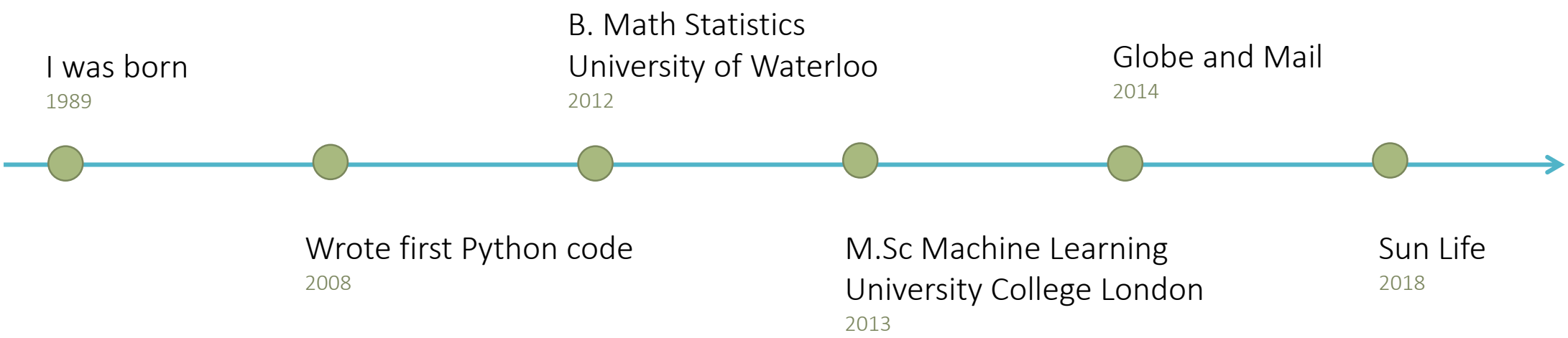
Modelling user journeys with Keras

Jennifer Nguyen

PyCon Canada 2019

My journey



jennguyen.ca/timeline

What

… is a user journey?

E-commerce site

VIEW

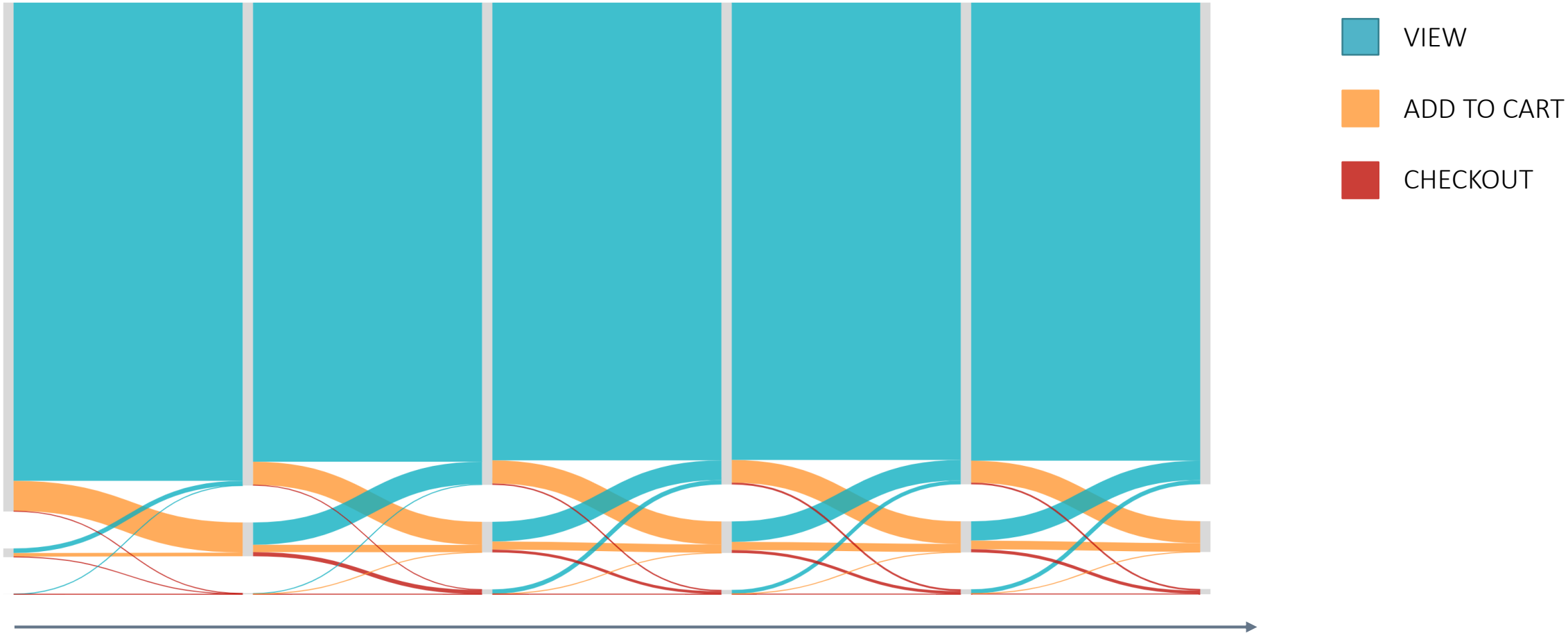
ADD TO CART

CHECKOUT

Can we predict a user's path?

A user journey is a sequence of actions

Retail Rocket User Journeys



Kaggle Ecommerce Dataset

Why

…do we care about user journeys?

User journeys can help

Course Correct

Diagnose

Resource Planning

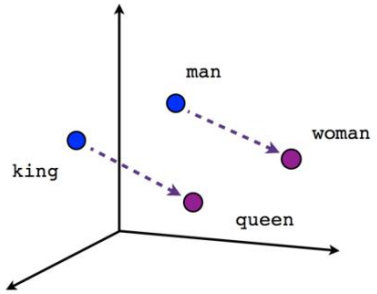
How

…can we predict a user’s journey?

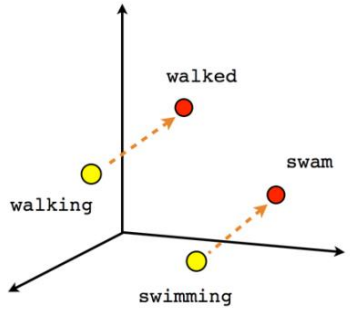
Embeddings

A vector representation of an entity, e.g., a word, an action, a user, etc.

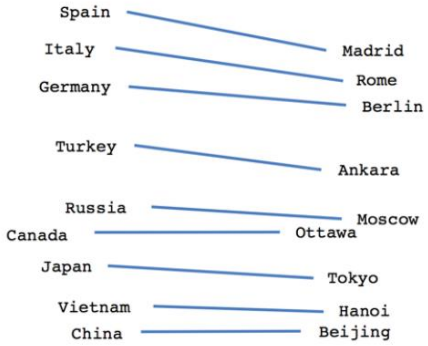
We want to map each action to a vector



Male-Female



Verb tense

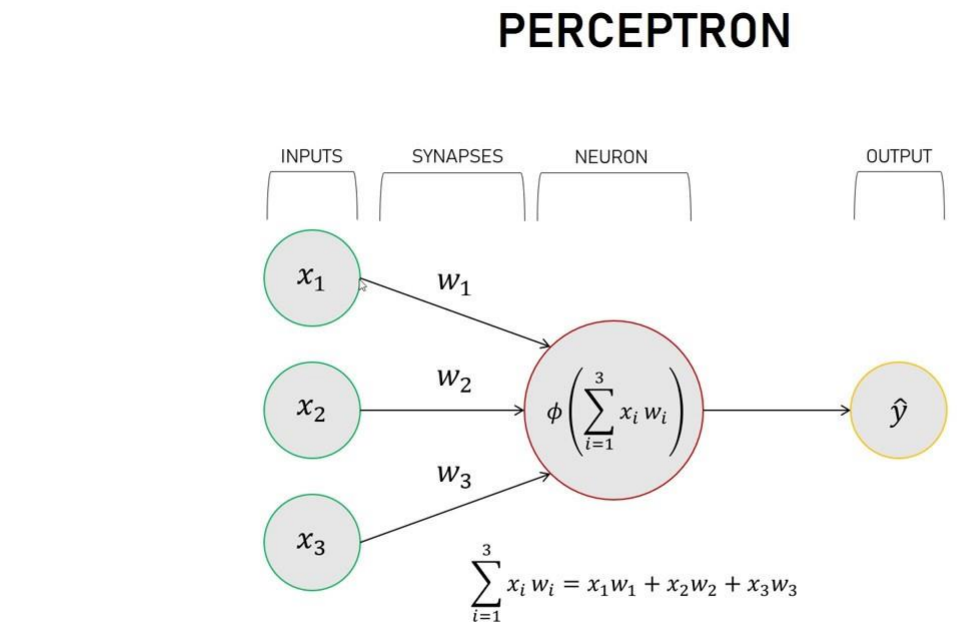


Country-Capital

Word2Vec: words are mapped to an n-dimensional vector space (Mikolov et al., 2013)

