P0: Introduction to Keras

Natural Language Understanding

Interuniversity Master’s Degree in Artificial Intelligence
Academic Year 2023-2024

UNIVERSIDADE DA CORUÑA

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

UniversidadeVigo

v1

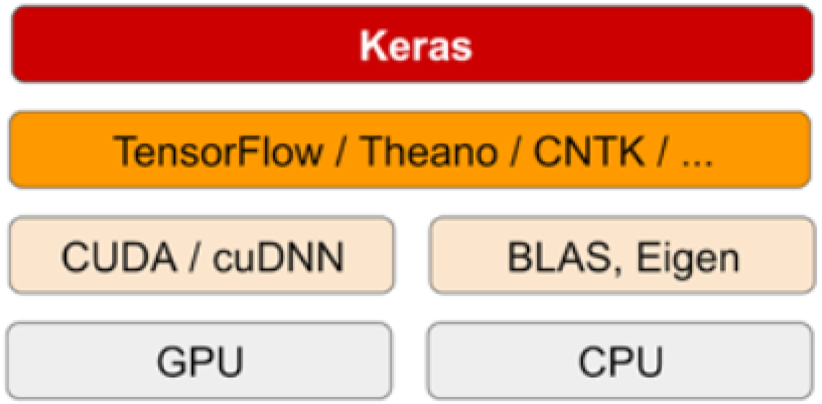
Objectives P0

- Introduce Keras and programming environments for Python notebooks
- Get acquainted with basic libraries useful for NLU (preprocessing, tokenization, embeddings, etc.)
- Be able to train and use neural network models for classification problems in a text domain

Python and notebooks

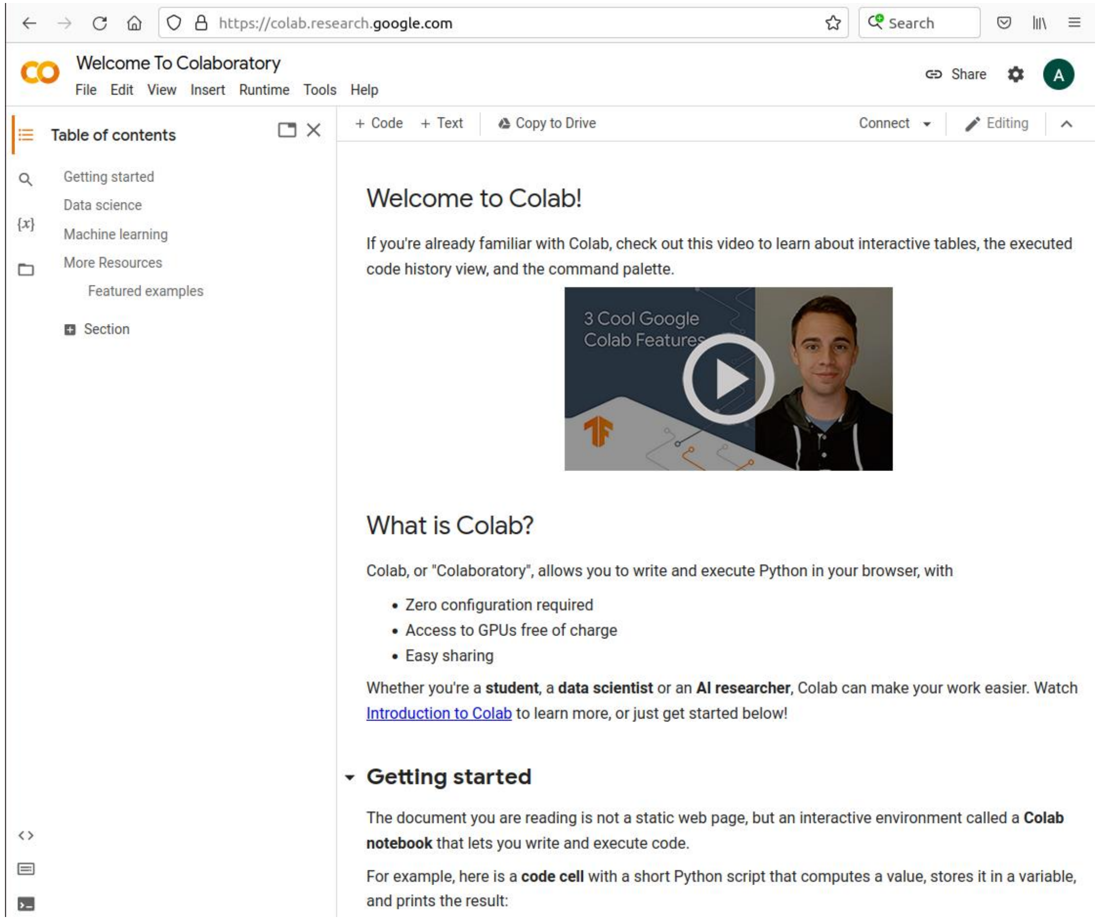
- Jupyter notebooks
  - https://jupyter.org
  - Great to run deep learning experiments
  - Combine interactive and incremental running and documenting (text-editing) into a single Web GUI
  - Requires installing libraries in your local computer
- Google Colaboratory
  - https://colab.research.google.com/
  - Looks like an online notebook, you can upload your own jupyter notebooks
  - No configuration and installation needed (Keras available already)
  - Access to GPU/TPU free of charge (if really needed)

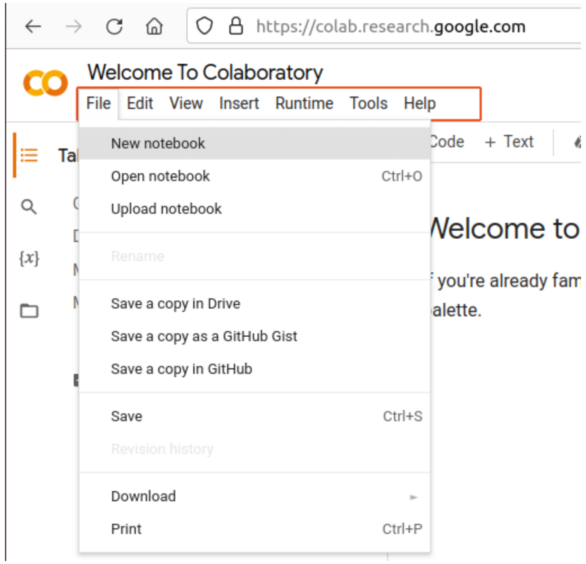
KERAS



- The Python Deep Learning API
- URL: https://keras.io/
- Model-level library with high-level building blocks for developing deep-learning models
  - user-friendly API
  - same code to run seamlessly on CPU or GPU
  - built-in support for convolutional networks, recurrent networks (for sequence processing)...

