text{this}|\text{I loved})P(\text{movie}|\text{loved this})$

# Sentence probability (cont.)

- Skip gram

- $P(\text{sentence}) = P(\text{context of I}|\text{I})P(\text{context of loved}|\text{loved})$

$$P(\text{context of this}|\text{this})P(\text{context of movie}|\text{movie})$$

- Neural Networks

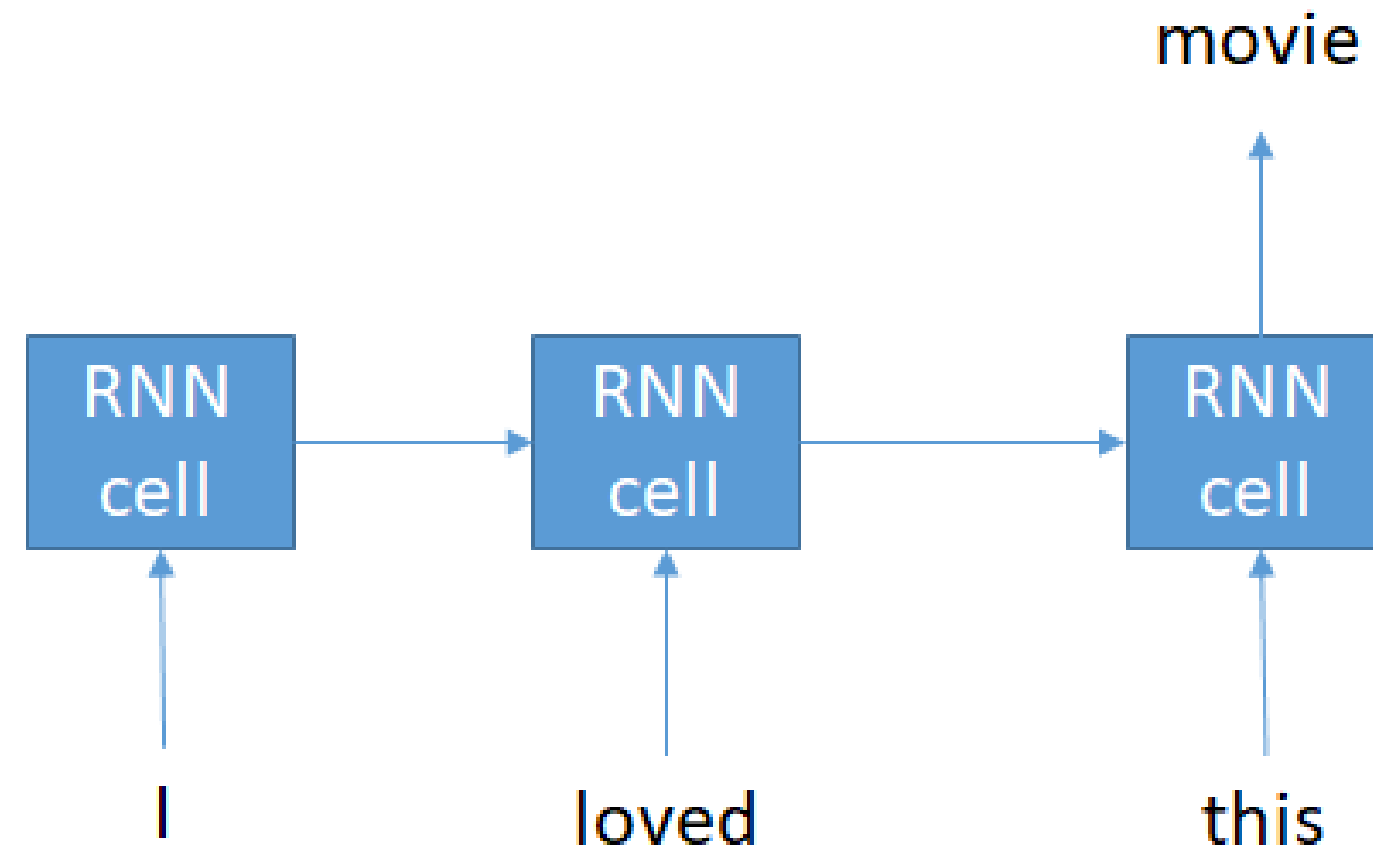
- The probability of the sentence is given by a `softmax` function on the output layer of the network



