

Applied Data Science Project

L11 - Project development tools I

Giuseppe Rizzo

Turin, October 7, 2021

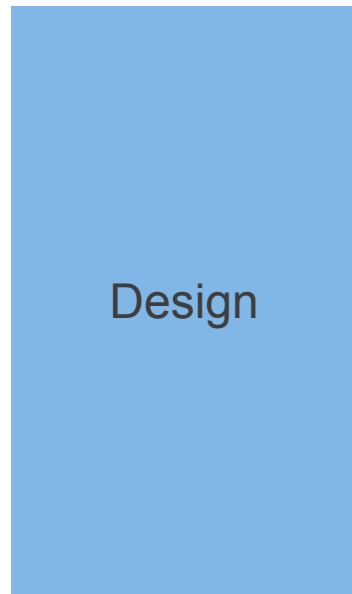


**Politecnico
di Torino**

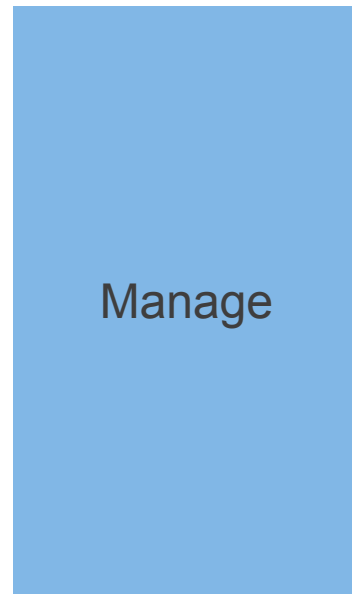


e l i s
European Laboratory for Learning and Intelligent Systems

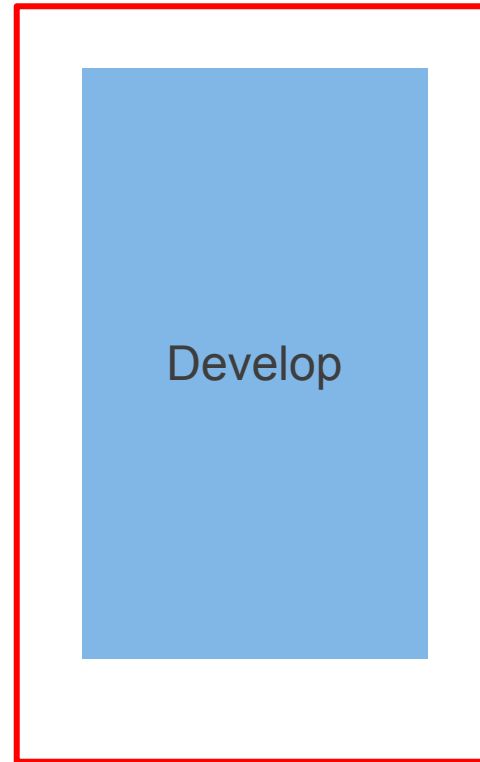
Pillars



Design



Manage



Develop



Communicate



Development

It is where the magic happens

A machine intelligence is created to generate the outputs that meet objectives and requirements with the involvement of a team tasked on activities with due dates

Objectives and requirements have been defined in the Design pillar

Activities and due dates in the Manage pillar

Knowledge tools

- agile and scrum
- collaborative workspaces
 - program development
 - repository
 - communication among project developers

Knowledge tools

- agile and scrum

Lecture 8 & 10

- collaborative workspaces
 - program development
 - repository
 - communication among project developers

Apps

- agile and scrum

- collaborative workspaces
 - program development
 - version control
 - communication among project developers

Colaboratory

Github

Slack

Program development

- “Divide and rule”
- Tasks are mapped into modules
- One module has one lead developer and, eventually, contributors
- Choose the programming language according to:
 - ecosystem of software modules that you can utilize
 - easiness of model integration
 - familiarity. Do not be afraid to switch to another (similar programming language) since they share most of the features and development patterns

Python is considered the default language for developing machine intelligence nowadays



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

An application where to develop, share, and also test on dedicated hardware (GPU to speed up the computing)



Overview

The screenshot shows the Google Colaboratory web interface. On the left is a sidebar with a 'Table of contents' panel. The main area displays a notebook titled 'What is Colaboratory?'. Annotations with arrows point to various UI elements: a red box around the 'Share' button in the top right is labeled 'share with collaborators'; a red box around the '+ Text' button in the top toolbar is labeled 'add a block for writing explanations'; a red box around the '+ Code' button in the top toolbar is labeled 'add a block for code writing'; and a large red box around the main content area is labeled 'Notebook containing both code blocks and explanations in a narrative fashion'.