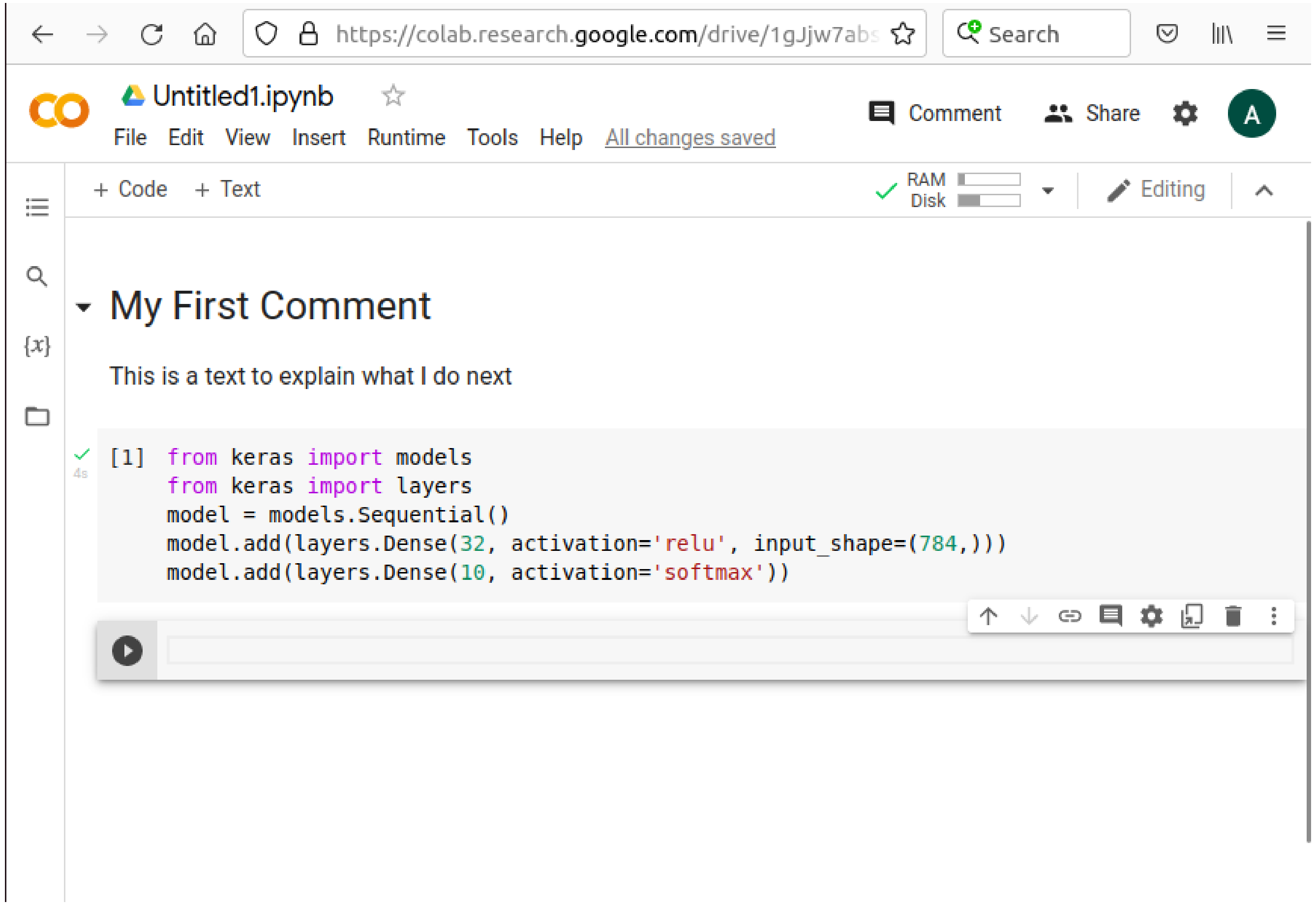
Colab





https://colab.research.google.com/

Colab environment



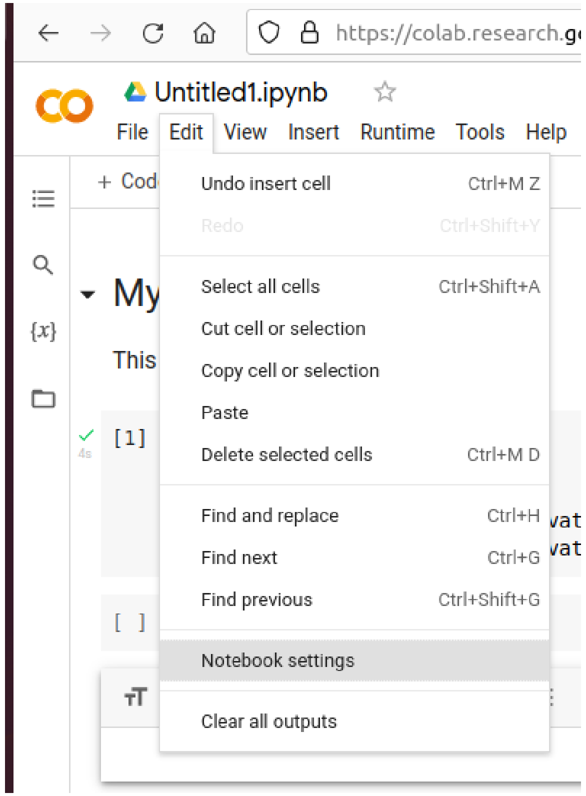
Text cells

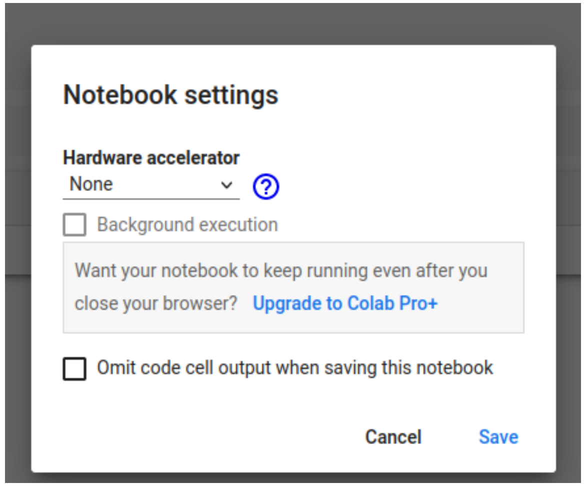
- Formatting and style in Markdown

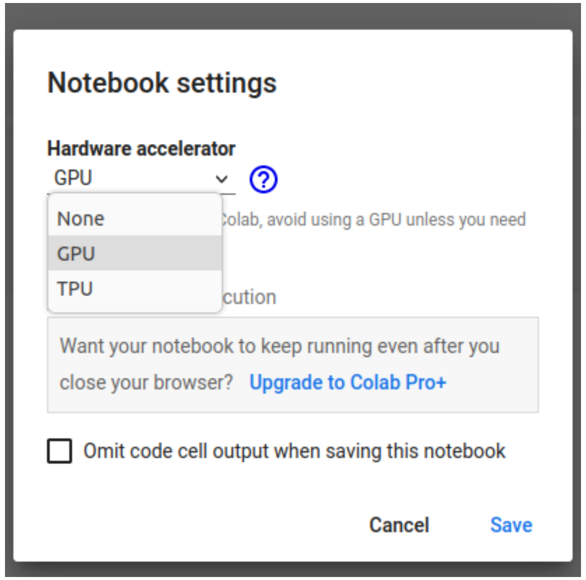
Code cells

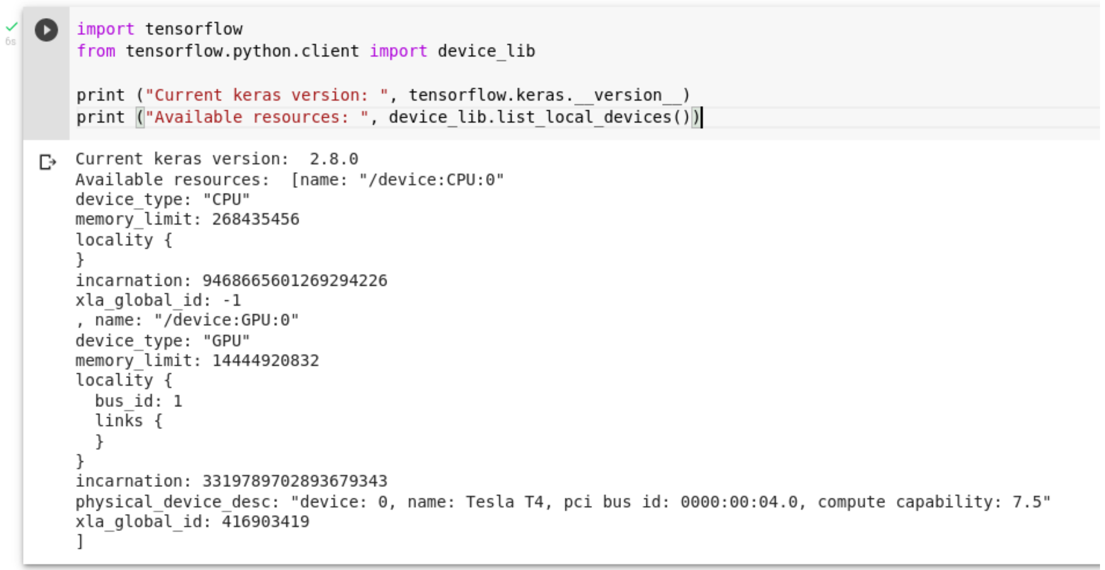
- control on order to run cells interactively