Neural Networks

A collection of neurons, each of which is a function mapping a set of inputs $x_1,...,x_n$ to an output y

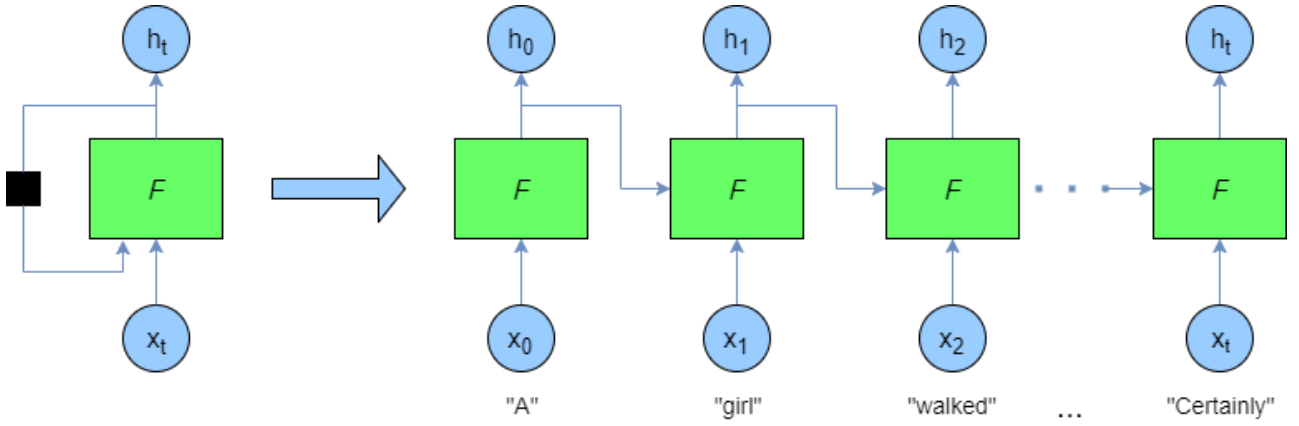


A neural network is composed of many perceptrons

Recurrent Neural Networks (RNN)

A sequence of neural networks where the output of the NN at time *t-1* is used as input for the NN at time *t*.

RNNs are helpful when order matters

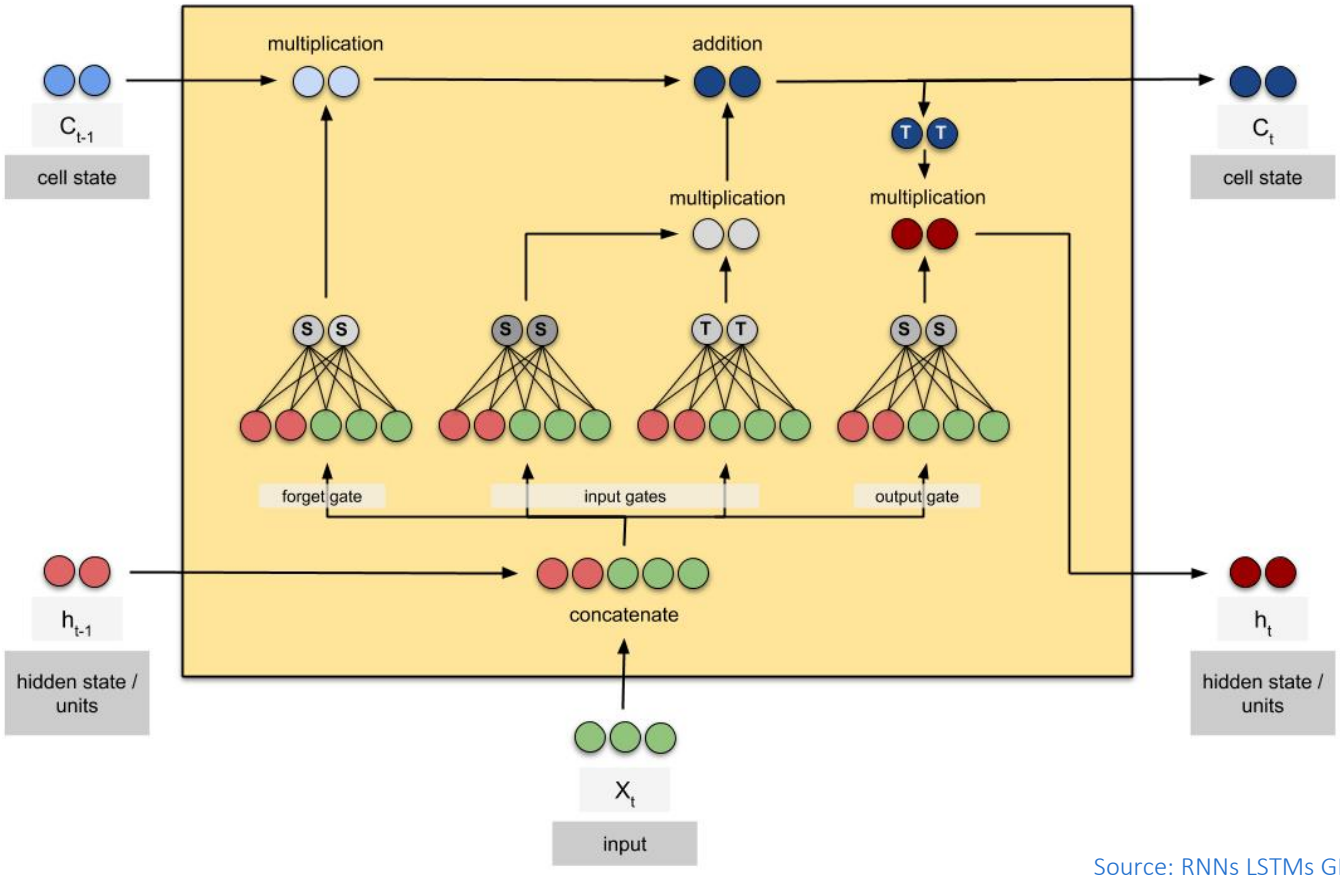


Source: Adventures in Machine Learning: Keras LSTM Tutorial

RNNs are able to “remember” information from the past to predict the future

Long Short Term Model (LSTM)

An RNN architecture that allows for finer control of the inputs to “remember” and the ones to “forget”

