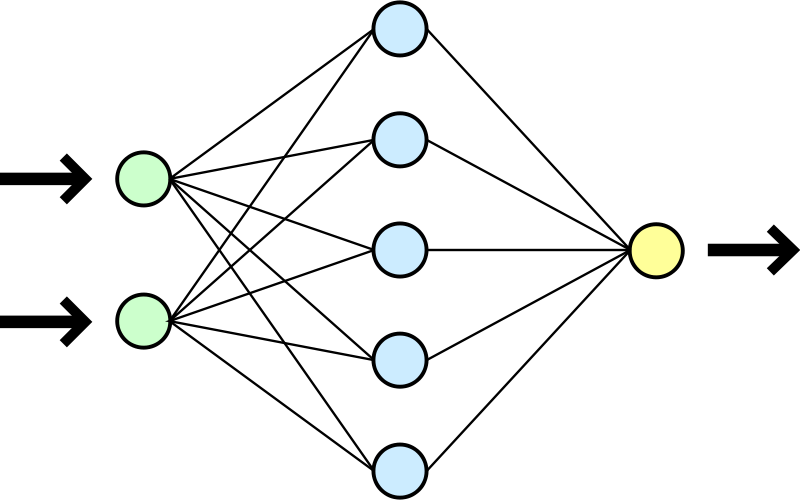
Computer Science Challenges



-CSC 1028-

-2020-

-Oisín Cassidy Dawson-

Contents:

[**Possible challenges:** 3](#_Toc37615756)

[**Chosen challenge:** 3](#_Toc37615757)

[Things to research 3](#_Toc37615758)

[**Python: The basics** 3](#_Toc37615759)

[Commenting 4](#_Toc37615760)

[Declaring variables 4](#_Toc37615761)

[Arrays and collections of data 4](#_Toc37615762)

[Conditional statements 5](#_Toc37615763)

[Loops 5](#_Toc37615764)

[Methods 5](#_Toc37615765)

[User input 6](#_Toc37615766)

[Try catch blocks 6](#_Toc37615767)

[**What is machine learning?** 6](#_Toc37615768)

[What is a neural network? 6](#_Toc37615769)

[What makes up a Neural network? 8](#_Toc37615770)

[What does a neuron do? 8](#_Toc37615771)

[Neuron calculation example 9](#_Toc37615772)

[Complex example 10](#_Toc37615773)

[How does the network learn? 11](#_Toc37615774)

[**Making a neural network** 12](#_Toc37615775)

[The code 13](#_Toc37615776)

[**Python setting up a virtual environment (Python 3.8)** 15](#_Toc37615777)

[Using the virtual environment 16](#_Toc37615778)

[**Setting up TensorFlow (Pyhton 3.7)** 17](#_Toc37615779)

[**Making Poetry** 18](#_Toc37615780)

[Step 1 – Get a Clone of the project 18](#_Toc37615781)

[Step 2 – Setup python in a virtual environment 18](#_Toc37615782)

[Python not recognised error fix 19](#_Toc37615783)

[Step 3 – install all the needed modules 21](#_Toc37615784)

[Step 4 – installing the model 21](#_Toc37615785)

[Step 5 – creating a training dataset 21](#_Toc37615786)

[Step 6 – encoding the dataset for training 22](#_Toc37615787)

[Step 7 – training the model 23](#_Toc37615788)

[**How to find research documents** 25](#_Toc37615789)

[**Textgenrnn** 26](#_Toc37615790)

[Using the interface 26](#_Toc37615791)

[Interface options 26](#_Toc37615792)

[**References** 27](#_Toc37615793)

# **Possible challenges:**

To start this module, I had to pick 4 possible challenges I wanted to tackle and then I would be assigned on to work on, these are the 4 I chose:

1. C4.C.xiv: Automated creativity

Given some information generate plots/story synopses.

1. C5.C.i: Procedural interactive content

Auto-generate levels and have an ai solve them efficiently

1. C8.iii.i: Novel user interfaces

Visualization of data in VR

1. C6.B: Big data and data analysis

Clean and normalise tweets and categorise them.

# **Chosen challenge:**

My chosen challenge is:

C4.C.xiv:

Use existing techniques such as GAN’s or constraint solving algorithms to learn how to synthesise examples that are similar to the dataset. Plots/story synopses(e.g. IMDB movie plot synopsis)

## Things to research

The following is a list of all the things that I have researched to help tackle the problem. Each things I need to learn will be described in its own part.

* Pyhthon
* Machine learning
* General adversarial networks (GAN’s)
* Constraint solving algorithms
* Template rules-based methods
* Statistical methods
* Natural language processing (NLP)
* HuggingFace’s transformers library : <https://github.com/huggingface/transformers>
* SpaCy : <https://spacy.io/>
* GPT-2
* Gwern -  <https://www.gwern.net/GPT-2>
* Word2vec
* Deep learning

# **Python: The basics**

Pythons syntax is very different to other languages such as c-based languages and java. For example, semicolons are not required at the end of a line of code, curly braces are not used either and indentation is used to define scope instead. The following is intended for people who are sufficient in other languages such as java or c-based languages. Most of the following information on python was found at <https://www.w3schools.com/python/default.asp>

## Commenting

Python does not support multiline commenting, so the following is the only way to comment in code:

# This is a comment

## Declaring variables

In python when we declare variables, we don’t need to assign a data type to them it is automatically done so the following would work

Number = 14

Name = “Bill”

Height = 5.8

Sex = ‘M’

## Arrays and collections of data

Along with arrays There are 4 data collection types in python:

* Array:

Arrays act as they would in other languages.

* List:

This Is a collection which is ordered and changeable. It allows duplicate members.

* Tuple

This is a collection which is ordered and unchangeable. It allows duplicate members.

* Set

This is a collection which is unordered and unindexed. No duplicate members are allowed.

* Dictionary

This is a collection which is unordered, changeable and indexed. It doesn’t allow duplicate members

Collections are declared with like so:

arrayOfNumbers = [1,2,3,4,5,5] # Array

listOfNumbers = [1,2,2,4,5,5] # List

tupleOfNumbers = (1,2,2,3,4,5) # Tuple

setOfNumbers = {1,2,3,4} # Set

dictionaryOfWords = { #Dictionary

“brand” : “Ford”,

“model” : “Mustang”,

“year” : 1964

}

Python has 0 indexing on their lists so 0 is position 1.

Python also has something called negative indexing. This allows you to select items from a list working backwards.

arrayOfnumbers[-1] # This would equal 5

Python can also use ranges when getting list indexes

arrayOfnumbers[2:4] # This would return 3,4 and 5

You can get the length of a list using the following method

len(arrayOfnumbers) # will return 6

To add an item to a list you do the following

arrayOfnumbers.append(4) # this will add 4 to the end of the list

arrayOfnumbers.append(3,7) # this will add the number 7 at the 4th position of this list

## Conditional statements

If, if else and nested if else statements are done as shown:

if condition :

#Do this

if condition :

#do this

else :

#do this

if condition :

#do this

elif condition :

#do this

else :

#do this

## Loops

The 2 loops you can use in phyton are for loops and while loops they are constructed as following:

For Loop:

Repetition loop:

for x in y:

#do this

Range loop:

for x in range(y):

# do this for all values up to position y

for x in range(m,n):

# do this for all values from position m to position n

While loop:

while x < y:

# do this while x is less than y

## Methods

Methods in python are declared using the def keyword

def myMethod():

# The methods code

Methods can also be declared with parameters:

def myMethod(param1,param2):

# code of the method

Methods are called as so:

myMethod(x,y) # method call with 2 parameters given

## User input

To allow user entry we use the following:

X = input(“Enter x variable: ”) # this prints “Enter X variable:” to console and then accepts user input

## Try catch blocks

Try catch blocks in python are called try except blocks and are structured like this:

try:

# do this

except:

#if an exception is thrown

Those are the basics of phyton

# **What is machine learning?**

As defined by expertsystem.com:

“Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machinelearning focuses on the development of computer programs that can access data and use it learn for themselves”