- GPU/TPU may be available

Colab GPU/TPU use *







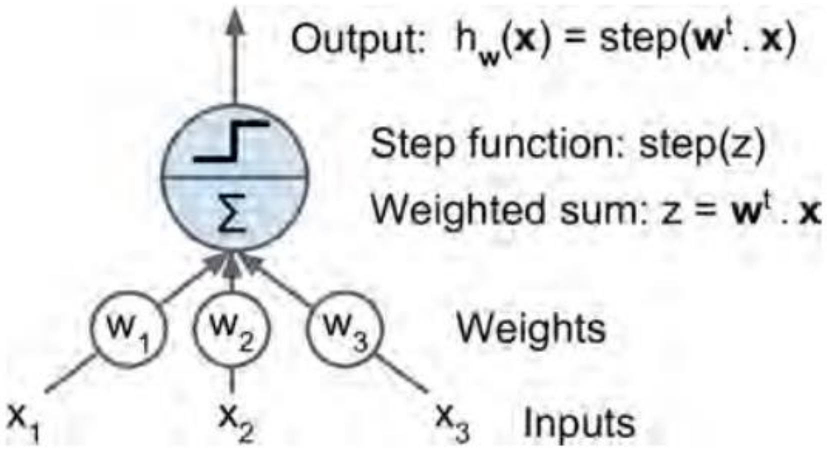


*enable it only if really necessary

Part I

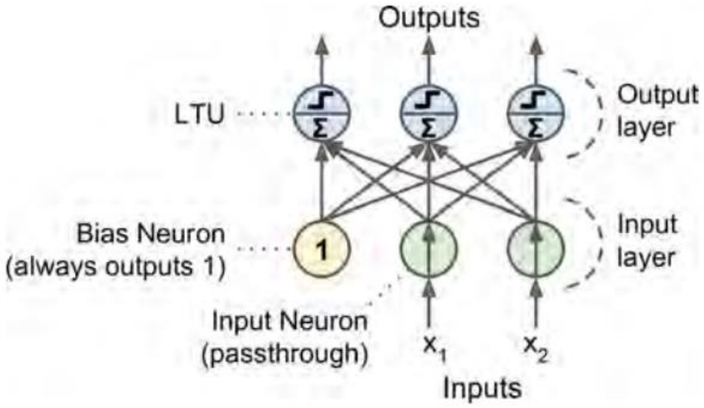
Formative assignment (P0_part1.ipynb)

The Perceptron and Artificial Neural Networks

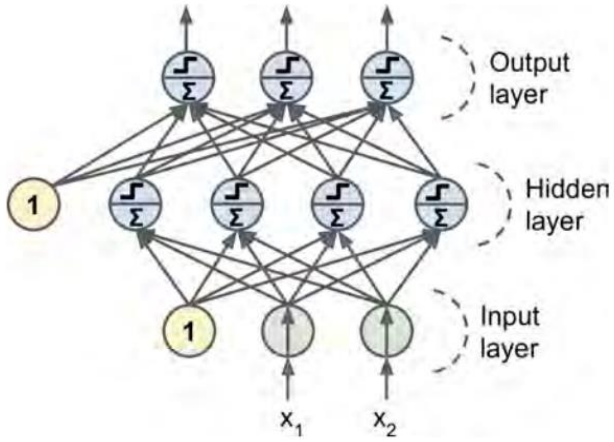


$(z = w_1\, x_1 + w_2\, x_2 + \cdots + w_n\, x_n = \mathbf{w}^T \cdot \mathbf{x})$