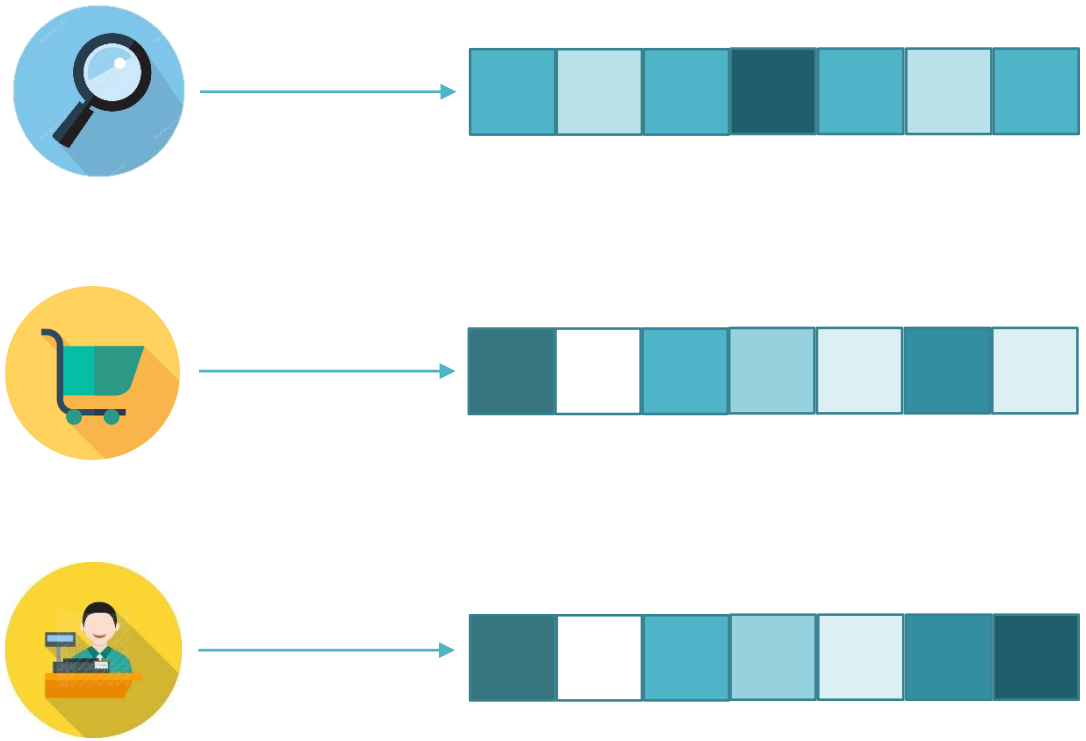


Source: RNNs LSTMs GRUs

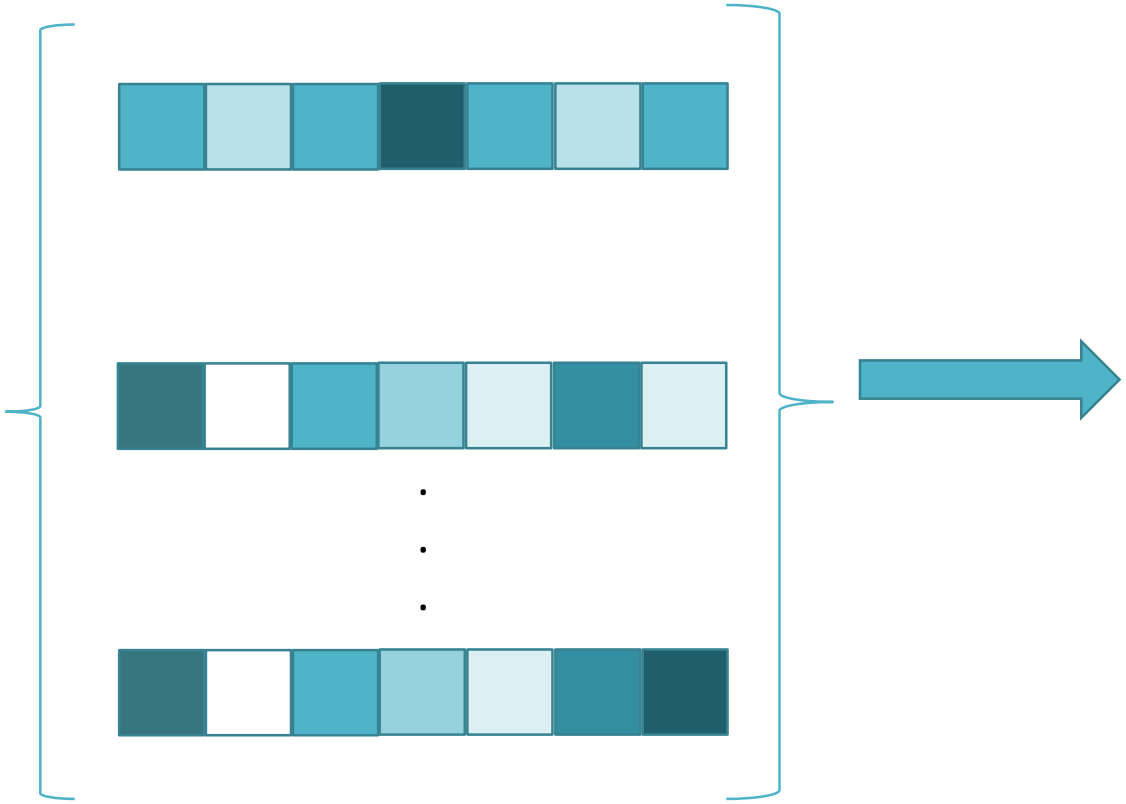
The different gates of a LSTM help to solve for the vanishing gradient problem by deciding which inputs to keep

Modelling approach

EMBED ACTIONS



Modelling approach



MODEL INPUTS

1. Fixed length RNN
2. Multistep prediction RNN
3. Multiclass classification using a NN

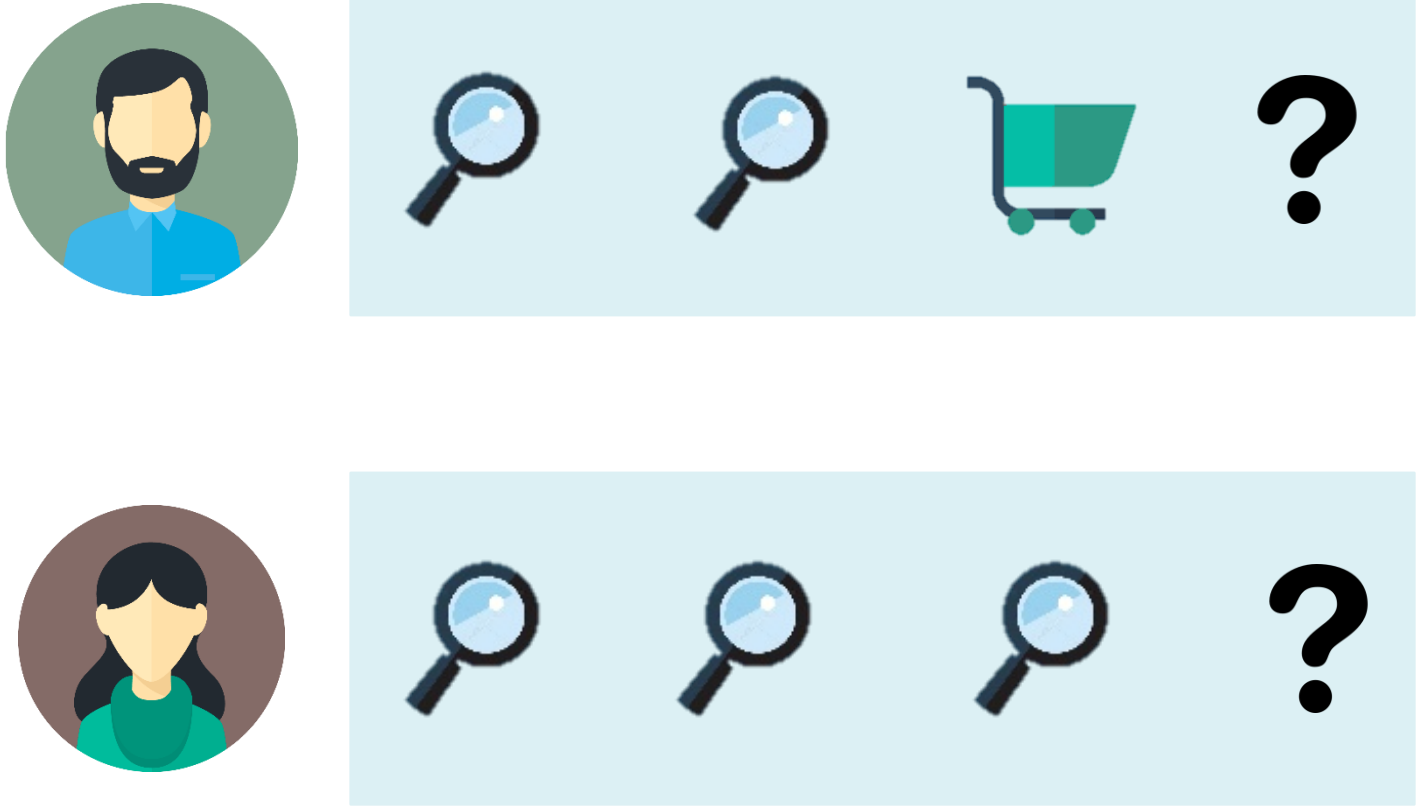
MODEL ARCHITECTURES

Sequential Modelling

Use the first n actions to predict the n+1 action:

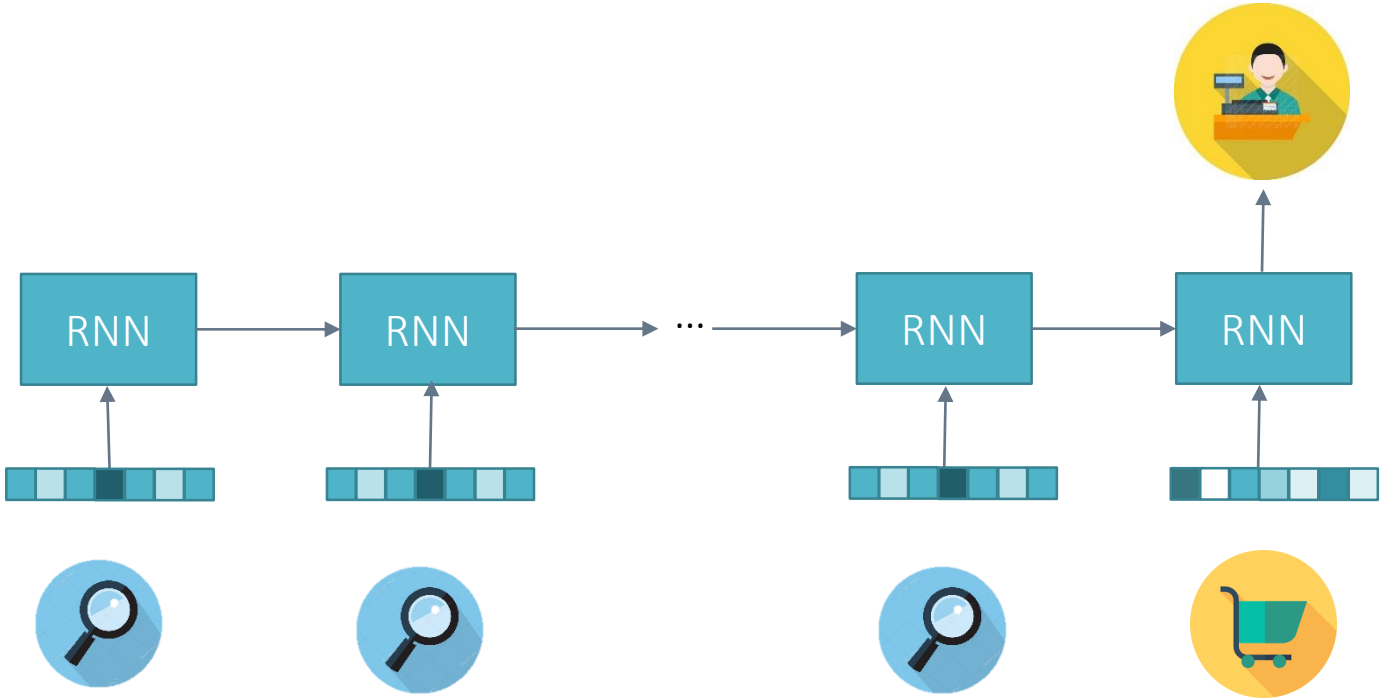
- View
- Add to cart
- Checkout

Take into account the order of the first n actions



Fixed length RNN

Input into an RNN is fed *sequentially*



Basic RNN architecture

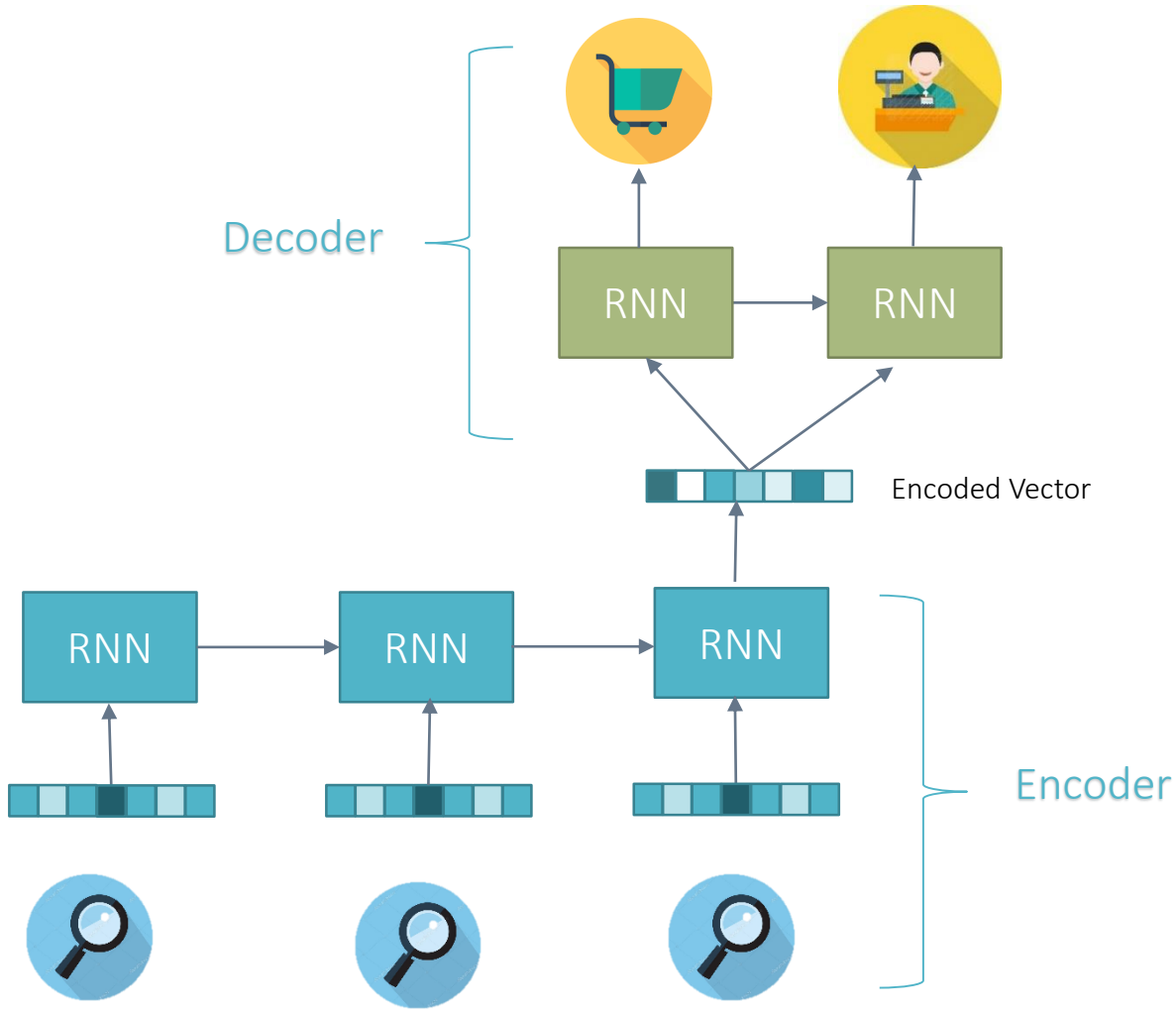
Multistep prediction

Predict the next m actions

- View
- Add to cart
- Checkout

Sequence to Sequence

To predict the next m actions, we use an encoder-decoder architecture, which uses two RNNs