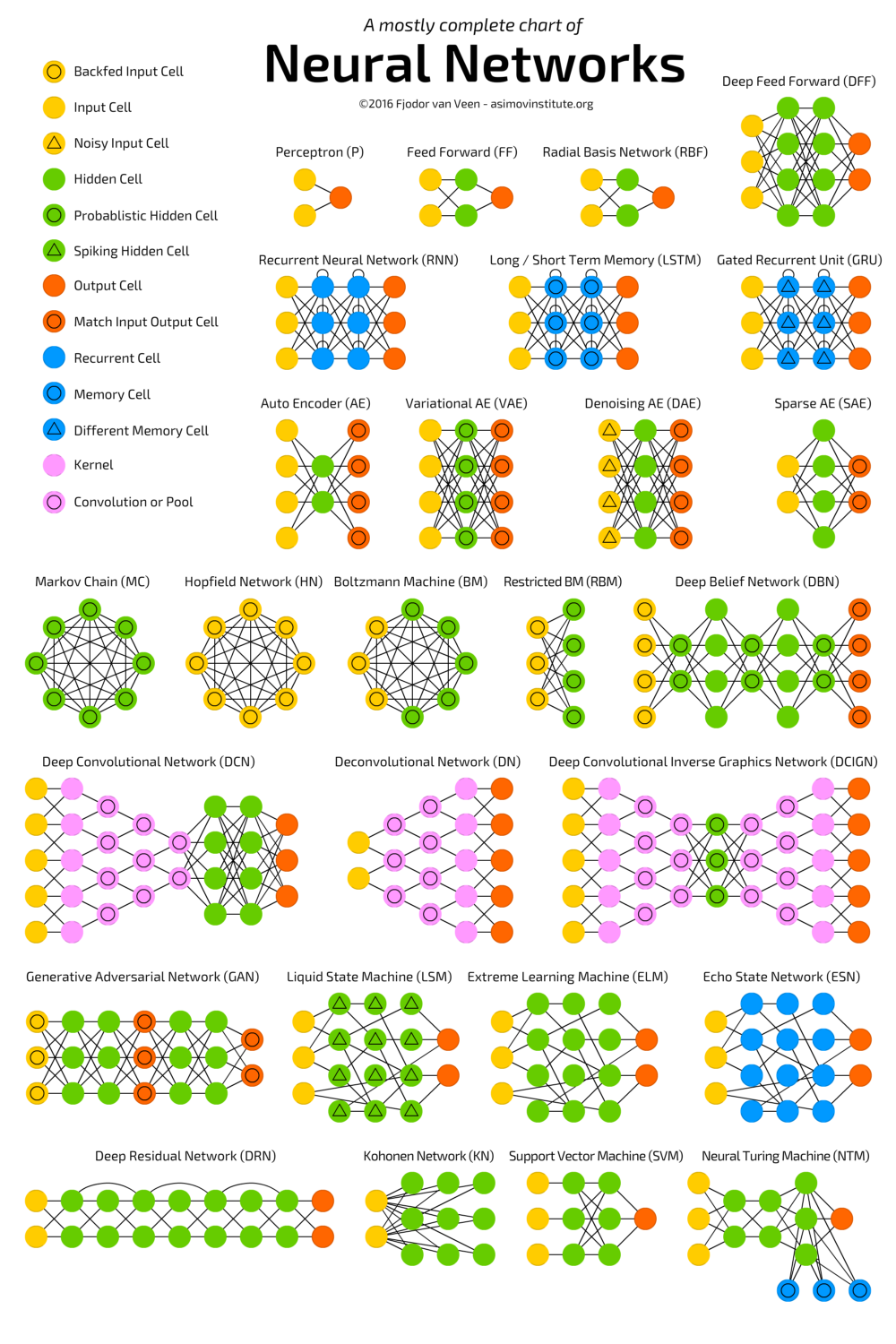
Machine learning is done with the use of neural networks

## What is a neural network?

As defined by Investopedia.com:

“A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria.”

There are lots of different types of Neural networks and each serves a specific purpose for its required task. The simplest of which is called the perceptron



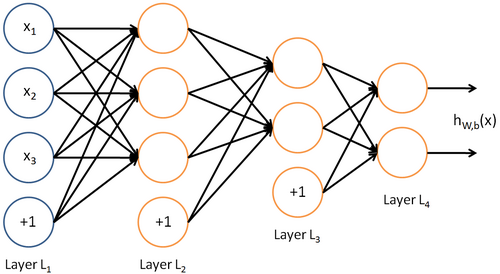
## What makes up a Neural network?

As the name suggests a Neural network is based on nature as it tries to imitate the brain. It is a connection of neurons that form a network where information must be passed along the neurons for some action to happen.

The complete operation of a neural network(NN) is easy to understand Some inputs are put into the first neurons, they are passed through the network as calculations are done to the input numbers, they are then outputted from the network.

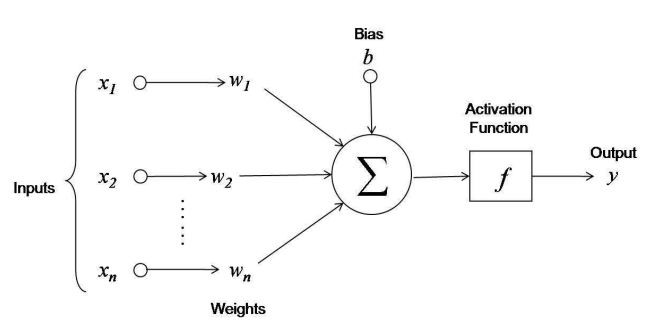
The most basic of NN’s are arranged in columns, so that a neuron on the column can only be connected to neurons from columns and . Other types of architecture are available, but we will focus on the simplest form for now.

Now that we have a basic understanding, we can represent this network .



NN’s are usually read from left to right. Layer 1 is the input layer where data that will be processed by the NN is input, layer 2 and layer 3 are internal layers called the hidden layer, this is where the calculations to our input are done. Layer 4 is the output layer and is where the result of the input will be output. The neurons with +1 at the bottom of L1, L2 and L3 are called bias, these will be explained later.

## What does a neuron do?

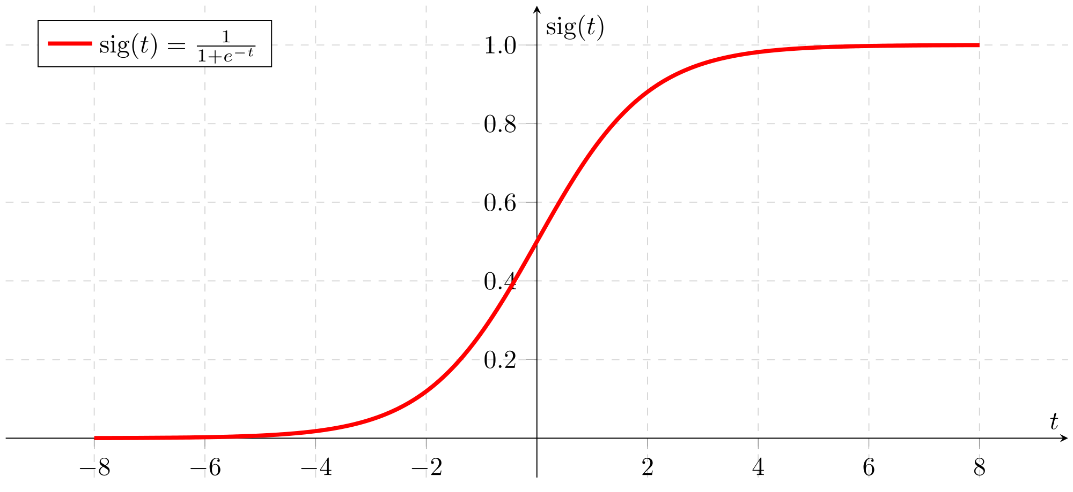
The operation of each neuron is simple:

A neurons job is to add up the value of all its previously connected neurons. Before adding the number from the previous neuron that number is multiplied by the weight.