Liner Threshold unit



Perceptron



Multi-layer Perceptron

Workflow overview

Define data

- Define your training data: input tensors and target tensors

Define layers

- Define a network of layers (or model)

Configure

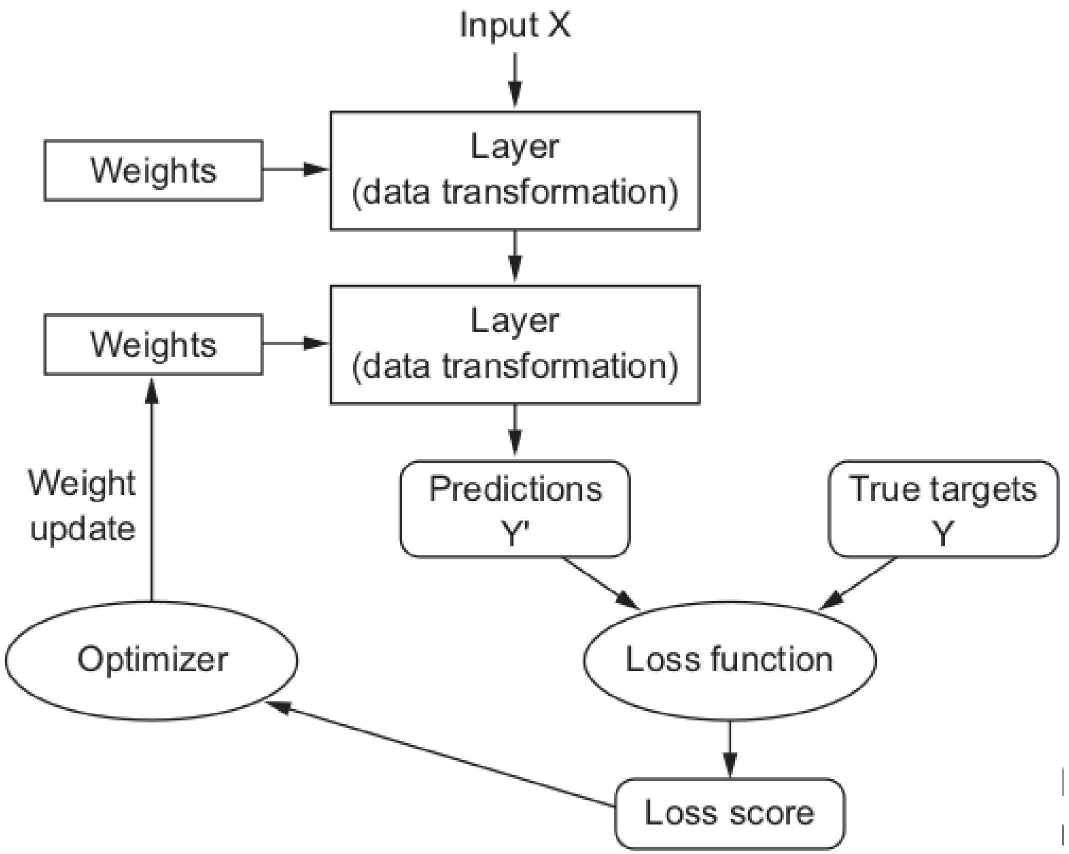
- Configure the learning process by choosing a loss functions, an optimizer and some metrics to monitor and guide the learning process

Train

- Train your model using the fit method

Evaluate

- Evaluate your model with the test data



Inspecting the provided Keras code

Three Examples

- Example 1: Prediction of patients with diabetes
- Example 2: Digit recognition with the MNIST dataset
- Example 3: Classifying sentiment of movie reviews

Your goal

- To inspect and run the code, looking at how the workflow is adapted to different problems in Keras
- Discuss with your partner key aspects about the code and outcomes
- Try out small changes to the provided code

Example 3: Classifying movie reviews

- Internet Movie Database (IMDB)
- Dataset: https://ai.stanford.edu/~amaas/data/sentiment/
  - Included in Keras already, will be downloaded first time you use it
- 50K polarized reviews
  - 25K for training, 25K for testing, each with 50-50% negative/positive review labels
- The goal is to learn to classify whether a review is positive or negative based on the text
  - Binary classification problem

Define Data

- train_data and test_data are lists of reviews
- each review is a list of word indices (encoding a sequence of words)
- train_labels and test_labels are lists of 0s and 1s, where 0 stands for negative and 1 stands for positive

```
from keras.datasets import imdb

# only keep the top num_words most frequent. Discard rare words.
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)


word_index = imdb.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in train_data[0]])


print(decoded_review)
```

- word_index maps words to integers

Define Data

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
  results = np.zeros((len(sequences), dimension))
  for i, sequence in enumerate(sequences):
    results[i, sequence] = 1.
  return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)

y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

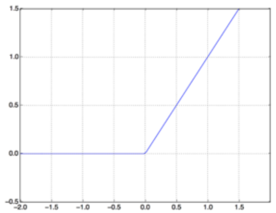
- Numpy has a bunch of useful operations to deal with vectors, arrays, tensors
- Remember that NNs don´t work with words but with numbers as input
- Vectorize to keep length manageable and get a specific data representation
  - Kind of one-hot encoding with 0s and 1s whether terms/tokens are or not present in a given review
- Also vectorize labels