Encoder Decoder Architecture

NN multiclass classification

Use the first n actions to predict the n+1 action:

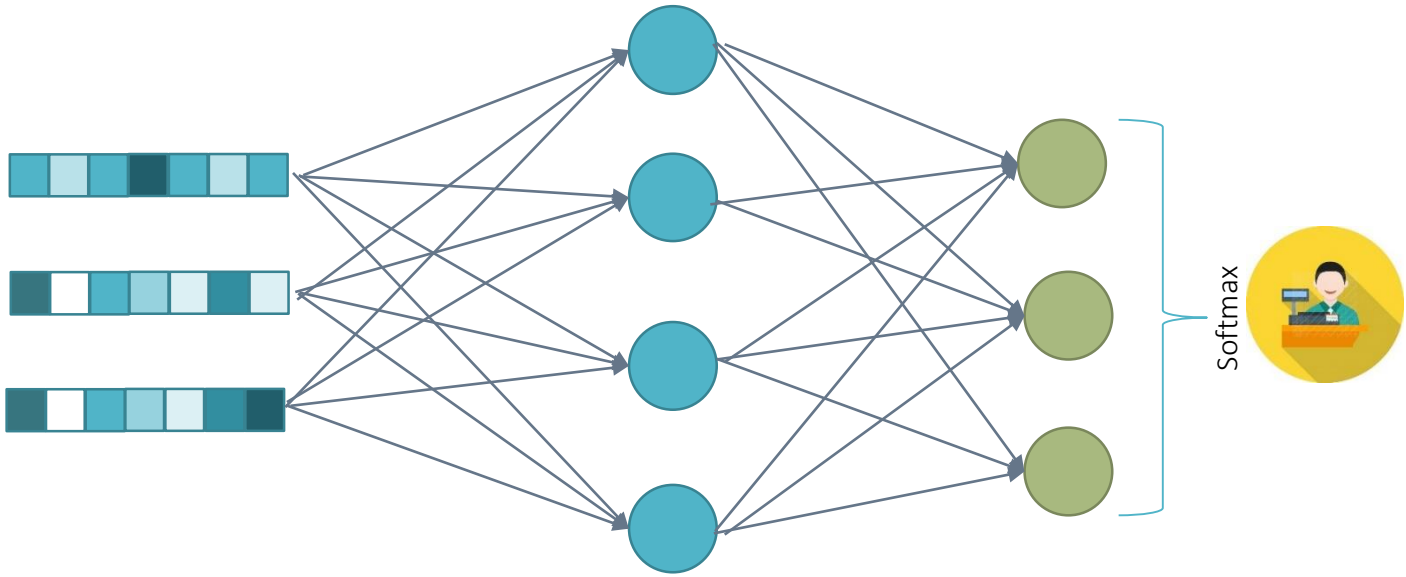
- View
- Add to cart
- Checkout

Order of actions do not matter

Continuous bag of words

Continuous bag of words architecture uses the previous set of words to predict the next word.

The relationships between the input words and the next word is used to learn the embeddings.

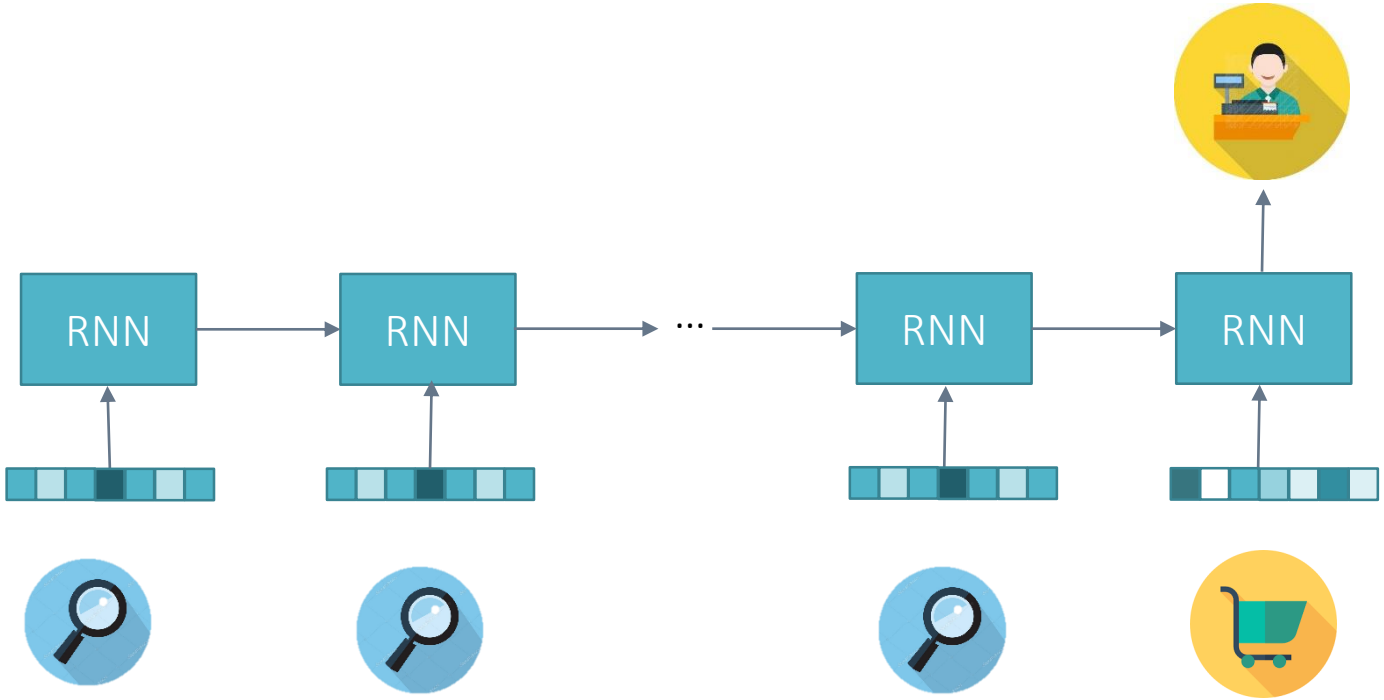


Hidden Layer

Continuous Bag of Words (Mikolov et al., 2013)

Sequential Modelling

In contrast to a NN, input into an RNN is fed *sequentially*



Basic RNN architecture

Anatomy of a Keras model: RNN

THE CODE

```
def rnn_model(input_length, num_units, num_actions):
  # sequence of actions
  A = Input(shape=(input_length,), name="actions") #length of sequence

  #create embedding for actions
  a = Embedding(input_dim=num_actions, output_dim=num_units, name='action_embedding')(A)

  #add LSTM Layer
  L = LSTM(num_units,return_sequences=False)(a)

  #add regular NN layer
  predictions = Dense(num_actions, activation='softmax')(L)

  return Model(inputs=A, outputs=predictions, name='RNN_model')
```