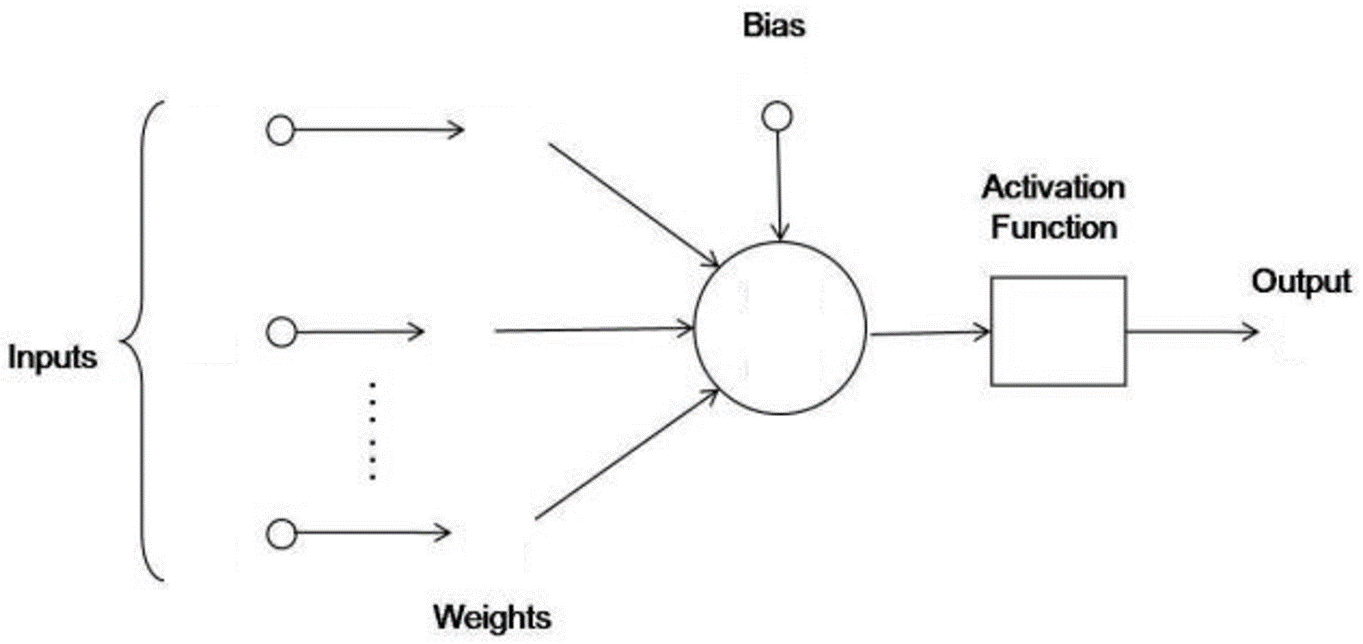
The weight is a number assigned to the connection (line) between neurons and each connection has its own weight. The weight values are modified during the learning process to better fine tune to NN.

optionally a bias value can be added. This is a specified value and is chosen before the learning phase of the network.

Finally, after the final number has been calculated it is run through an “Activation function” to obtain the final value for the neuron. The activation function is usually just to convert the value calculated to a number between 0 and 1 (For this example we will use a sigmoid function). Other functions exist and may change the limits, but they all have the same intention of limiting the number.

Sigmoid activation function:

### Neuron calculation example

Here is an example of a neuron calculating a value with the following information:

0.6787(4dp)

0.7476

0.5

0.04

0.78

0.25

0.98

0.17

0.46

= 0.46 = 0.25

= 0.17 = 0.78

= 0.98 = 0.04

= 0.5

Before activation function =

= 0.5 + ((0.46\*0.25) + (0.17\*0.78) + (0.98\*0.04)) = 0.7476

Applying activation function =

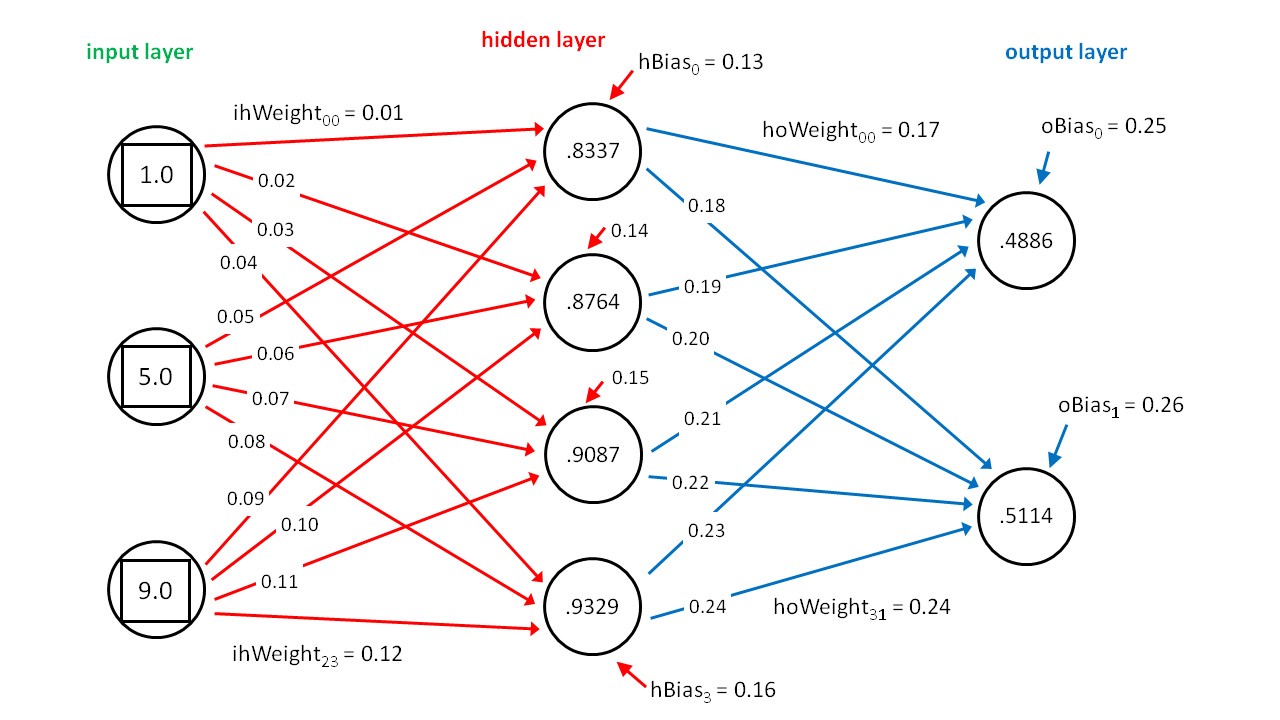
= = 0.6787(4dp)

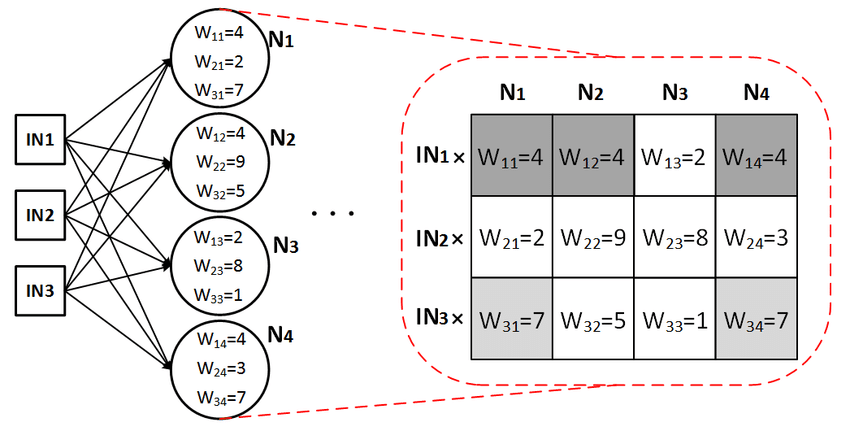
Neuron formula for this example=

General formula =

### Complex example

Here is a more complex example with hidden layers:



In cases like these, matrices are used in the calculation as using numbers would be more complicated. Each input node acts as a scaler and the weights are arranged in matrices:

Bias is still added to each neuron in the hidden layer individually and the activation function is still applied.

Apart from the sigmoid activation function other popular activation functions include:

Rectified linear units (ReLU):

With ReLU we ensure the output doesn’t go below into the negatives therefore if then y remains the same but if the we set equal to .

tanh

This is just a simple activation function where we return tanh(y)

## How does the network learn?

Now that we understand architecturally how a NN works we need to understand how it learns as this is how we can turn an arbitrary amount of calculations into a useful tool.

To teach a neural network we need to train it for its specific job, we do this by giving it a large training set where the input and output is known so we will be able to see if the NN is correctly processing the data. At the beginning the NN won’t get the answer correct, unless it does by pure chance, but more often than not the output will not be what was expected. This is where the neural network starts leaning.