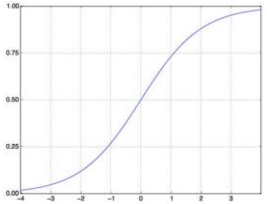
Define Layers

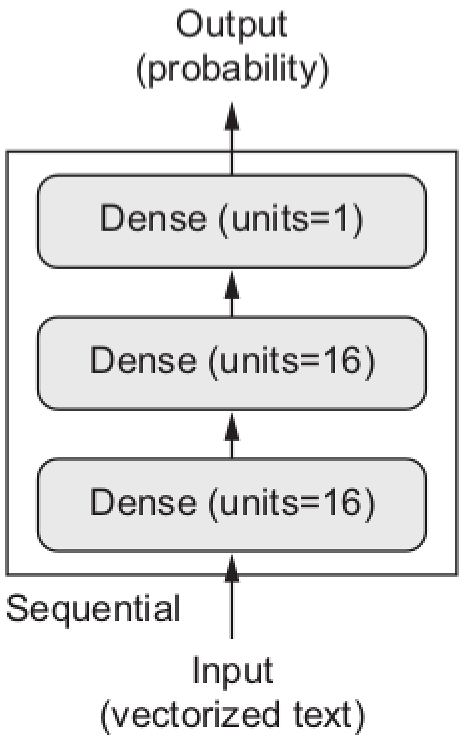
```
from keras import models
from keras import layers


model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

model.summary()
```

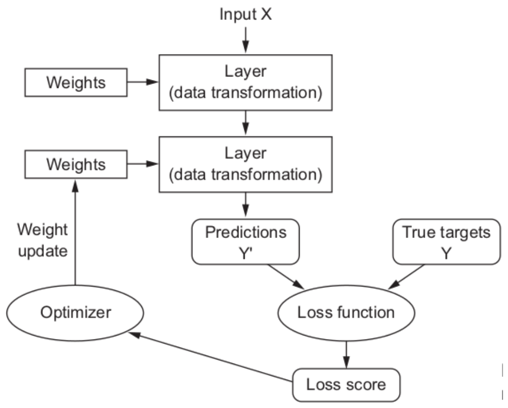






- Input layer is set to 10000 tokens in agreement with the vectorization done before
- Dense layers, fully connected
- Output layer with a neuron with a sigmoid function

Configure




```
model.compile(optimizer='rmsprop', loss='binary_crossentropy',metrics=['accuracy'])
```

```
from keras import optimizers
model.compile(optimizer=optimizers.rmsprop_v2.RMSprop(learning_rate=0.001),loss='binary_crossentropy
', metrics=['accuracy'])
```

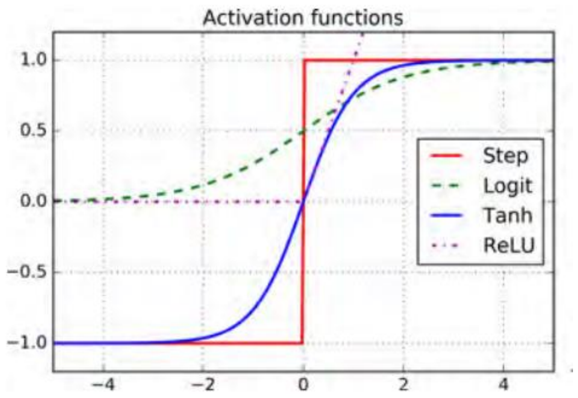
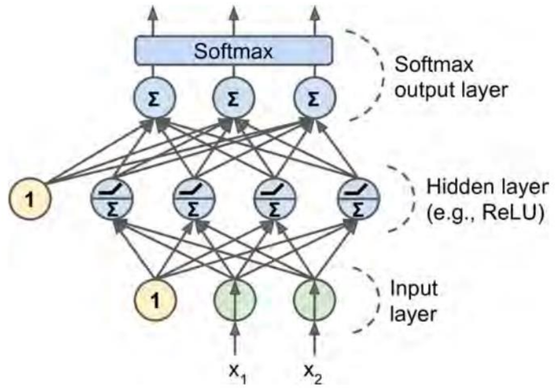
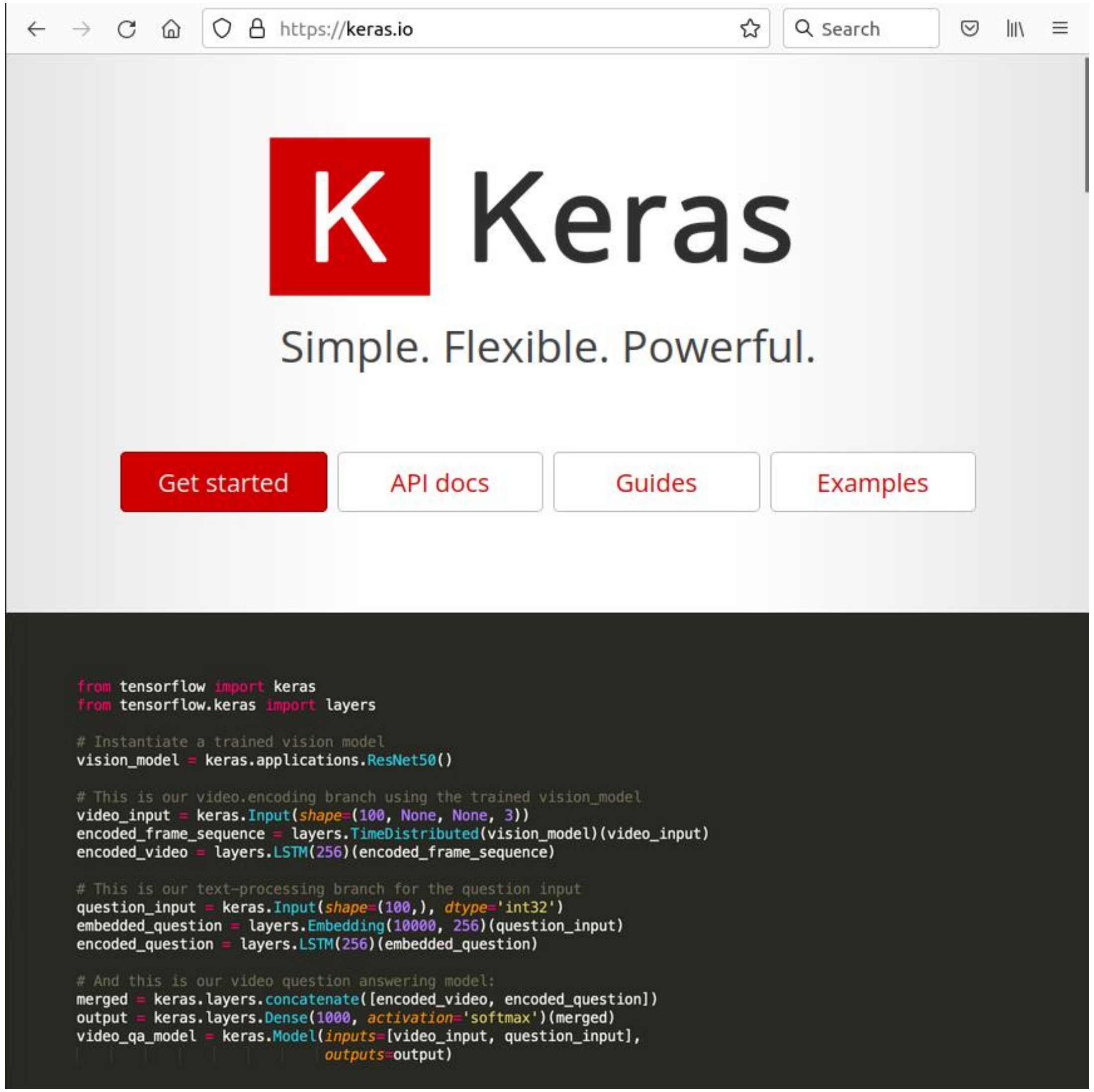


Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	None	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`



Resources


https://keras.io
Search
Keras
Simple. Flexible. Powerful.
Get started
API docs
Guides
Examples
from tensorflow import keras
from tensorflow.keras import layers
# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()
# This is our video.encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)
# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)
# And this is our video question answering model:
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model(inputs=[video_input, question_input],
outputs=output)

Train

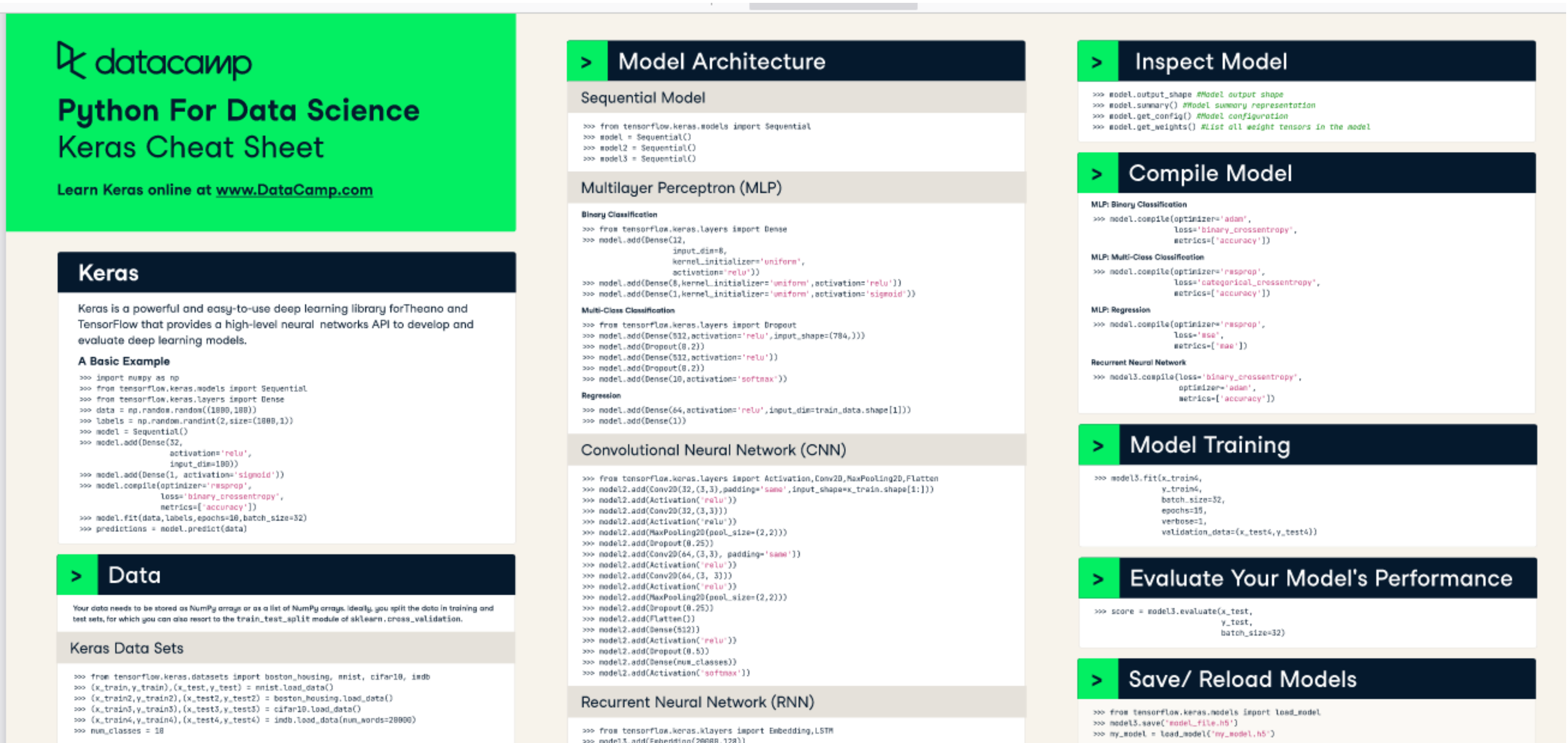
```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,partial_y_train,epochs=20,
batch_size=512, validation_data=(x_val, y_val))
```