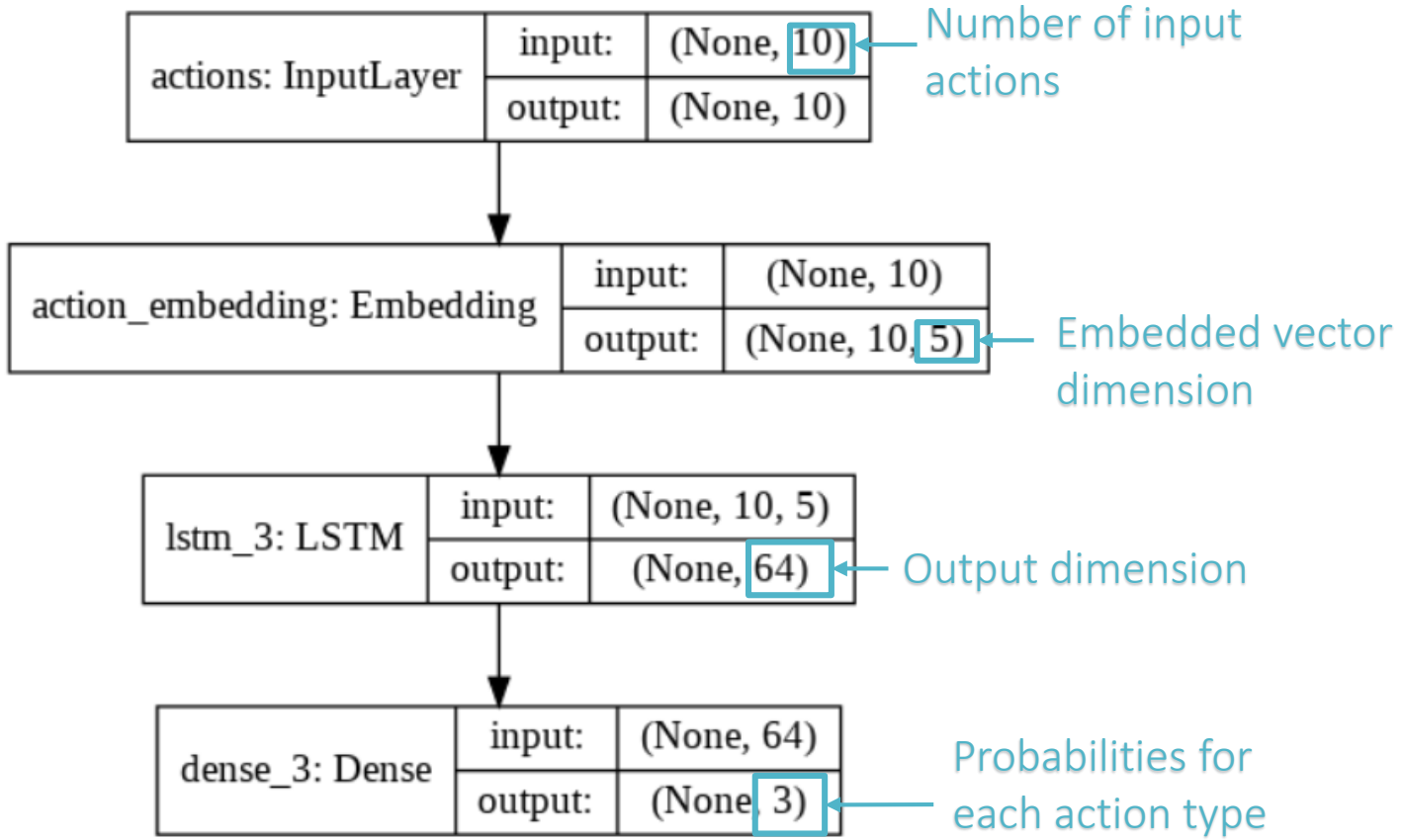
Anatomy of a Keras model: RNN

THE CODE

```
def rnn_model(input_length, num_units, num_actions):
  # sequence of actions
  A = Input(shape=(input_length,), name="actions") #length of sequence

  #create embedding for actions
  a = Embedding(input_dim=num_actions, output_dim=num_units, name='action_embedding')(A)

  #add LSTM Layer
  L = LSTM(num_units,return_sequences=False)(a)

  #add regular NN layer
  predictions = Dense(num_actions, activation='softmax')(L)

  return Model(inputs=A, outputs=predictions, name='RNN_model')
```

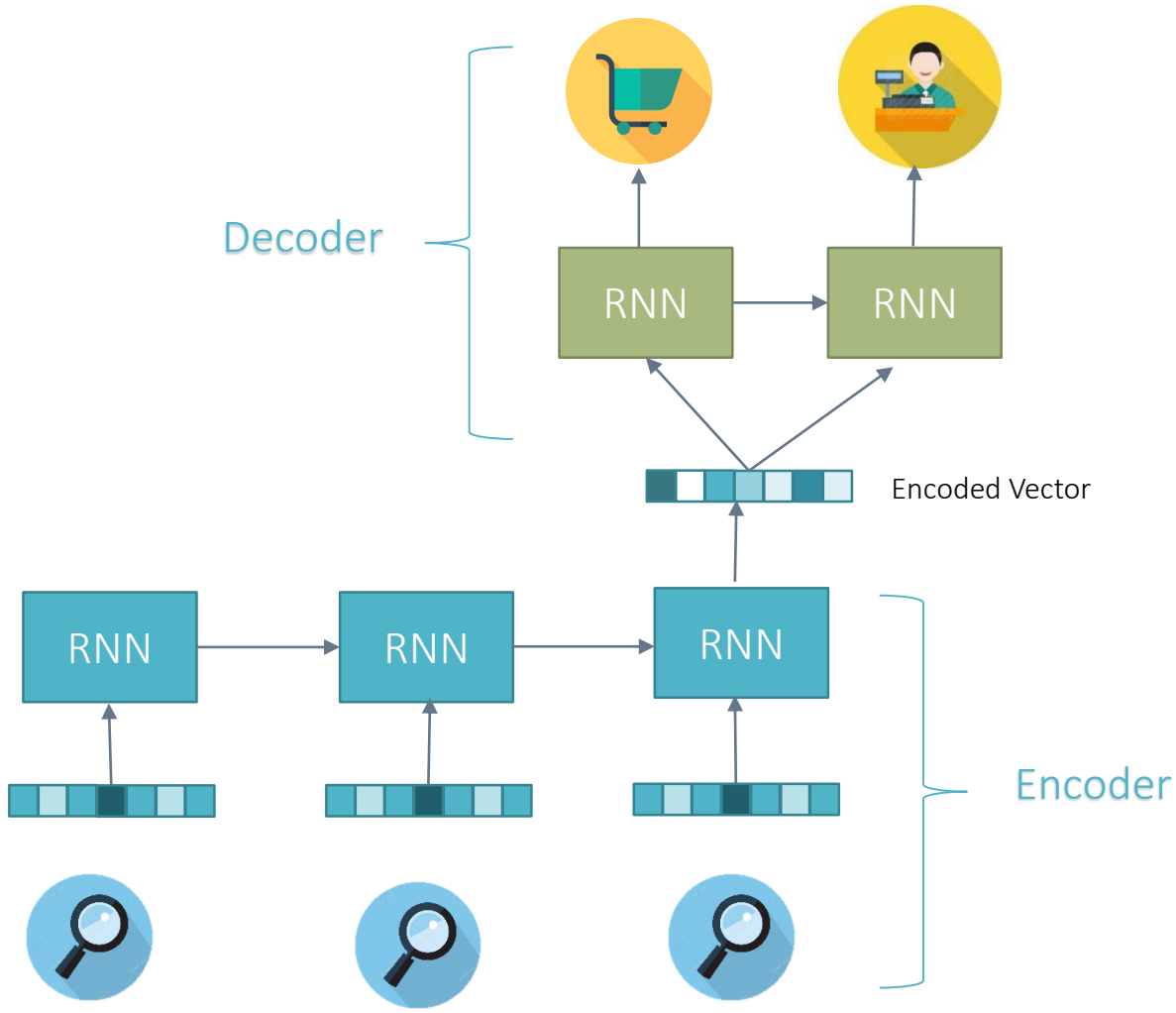
Anatomy of a Keras model: RNN

THE LAYERS



Sequence to Sequence

To predict the next m actions, we use an encoder-decoder architecture, which uses two RNN models