add a block for writing explanations

Share

share with collaborators

add a block for code writing

Table of contents of the notebook

Notebook containing both code blocks and explanations in a narrative fashion

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Section

What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

Menu bar



Welcome To Colaboratory

File Edit View Insert **Runtime** Tools Help

define the runtime
configurations, among those
the hardware acceleration if
needed

Notebook settings

Hardware accelerator

GPU 


To get the most out of Colab, avoid using a GPU unless you need one. [Learn more](#)

☐ Omit code cell output when saving this notebook

Cancel

Save

Share



Simple Sentiment Analysis.ipynb

File Edit View Insert Runtime Tools Help

Comment

Share

Settings

Share with people and groups

Add people and groups

Giuseppe Rizzo (you)

Owner

[Send feedback to Google](#)

Done

Get link

Restricted Only people added can open with this link

[Change to anyone with the link](#)

[Copy link](#)

Connect your Google Drive

Enable authorization to import data from Google Drive

```
1 from google.colab import drive
2 drive.mount('/content/drive/')

```

List the files in your drive

```
1 !ls "/content/drive/My Drive/"

```

Running with Google Colab

Upload YOUR_PYTHON_FILE.py to Google Drive & Run with Google Colab

```
1 !python3 "/content/drive/My Drive/Colab  
  Notebooks/YOUR_PYTHON_FILE.py"
```

Run with Google Colab to Download YOUR_PYTHON_FILE.py from Google Drive

```
1 from google.colab import files  
2 files.download('/content/drive/My Drive/Colab  
  Notebooks/YOUR_PYTHON_FILE.py')
```

Bash commands

Bash commands are executed with the environment “!”

Download an external file

```
1 !wget http://ai.stanford.edu/~amaas/data/sentiment/aclImdb\_v1.tar.gz -P "/content/drive/My Drive/Colab Notebooks"
```

Clone a repository

```
1 !git clone https://github.com/pytorch/examples.git
```

Colab == virtual environment

The environment can be customized with the addition of python packages

Install

```
1 !pip install torchtext
```

Show a version

```
1 !pip show torchtext
```



Example

Simple Sentiment Analysis.ipynb

Simple Sentiment Analysis

In this series we'll be building a machine learning model to detect sentiment (i.e. detect if a sentence is positive or negative) using PyTorch and TorchText. This will be done on movie reviews, using the [IMDb dataset](#).

In this first notebook, we'll start very simple to understand the general concepts whilst not really caring about good results. Further notebooks will build on this knowledge and we'll actually get good results.

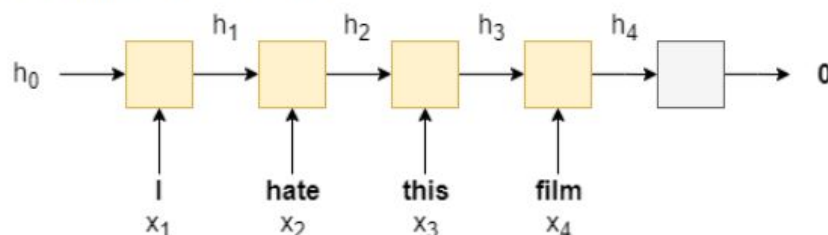
Introduction

We'll be using a **recurrent neural network** (RNN) as they are commonly used in analysing sequences. An RNN takes in sequence of words, $X = \{x_1, \dots, x_T\}$, one at a time, and produces a *hidden state*, h , for each word. We use the RNN *recurrently* by feeding in the current word x_t as well as the hidden state from the previous word, h_{t-1} , to produce the next hidden state, h_t .

$$h_t = \text{RNN}(x_t, h_{t-1})$$

Once we have our final hidden state, h_T , (from feeding in the last word in the sequence, x_T) we feed it through a linear layer, f , (also known as a fully connected layer), to receive our predicted sentiment, $\hat{y} = f(h_T)$.

Below shows an example sentence, with the RNN predicting zero, which indicates a negative sentiment. The RNN is shown in orange and the linear layer shown in silver. Note that we use the same RNN for every word, i.e. it has the same parameters. The initial hidden state, h_0 , is a tensor initialized to all zeros.



Note: some layers and steps have been omitted from the diagram, but these will be explained later.



Thank you for your attention.

Questions?



CONTACTS

Giuseppe Rizzo

Team Leader

p. +39 011 2276244

e. giuseppe.rizzo@linksfoundation.com



FONDAZIONE LINKS

Via Pier Carlo Boggio 61 | 10138 Torino

P. +39 011 22 76 150

LINKSFOUNDATION.COM