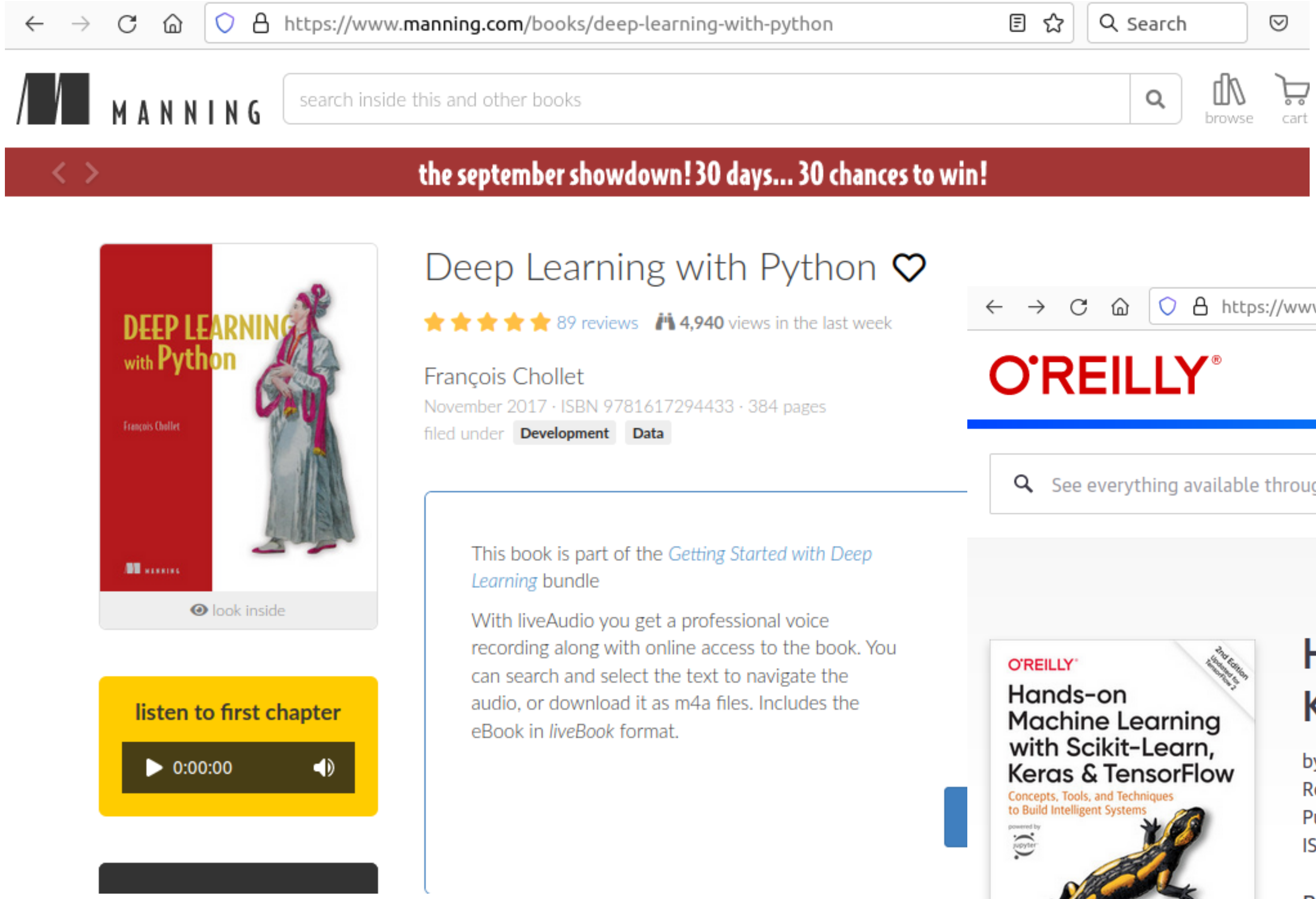
- We can set a subset for validation as in the example or using the
- With .fit() train for 20 epochs in minibatches of 512 samples
- model.fit() returns a History object.
    - Try history_dict = history.history,... for plotting training/validation loss/accuracy history

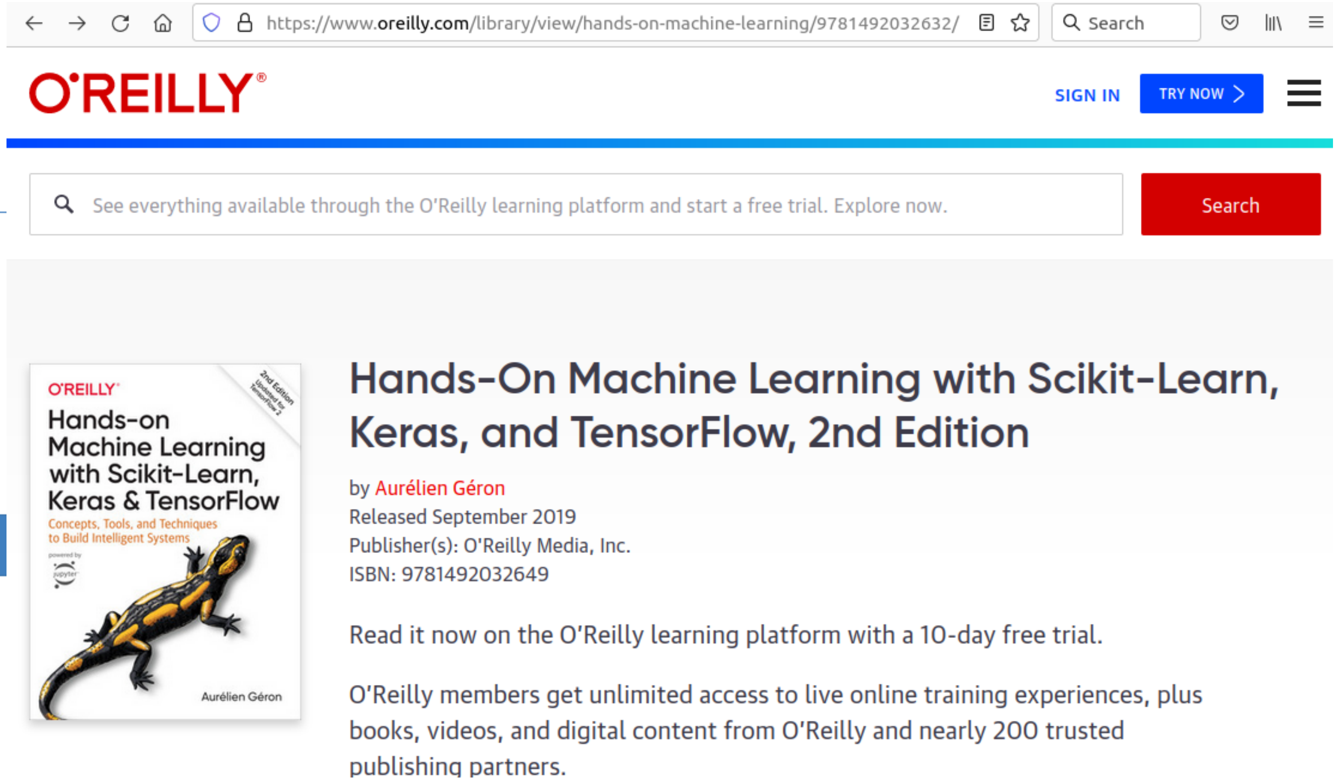
Resources

- Cheatsheet Keras: https://www.datacamp.com/cheat-sheet/keras-cheat-sheet-neural-networks-in-python



Resources


https://www.manning.com/books/deep-learning-with-python
Search
MANNING
search inside this and other books
browse
cart
the september showdown! 30 days... 30 chances to win!
DEEP LEARNING with Python
François Chollet
look inside
listen to first chapter
0:00:00
Deep Learning with Python
89 reviews
4,940 views in the last week
François Chollet
November 2017 · ISBN 9781617294433 · 384 pages
filed under Development Data
This book is part of the Getting Started with Deep Learning bundle
With liveAudio you get a professional voice recording along with online access to the book. You can search and select the text to navigate the audio, or download it as m4a files. Includes the eBook in liveBook format.


https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
Search
O'REILLY
SIGN IN
TRY NOW
See everything available through the O'Reilly learning platform and start a free trial. Explore now.
Search
O'REILLY
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
Aurélien Géron
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition
by Aurélien Géron
Released September 2019
Publisher(s): O'Reilly Media, Inc.
ISBN: 9781492032649
Read it now on the O'Reilly learning platform with a 10-day free trial.
O'Reilly members get unlimited access to live online training experiences, plus books, videos, and digital content from O'Reilly and nearly 200 trusted publishing partners.