# Link to RNNs

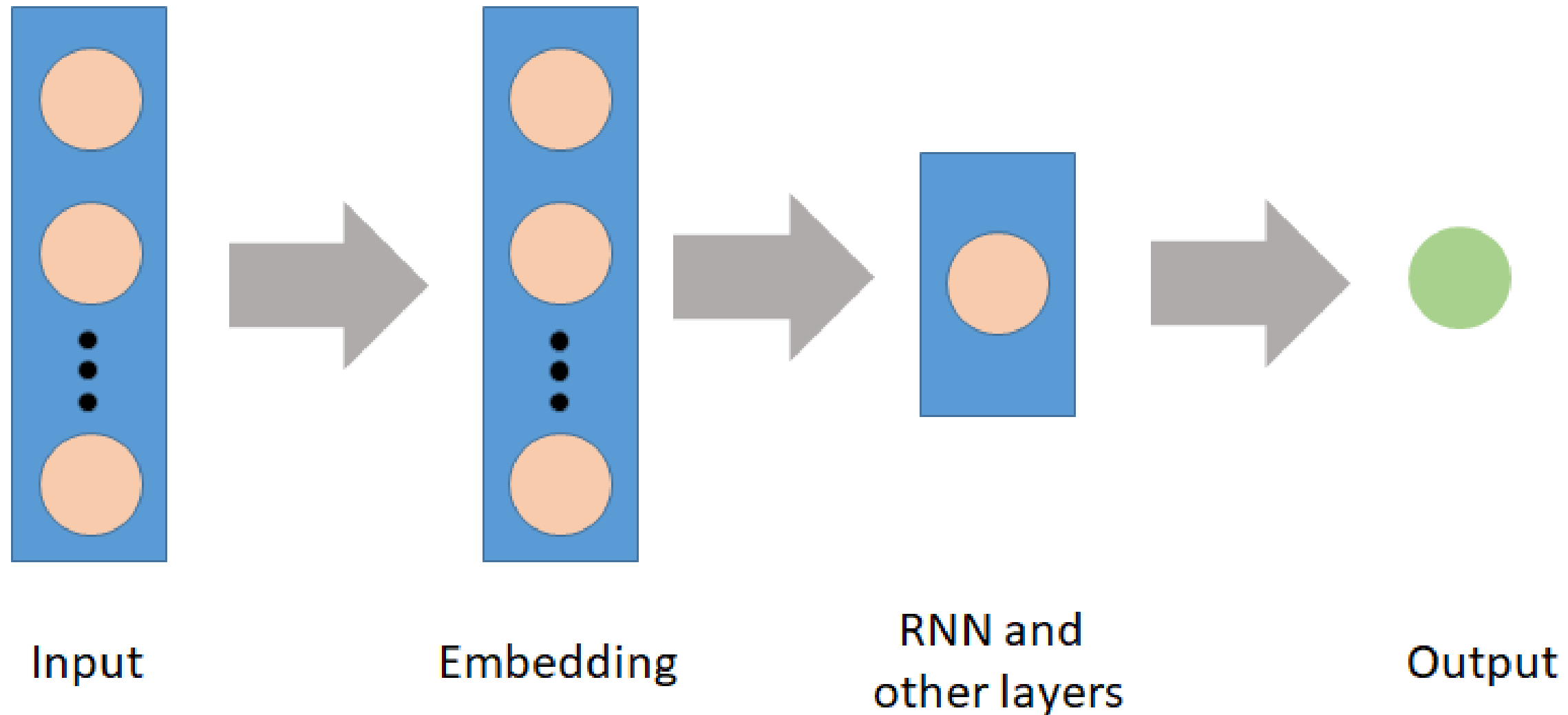
Language models are everywhere in RNNs!