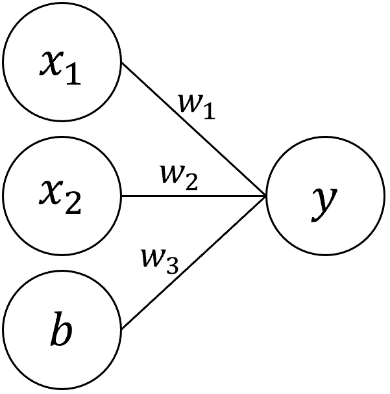
When an output is given that isn’t the expected outcome the NN will change the weights of the connections. After changing the weights, it runs another known problem and sees if it is correct this time. That is in simplest terms how the network learns.

A more complex way of learning which is more common is called backpropagation. This process consists of going back through the NN from the output and inspecting all the connections to see how changing the rate would affect the output.

When making a NN you need to create a variable called “Learning rate” which will handle how the network learns. The learning rate is a value that is used to change the weights, the larger the value the more it will change a weight.

# **Making a neural network**

The simplest neural network is called the perception neural network(P). To understand the basics of neural networks I will make a perception network in python3.

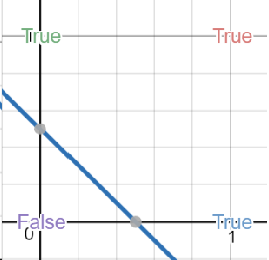
My network will be used to solve the OR gate problem. It will have 4 neurons overall, 2 input, 1 bias and 1 output. There will be 3 weights to connect the neurons. This is what my NN will look like:

This type of network is called a classifier as it is most useful for classifying things. For example, if we look at the logic table for an OR gate:

|  |  |  |
| --- | --- | --- |
| A | B | OR |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

|  |  |  |
| --- | --- | --- |
| A | B | OR |
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

We have 2 inputs (A and B) and one output. Given 2 inputs our NN has to classify which output is expected. A classifier is able to solve a classification problem where when plotted in cartesian coordinates the separate classifications can be separated by a single line. So if we use the A and B as our XY coordinates we get a graph like this:



This is a problem that can be solved with a classifier NN.

## The code

The first thing we need to do is import the library’s we need and to classify the variables for the NN:

Import numpy, random, os

lr = 1 # This is the learning rate of the NN

bias = 1 # The value of the bias neuron

weights = [random.random(),random.random(),random.random()] # This creates 3 random weights and puts them in a list, 1 for each neuron connection

Next, we must create the function that handles the neuron calculation. So, we have to multiply the inputs by the weights and add them, multiply the bias by the weights and add it and then finally apply the activation function. In this step we will also create the functionality to teach the NN.

Def Classifier(x1,x2,expectedOutput) :

# Calculate neuron

output = x1\*weights[0] + x2\*weights[1]+bias\*weights[2]

# Activation function

if output > 0 :

output = 1

else :

output = 0

# Create a metric to track error amount

error = expectedOutput – output

# Modify the weights based on the error

weights[0] += error \* x1 \* lr

weights[1] += error \* x2 \* lr

weights[2] += error \* bias \* lr

Here we create the function with 3 inputs, x1,x2 and expectedOutput. The expected output is used in the learning process of the NN as we don’t want to tell it how to solve the problem we want to tell it what it should output given certain inputs so it learns how to turn its inputs to an expected outcome.

We are not using the sigmoid activation function here as our expected output will only ever be 1 or 0, using the sigmoid function would return a decimal value close to 1 or 0 depending on what the output should be. This activation function is called the Heaviside function and its only purpose it to turn any +ve output to 1 and any -ve output to 0. Mathematically this is what the function is:

If we where to use the sigmoid function, we would program it like this:

# Sigmoid function

output = 1/(1+numpy.exp(-output))

After applying our activation function, we need to create the functionality to modify the weights during the learning process. To do this we create a variable called error, this takes the given output away from the actual answer and uses this difference to change the weights in hopes of converging closer on the correct weight for the problem. Now that we have the functionality set up we have to train the NN.

for i in range(50) :

Classifier(1,1,1)

Classifier(1,0,1)

Classifier(0,1,1)

Classifier(0,0,0)

This is the training process, normally we don’t want to give the network its entire list of possibilities, but as there are only 4 cases it should be fine in this scenario. We choose training time based on how we want to train the network. In this Example we train the network on each possibility 50 times. Undertraining occurs in incorrect answers from the network while over training can make the network too specific to the training set.

exit = False

while exit == False:

print(“Input var X:”)

x = int(input())

print(“input var Y:”)

y = int(input())

output = x\*weights[0] + y\*weights[1] + bias\*weights[2]

#Activation function

If output > 0 :

Output = 1

else :

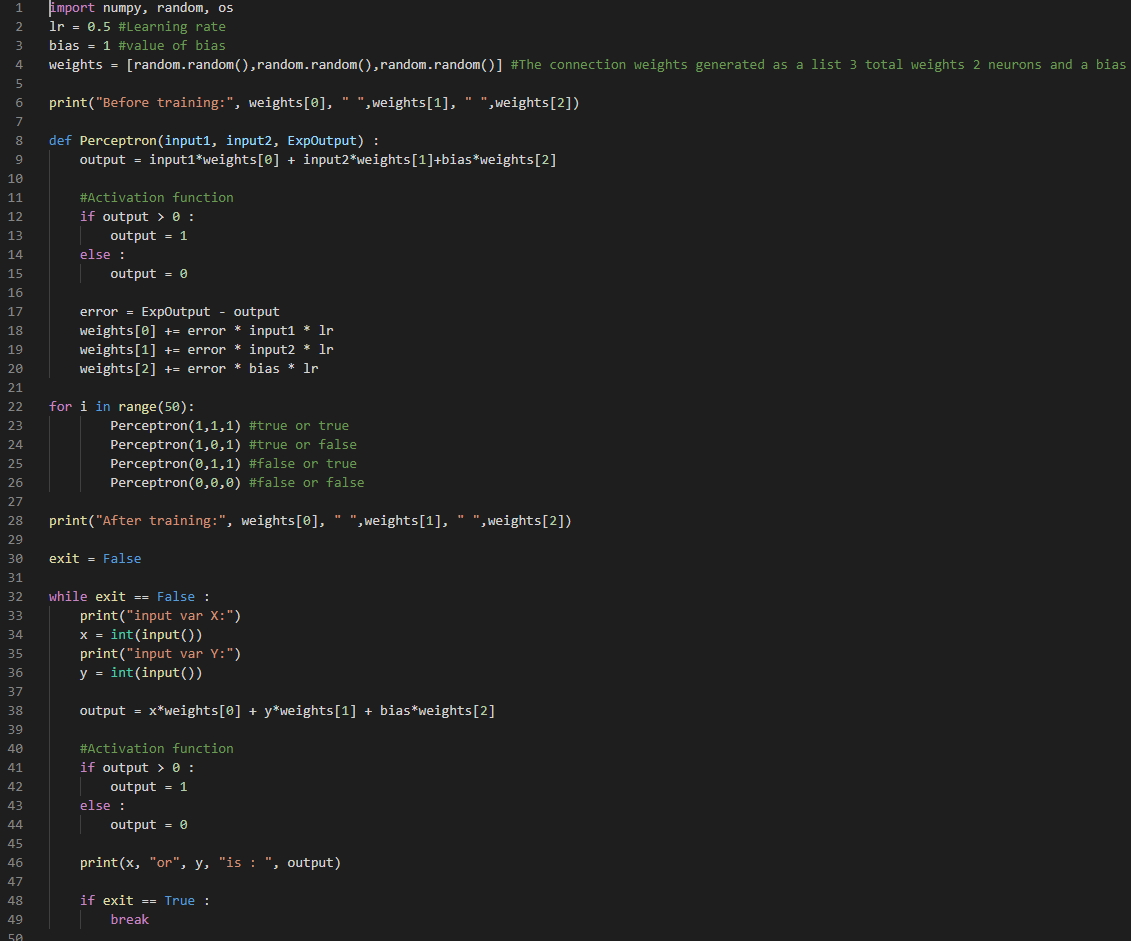
output = 0

print(x, “or”, y, “is :”,output)

if exit == True :

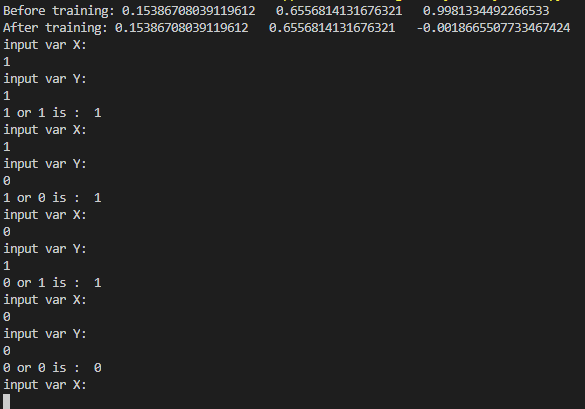
break

This last piece of code is so that user can enter the X and Y variables to check to make sure it works.