Encoder Decoder Architecture

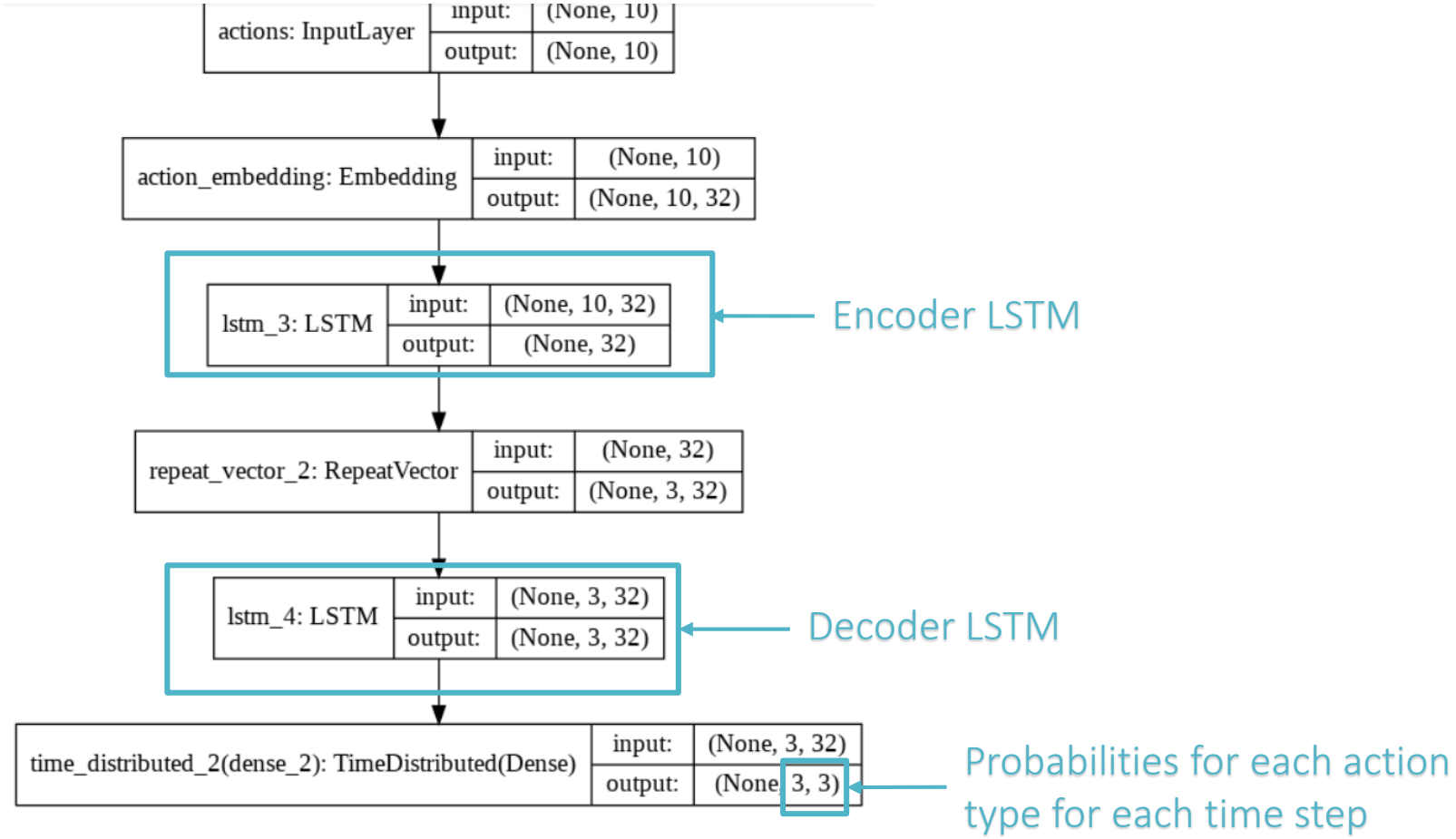
Anatomy of a Keras model: Seq2Seq

THE CODE

```
def multistep_rnn_model(input_length, forecast_length, num_units, num_actions):
  # sequence of actions
  A = Input(shape=(input_length,), name="actions") #length of sequence

  #create embedding for actions
  a = Embedding(input_dim=num_actions, output_dim=num_units, input_length=input_length, name='action_embedding')(A)

  # return_sequences = false to only produce the last output
  encoder = LSTM(num_units,return_sequences=False)(a)

  # repeat the output of the encoder for each output of the target
  temp = RepeatVector(forecast_length)(encoder)

  decoder = LSTM(num_units, return_sequences=True)(temp)

  # Apply a NN for each timestep output
  predictions = TimeDistributed(Dense(num_actions, activation='softmax'))(decoder)

  return Model(inputs=A, outputs=predictions, name="Multi-step Prediction")
```

Anatomy of a Keras model: Seq2Seq

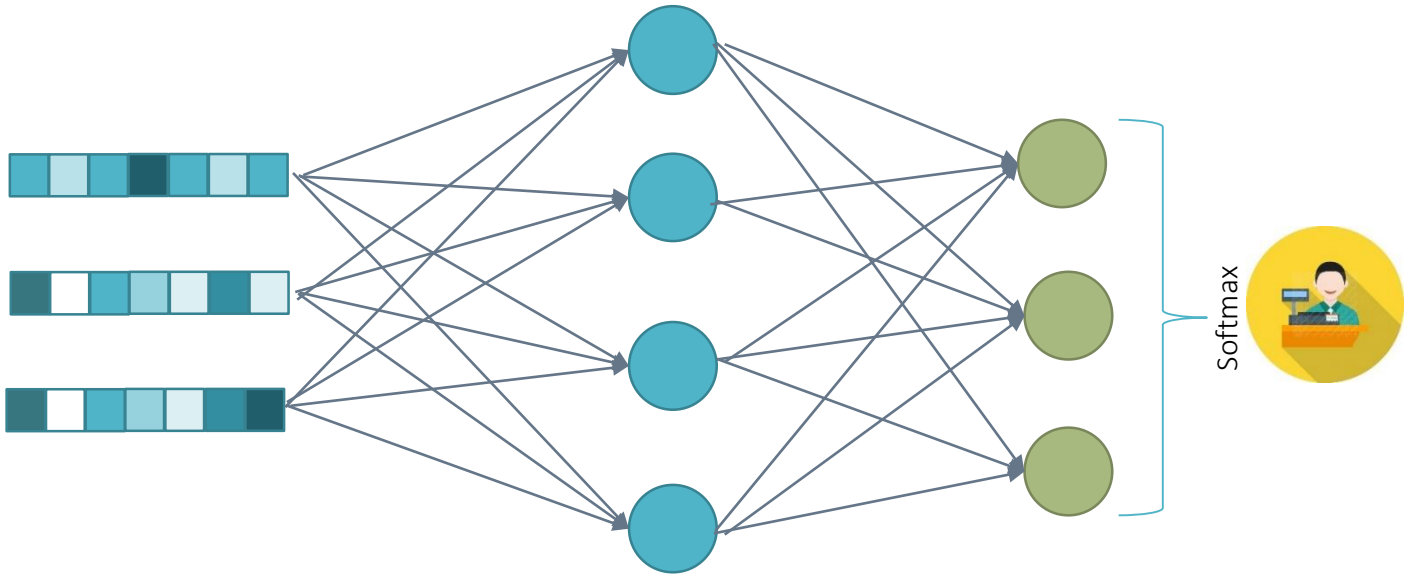
THE LAYERS



Continuous bag of words

Continuous bag of words architecture uses the previous set of words to predict the next word.

The relationships between the input words and the next word is used to learn the embeddings.