- The network itself



# Link to RNN (cont.)

- Embedding layer



# Building vocabulary dictionaries

```
# Get unique words
```

```
unique_words = list(set(text.split(' ')))
```

```
# Create dictionary: word is key, index is value
```

```
word_to_index = {k:v for (v,k) in enumerate(unique_words)}
```

```
# Create dictionary: index is key, word is value
```

```
index_to_word = {k:v for (k,v) in enumerate(unique_words)}
```

# Preprocessing input

```
# Initialize variables X and y
X = []
y = []
# Loop over the text: length `sentence_size` per time with step equal to `step`
for i in range(0, len(text) - sentence_size, step):
    X.append(text[i:i + sentence_size])
    y.append(text[i + sentence_size])
```

```
# Example (numbers are numerical indexes of vocabulary):
# Sentence is: "i loved this movie" -> (["i", "loved", "this"], "movie")
X[0], y[0] = ([10, 444, 11], 17)
```

# Transforming new texts

```
# Create list to keep the sentences of indexes
new_text_split = []
# Loop and get the indexes from dictionary
for sentence in new_text:
    sent_split = []
    for wd in sentence.split(' '):
        ix = wd_to_index[wd]
        sent_split.append(ix)
    new_text_split.append(sent_split)
```

# Let's practice!

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON

# Introduction to RNN inside Keras

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON



**David Cecchini**  
Data Scientist

# What is keras?

- High-level API
- Can use Tensorflow, CNTK or Theano frameworks
- Easy to install and use

```
$pip install keras
```

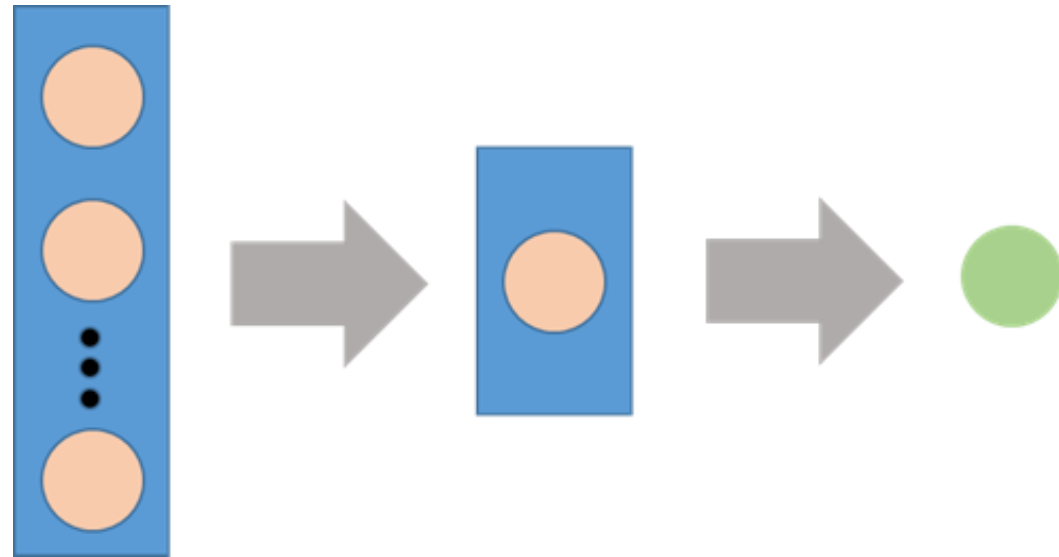
Fast experimentation:

```
from keras.models import Sequential  
from keras.layers import LSTM, Dense
```