Here is the final compiled code in visual studio code. As you can see, we can make this neural network in less than 50 lines of code.

I have added a few more bits of code through to display the weights before and after training.

This is the console after compiling the code and trying all possible combinations of the OR gate.

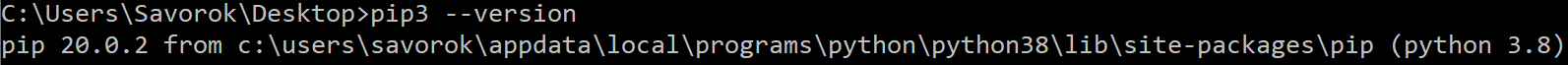
Here we can see that the weights are displayed before and after training. The only weight that changes is the weight on the bias. We can also see the network correct answers every scenario.

# **Python setting up a virtual environment (Python 3.8)**

For this project I am going to set setup tensor-flow in a virtual environment to use. Using a virtual environment will mean that I can install packages separately into it. This helps prevent updates breaking my code as that environment will have the packages specifically used for the projects.

To set up a virtual environment the fist step is to make sure you have python 3.3 or later. I am using python 3.8. to check your python version, use the following command in command prompt:

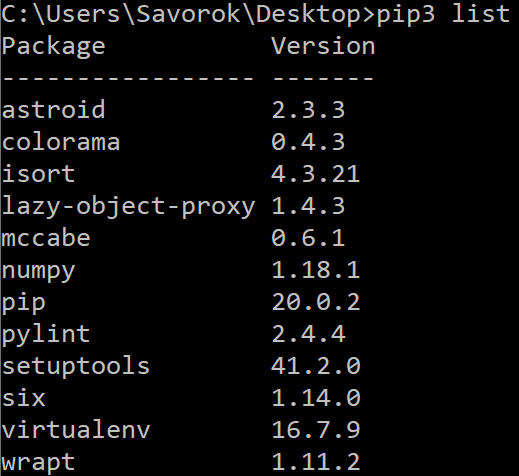
pip3 --version

 You should get something like this:

Next lets check to make sure we have the virtual environment package, use this command to see your installed packages

pip3 --list

 You should get a list like this but the most important thing is that it contains the “virtualenv” module.



Now lets set up the virtual environment. I am going to setup the environment on my desktop so it is easy to find so make sure you have navigated to the directory you want to create your venv in. to navigate to your desktop you can use this command:

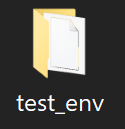
cd Desktop

now that we are In the desktop directory we can create the virtual environment. To do this use the following command:

python -m venv test\_env

 So, this command will create a virtual environment by navigating to python then using the command “-m” to select a specific module then calling the venv module. You must put the name of your virtual environment after venv. In my case I called my environment test\_env.

If done correctly you should have a folder created in your specified directory with the name you assigned it



Now lets activate our virtual environment, to do this we use the following command:

\*the name of your environment\*\Scripts\activate.bat

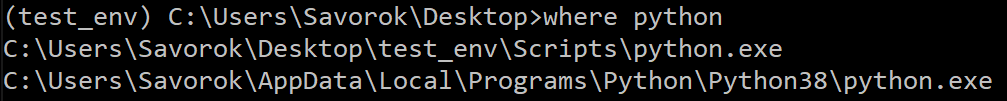
 We will know the environment is activated as in our console we will have it as a prefix for our command line:



To see exactly what environments are running we can use this command

where python

 This will tell us all the paths python has

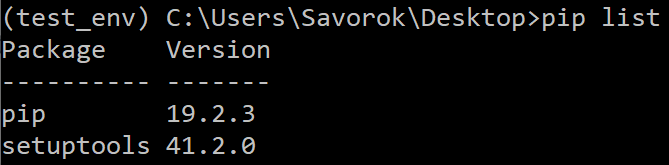


When we want to exit our virtual environment, we can use this command

deactivate

## Using the virtual environment

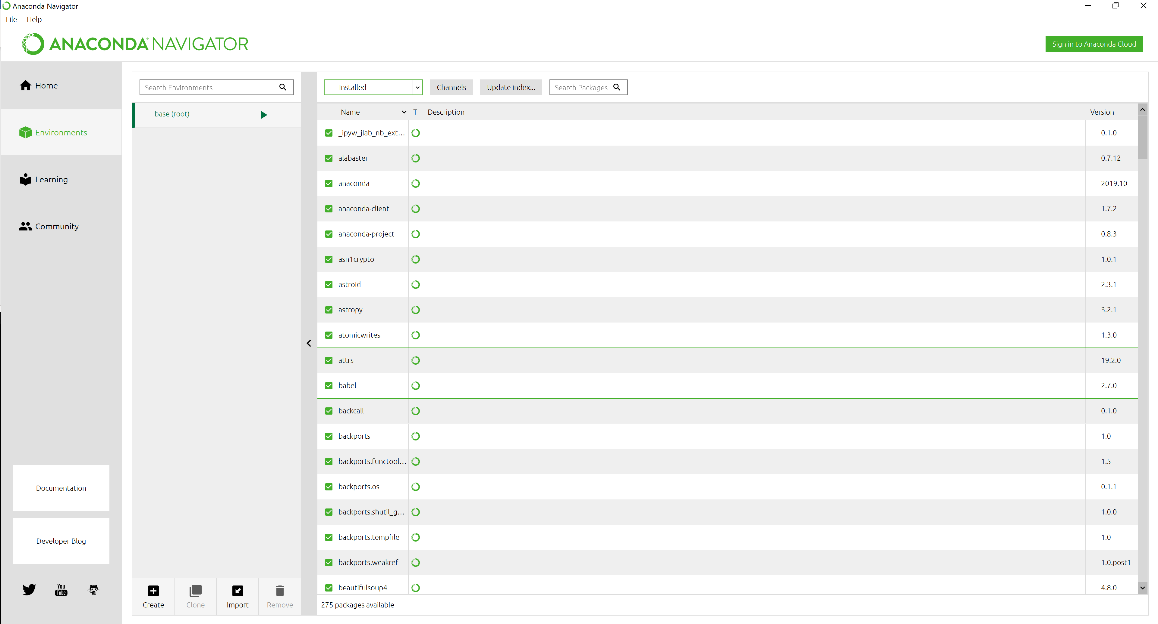
Now that we have the environment set up and we are in it we can run our pip list command to see what modules it is currently running:

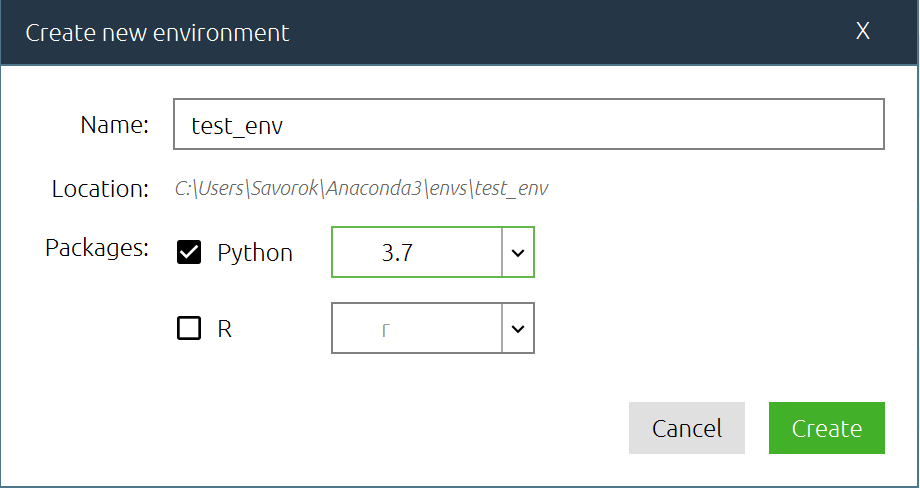


From this point we can now add whatever modules we need knowing they are in the virtual environment.