Hidden Layer

Continuous Bag of Words (Mikolov et al., 2013)

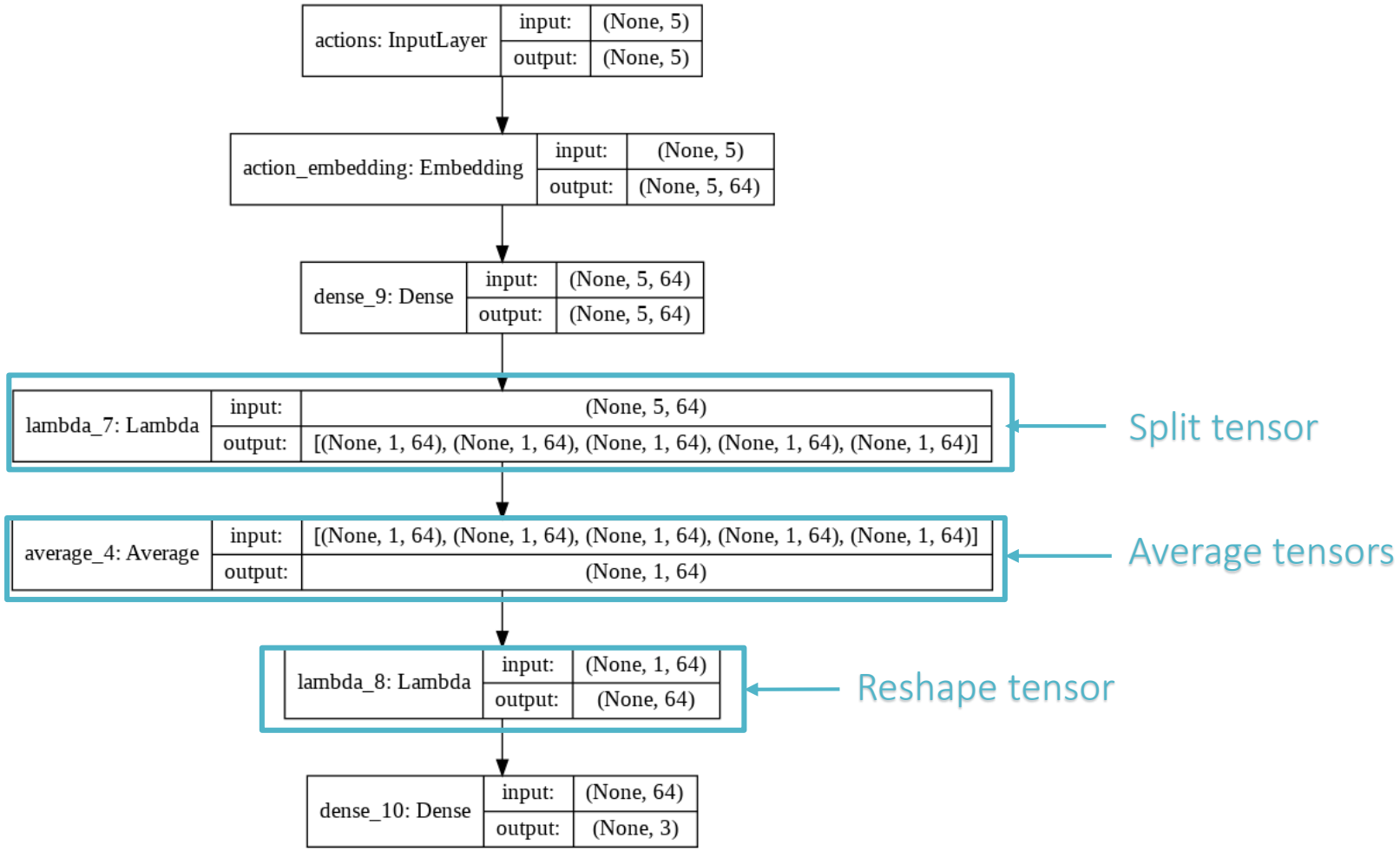
Anatomy of a Keras model: NN

THE CODE

```
def nn_model(input_length, num_units, num_actions):

  # Input sequence of actions
  A = Input(shape=(input_length,), name="actions") #length of sequence, size of vocab

  #create embedding for actions
  x = Embedding(input_dim=num_actions, output_dim=num_units, input_length=input_length, name='action_embedding')(A)

  # create the hidden layers
  x = Dense(num_units, activation='relu')(x)

  # split the tensors for each time step
  split = Lambda(my_split(input_length-1))(x)

  averaged = Average()(split)
  squeezed = Lambda(squeeze(axis=1))(averaged)

  predictions = Dense(num_actions, activation='softmax')(squeezed)

  return Model(inputs=A, outputs=predictions, name='Plain NN')
```

Anatomy of a Keras model: NN

THE LAYERS

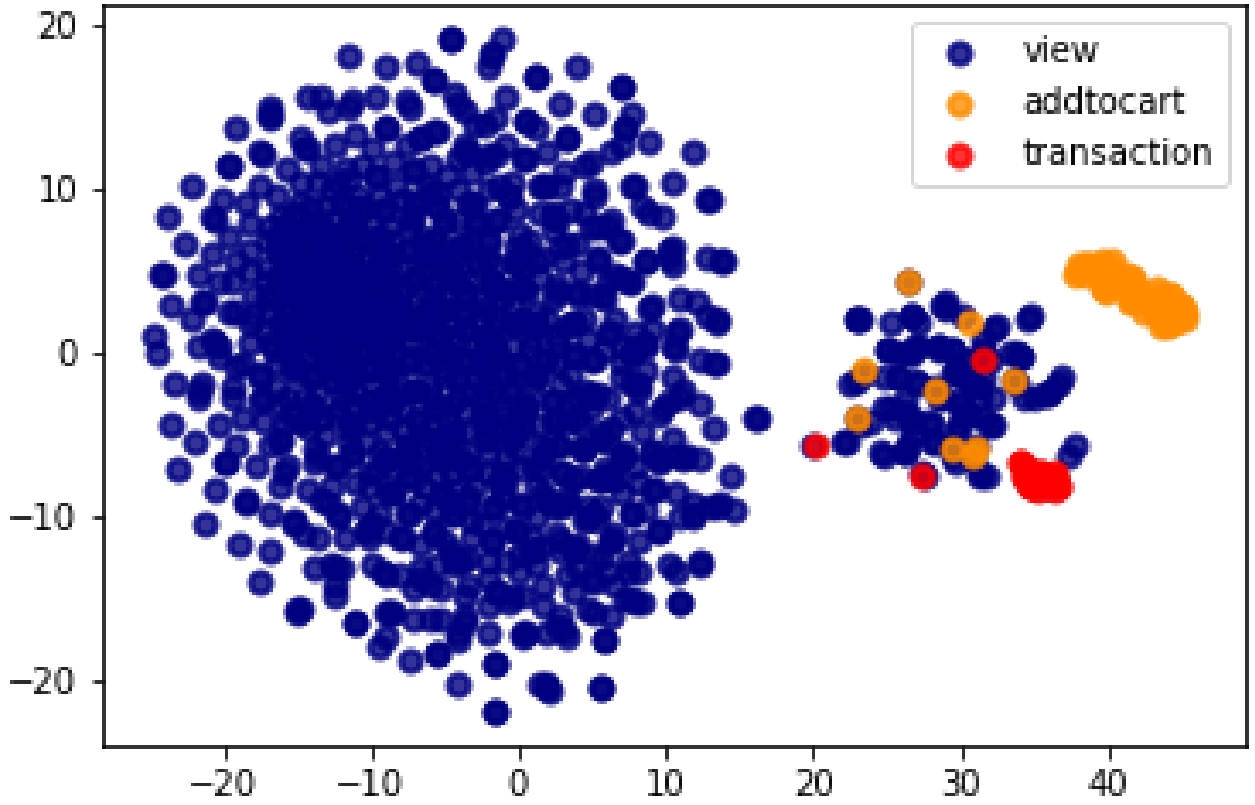


Model performance

Hidden Dimensions	Multi-class NN	Fixed Length RNN	Multistep Prediction
32	0.48	0.62	0.38
64	0.51	0.60	0.45
128	0.51	0.55	0.38

Weighted accuracy trained using n=10 timesteps

Embeddings of users



Key takeaways

1. Many possible architectures to solve the problem
2. Make sure layers are compatible
3. Higher dimensions requires more data

Resources

Jupyter Notebook with Code - bit.ly/2Q4IrR8

Adventures in Machine Learning: Keras LSTM Tutorial - bit.ly/2X3I76F

Keras Functional API for Deep Learning - bit.ly/2CASvJy

GitHub for Doc2Vec Implementation - bit.ly/2Q8sf1h

Kaggle Ecommerce Dataset - bit.ly/2rsD2ca

RNNs LSTMs GRUs - bit.ly/36UlFkJ

Questions