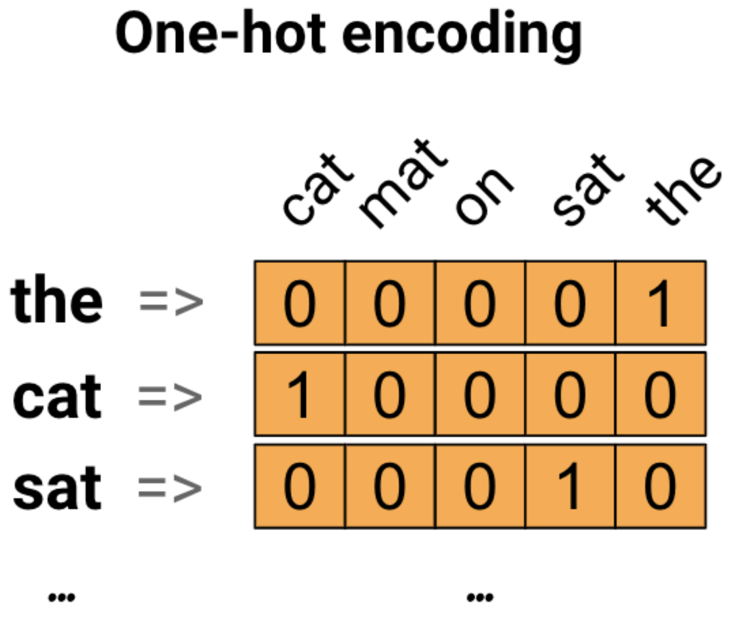
Working with text input

- Texts are composed of documents, paragraphs, words, characters,…
  - Sample review: "A thrilling movie. Absolutely recommended for those loving action..."

  - Tokenization: how to get from raw text to words (or tokens)
  - Words as tokens?
    - What about US, USA, United States of America? Mr. Jones vs. Indiana Jones? Absolutely vs. Absolutely vs. Absolute? Pictured/picturing vs. picture vs. pictur?

- Regardless how we tokenize, input must be numbers to feed NNs
  - We need Encodings and Embeddings

One-hot Encoding



- Raw text = "The cat sat on the mat"
- Word-level Tokens = "cat", "mat", "on", "sat", "the"

- Length of vocabulary = 5
- One-hot vector for "cat" token = [1, 0, 0, 0, 0]

- A sentence can be encoded using one-hot vectors: [1,1,1,1,1] *
- or a sequence of integer indexes: [5, 1, 4, 2, 3]

- What would it happen if...
  - switched the vector for "cat" and "the"?
  - there were several documents?
  - there were many documents leading to (a realistic situation with) thousands tokens?
    - Can we handle unlimited number of tokens?
    - Advantages and disadvantages of one-hot binary encoding vs. Sequences?
    - Can we keep the order of the word tokens in our chosen representation or are they like bag of words?

* toy unrealistic example to raise discussion

Word Embeddings

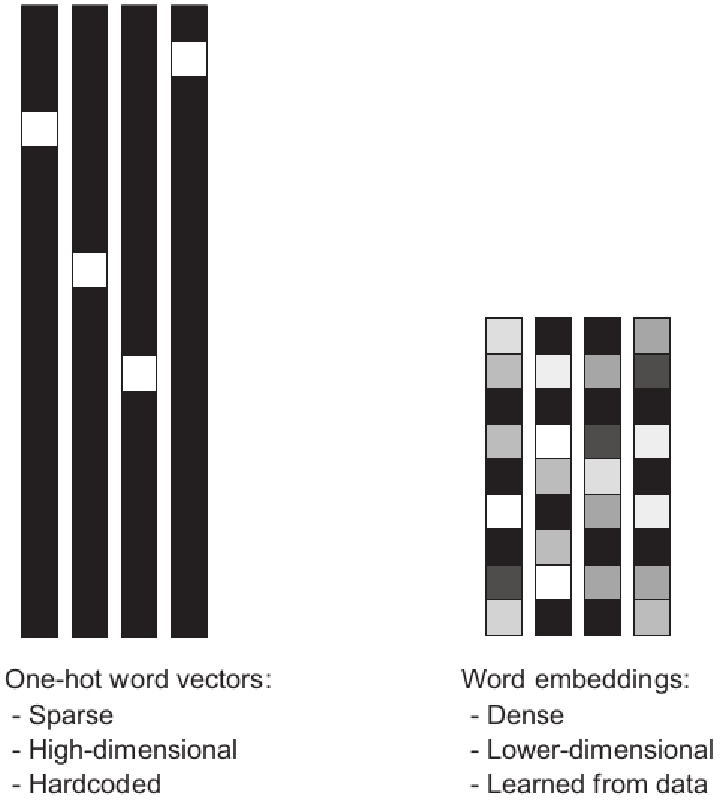
A 4-dimensional embedding

cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4
...		...		

- Lower-dimension and dense representation
- Vector values are learned from data (so not intended for humans)
- But enables meaningful vector arithmetic operations in properly trained embeddings (e.g., what is the "king – man + woman? => queen)

- What would it happen if...
  - there were several documents?
  - there were many documents leading to thousands tokens?
    - Can we handle unlimited number of tokens?
    - Advantages and disadvantages of one-hot encoding vs. Word embedding?
    - Can we keep the order of the word tokens in our chosen representation?

One-hot Encoding vs. Word Embeddings



- Find out more about these practical issues:
  - https://www.tensorflow.org/text/guide/word_embeddings

Part II

Summative assignment (P0_student_assignment_part2.ipynb)

P0 Final Assignment