# **Setting up TensorFlow (Pyhton 3.7)**

As tensor flow has not been updated for python 3.8.x we have to use python 3.7 for using it. To help me with creating virtual environments and other management I will use Anaconda navigator.

After installing anaconda x64 I opened the navigator to the main menu.

Next I selected the create new environment button and created an environment with python 3.7

With the environment created I opened the console command line for this environment and used the following command to install TensorFlow:

pip install --upgrade tensorflow

Now tensor flow is installed on this environment

# **Making Poetry**

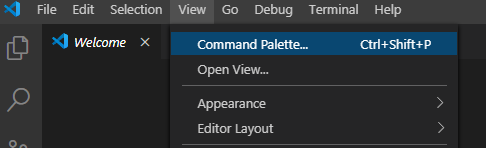
The following is a walkthrough of how to finish the tutorial from:<https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f> and how to get over the problems and hurdles I faced while attempting the tutorial.

## Step 1 – Get a Clone of the project

The first step in this tutorial is to get the GIT clone of the project. I used Visual studio code to import a clone of the project and saved it to a folder in my desktop for easy access.

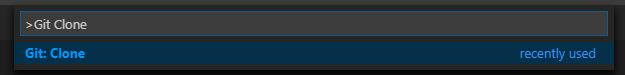
So make sure you have GIT installed from the following link: <https://git-scm.com/downloads>

Although it may differ for your IDE to clone a project in visual studio code go to the view tab and select command palette:



Or press Ctrl+Shift+P

Next type this into the command palette:



And then the following link into the Repository URL: <https://github.com/nshepperd/gpt-2.git>

Make sure you save it somewhere that you can easily access.

You should now have a clone of the project which consists of a folder called gpt-2.

Unless you are using GPU training the next step is to copy the files named encode.py and train.py into the src folder.

## Step 2 – Setup python in a virtual environment

Now we need to setup our Venv. I will be using anaconda to set up a new Venv running Python 3.7

Create an environment in anaconda and make sure you have python working in it by typing this into the cmd console:

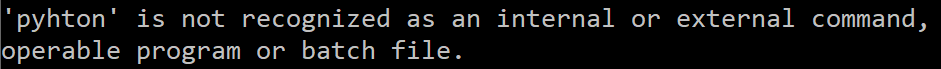
pyhton --version

 You should get “Pyhton 3.7.x” in the console (x dependant of the version of 3.7 you are using)

If you are getting an error about python not being recognised follow this fix otherwise you can ignore this part:

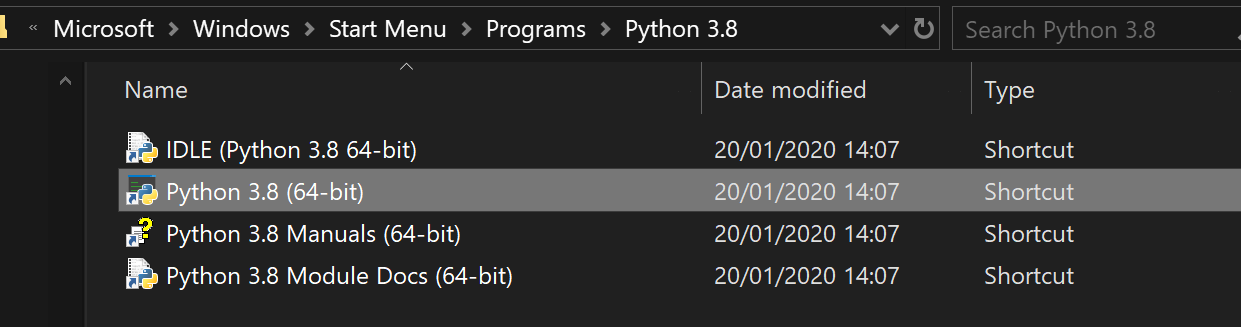
## Python not recognised error fix

If you are getting the following error



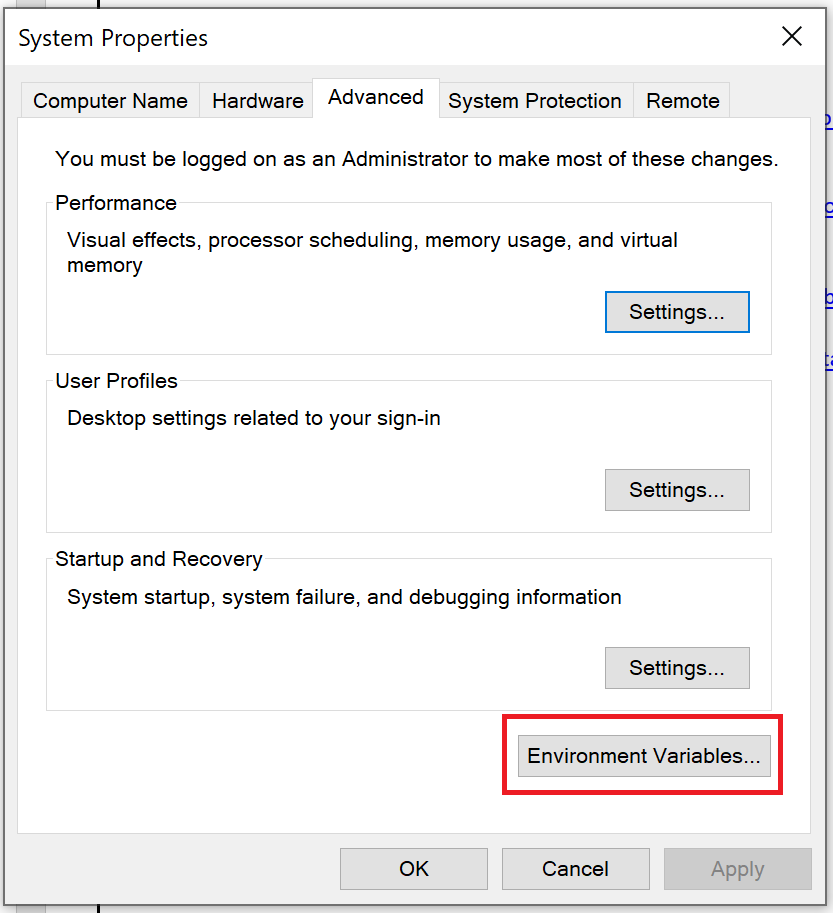
Then we need to define a python path for your system. To do this find where you have installed python. The easiest way to do this is to type python into your search bar and go to the source folder:

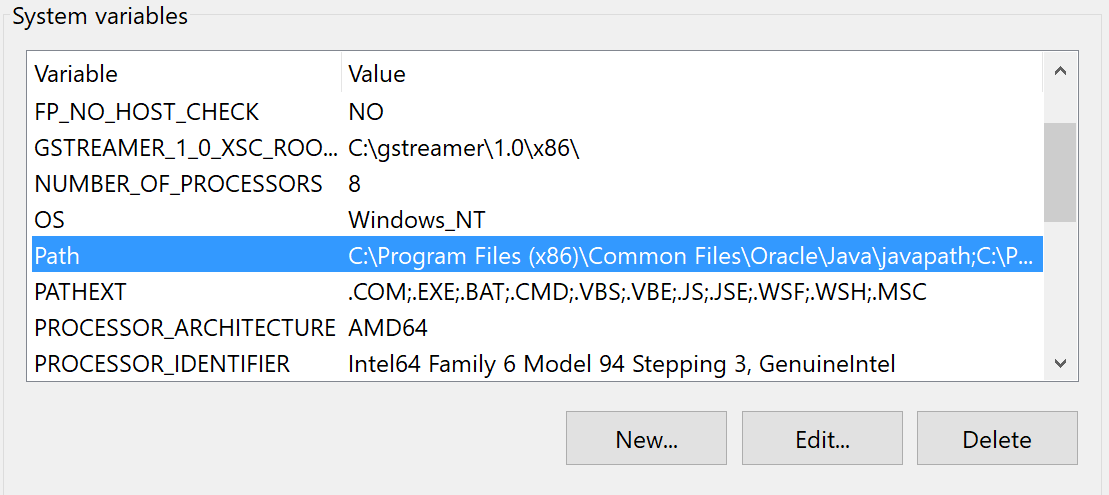




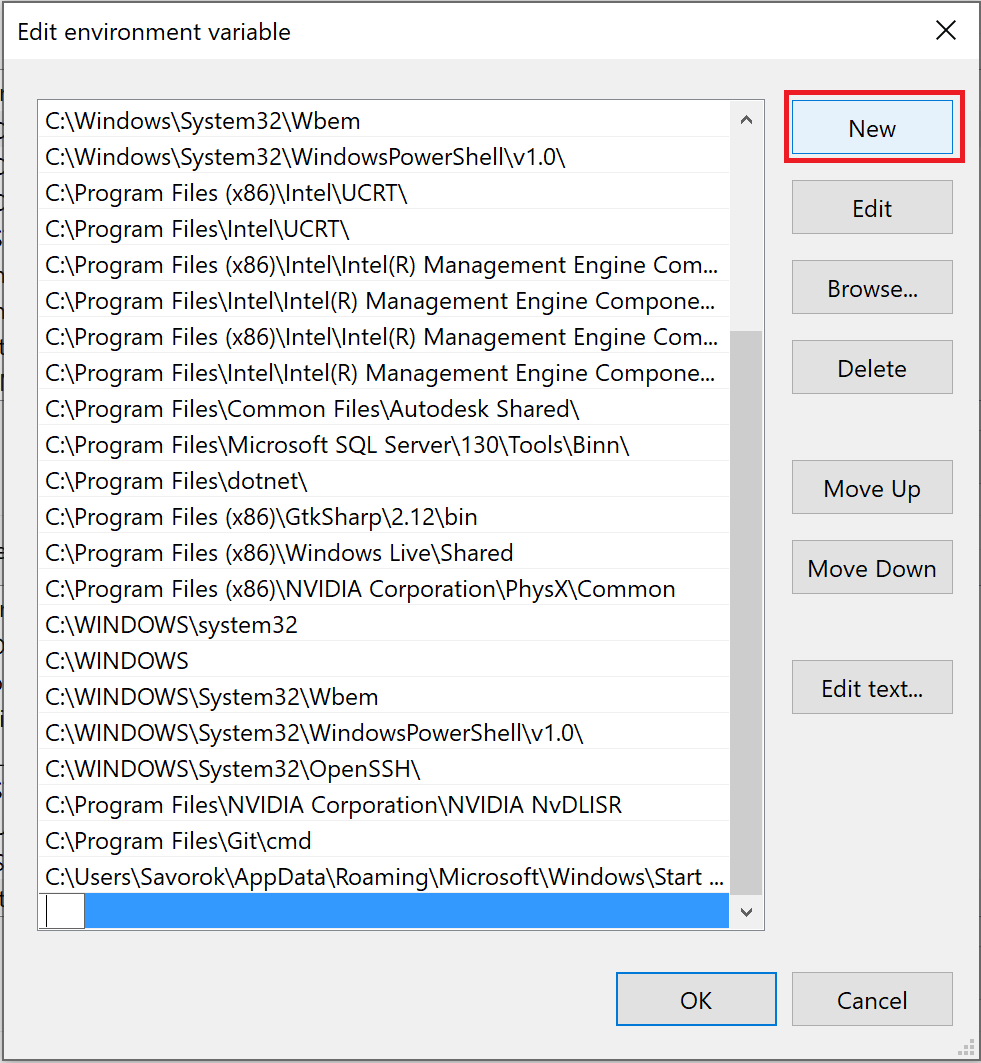
Copy the directory path of this folder. We will use this to point your computer to this folder when we are using python in cmd.

Next search for “Edit the system environment variables” in your search bar or find it in the control panel.

Go to environment settings:

And find the path variable in the system variable list

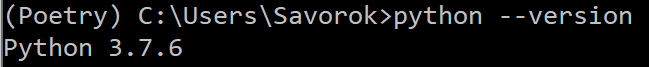
Click edit on this variable and then new to create a new variable.



This new variable should be the file path that you copied from earlier.

After doing this python should work in command line. You can check this by typing this into the CMD console.

python --version



## Step 3 – install all the needed modules

Now we need to install a few modules. The modules we need to install are:

* Fire 0.1.2
* Regex 2017.4.5
* Requests 2.21.0
* Tqdm 4.31.1
* Numpy
* Tensorflow 1.14 (This is important any other version of TF will not work must be 1.14)
* toposort

To install the modules, we use the following command in your venv command console:

pip install \*Name of module\*

 You DO NOT need to put in the version of the module just the name

To make sure you have all the modules installed use this command to check your modules

pip list

## 

## Step 4 – installing the model

Now we need to install the NN model that we will use for generating out poetry

To do this we first need to change our command line destination to inside our project folder. To do this go to where you downloaded the GIT clone and copy the path to the folder that holds the src folder. We can then change the directory by typing the following into the Venv command line:

cd \*Path\*

 your command line should now look similar to this:



Now that we are in this folder, we can run the download model script. To do this we use this command:

python download\_model.py 117M

 To ensure this has been downloaded check the project folder. It should now have a new folder inside of it called models.

## Step 5 – creating a training dataset

As we are creating our own data set you can use any plain text (in English) to train the model. I will be using poetry however for this tutorial. I will show you the formatting of the dataset so you can create your own to train the model for different types of content.

To create this data set, create a normal notepad .txt document and put in your wanted text. Separate each different text with the following:

<|endoftext|>

 Here is an example of what the break between text should look like should look like:

……….

There are beautiful beeches down beyond the hill.

Will you always stand there shivering?

<|endoftext|>

Water ruffled and speckled by galloping wind

Which puffs and spurts it into tiny pashing breakers

Dashed with lemon-yellow afternoon sunlight.

The shining of the sun upon the water

Is like a scattering of gold crocus-petals

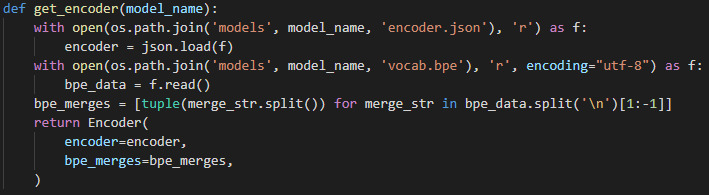
In a long wavering irregular flight.

………..

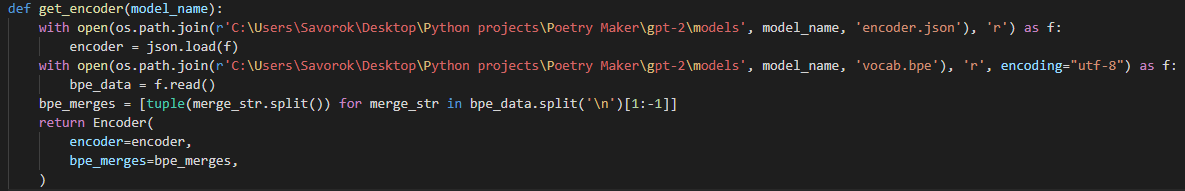
Finally put this .txt file into the source folder of the project.

## Step 6 – encoding the dataset for training

In step 1 we copied the encode.py and train.py file into the source folder so we need to change a few lines of code for the following step to work otherwise it wont be able to find the a directory and will throw an error during encoding.

Open up the encoder.py (not encode.py) and go to the get\_encoder definition:

We need to give the correct path to the open command as a raw string. To do this replace ‘models’ with the path to the models folder that is in the project folder. Then put an r before the apostrophe:



Once we have done this we can go back to our command line and Cd into the src folder. We then run this command:

pyhton encode.py \*Name of data set\*.txt \*Name of data set\*.npz

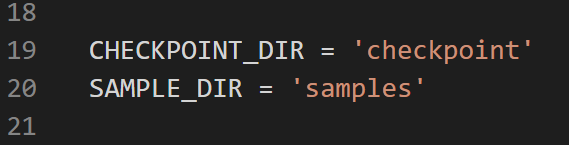
 For the name of the data set just put in the name of the .txt file. The .npz file can have a different name if you want.

## Step 7 – training the model

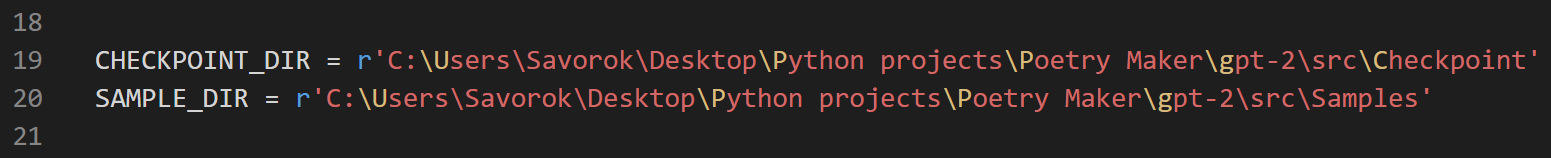
Now we need to train our model so we can start producing samples. Before we do this there’s a bit of code we need to adjust so the model can find the correct directory’s and then we need to create a few directory’s for the model to use.