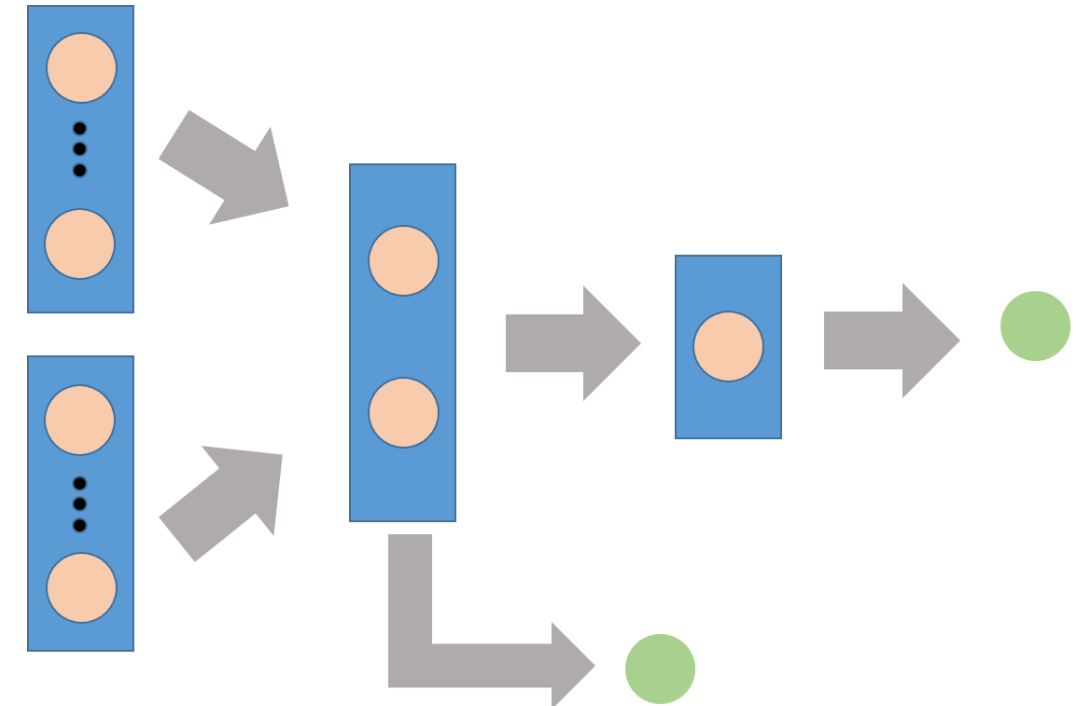


# keras.models

keras.models.Sequential



keras.models.Model



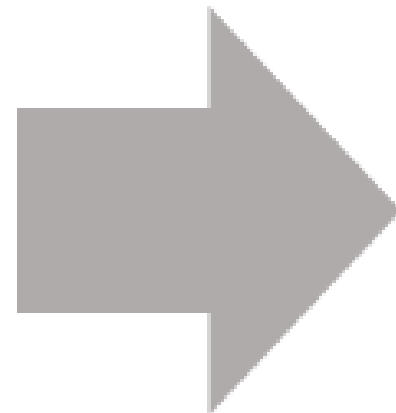
# keras.layers

1. LSTM
2. GRU
3. Dense
4. Dropout
5. Embedding
6. Bidirectional

# keras.preprocessing

```
keras.preprocessing.sequence.pad_sequences(texts, maxlen=3)
```

|        |       |          |      |
|--------|-------|----------|------|
| movie  | was   | great    |      |
| really | bad   |          |      |
| i      | loved | it       |      |
| actor  | is    | handsome |      |
| s2     |       |          |      |
| the    | best  | movie    | ever |
| could  | be    | better   |      |
| meh    |       |          |      |



|       |        |          |  |
|-------|--------|----------|--|
| movie | was    | great    |  |
| "0"   | really | bad      |  |
| i     | loved  | it       |  |
| actor | is     | handsome |  |
| "0"   | "0"    | s2       |  |
| the   | best   | movie    |  |
| could | be     | better   |  |
| "0"   | "0"    | meh      |  |

# keras.datasets

## Many useful datasets

- IMDB Movie reviews
- Reuters newswire

And more!

For a complete list and usage examples, see [keras documentation](#)

# Creating a model

```
# Import required modules
from keras.models import Sequential
from keras.layers import Dense
```

```
# Instantiate the model class
model = Sequential()
```

```
# Add the layers
model.add(Dense(64, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
```

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])
```

# Training the model

The method `.fit()` trains the model on the training set

```
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

1. `epochs` determine how many weight updates will be done on the model
2. `batch_size` size of the data on each step

# Model evaluation and usage

Evaluate the model:

```
model.evaluate(X_test, y_test)
```

```
[0.3916562925338745, 0.89324]
```

Make predictions on new data:

```
model.predict(new_data)
```

```
array([[0.91483957], [0.47130653]], dtype=float32)
```

# Full example: IMDB Sentiment Classification

```
# Build and compile the model
model = Sequential()
model.add(Embedding(10000, 128))
model.add(LSTM(128, dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
# Training
model.fit(x_train, y_train, epochs=5)
```

```
# Evaluation
score, acc = model.evaluate(x_test, y_test)
```



# Time to practice!

RECURRENT NEURAL NETWORKS FOR LANGUAGE MODELING IN PYTHON