In the src folder add two new folders “Checkpoint” and “Samples”.

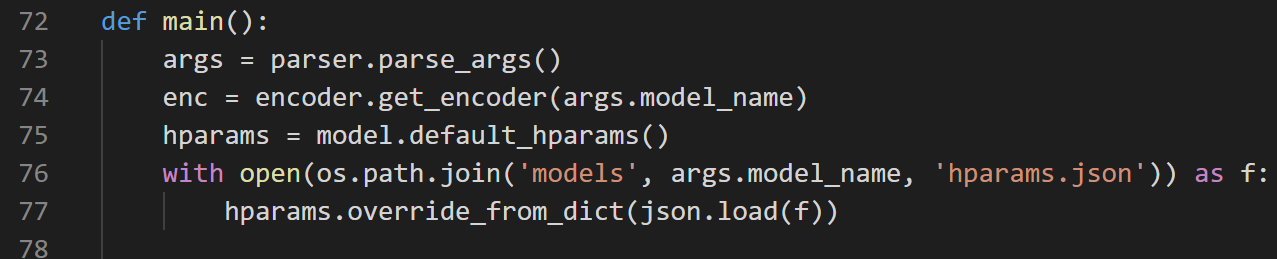
Now open the train.py file and go to lines 19 and 20:



Within these apostrophise put the root of your respective folders and then put an r before each one(This parses it to a raw string so the path will work) so for me it would now look like this:

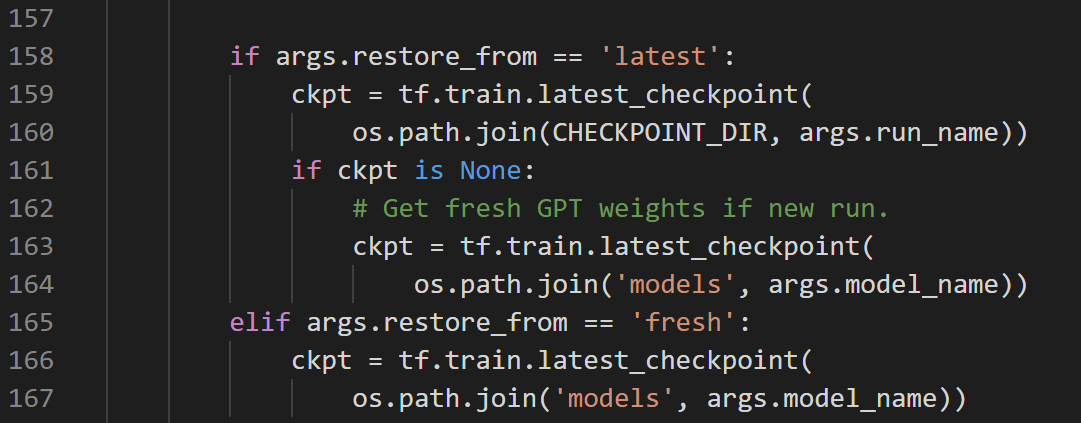


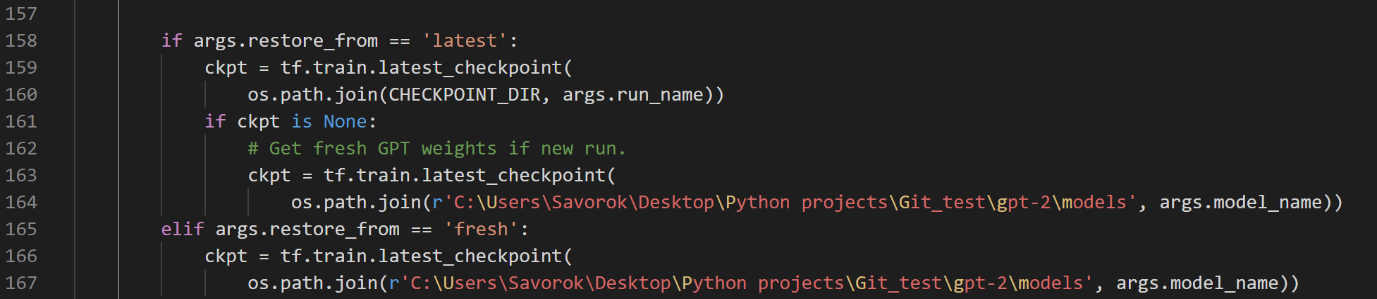
Next we need to go to the main method (it starts at line 72) and put in the path to our model’s folder. This is the folder that was created when you downloaded the model in the earlier steps. The line we need to add this is line 76:



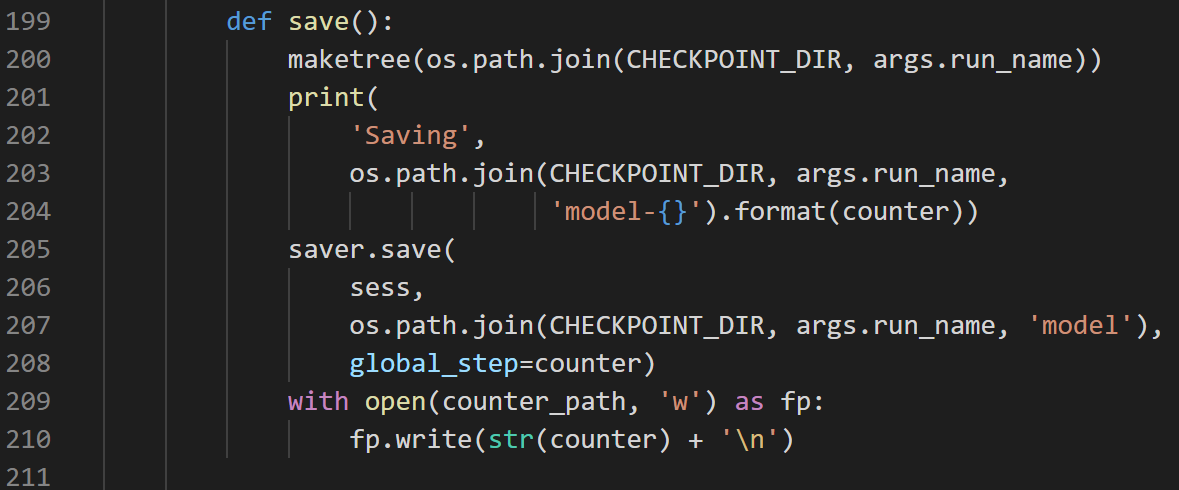


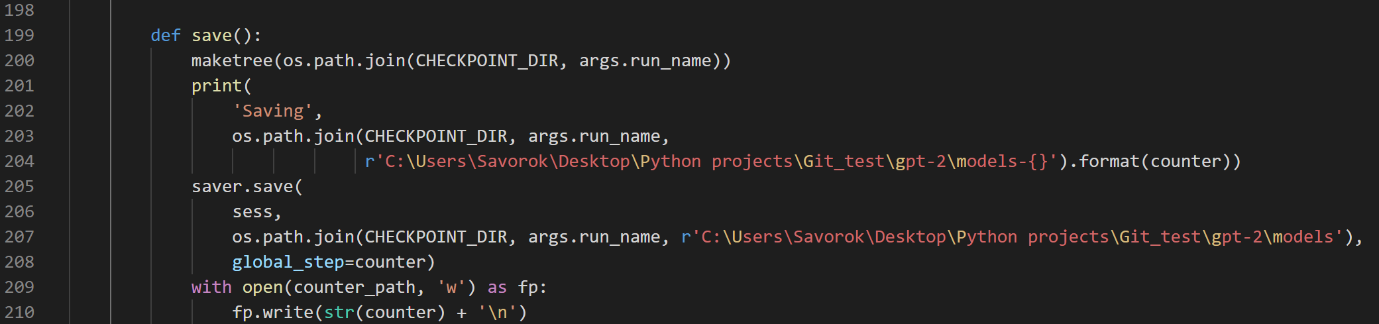
The next place we need to add this path is in some code around lines 160





The last place we need to add the root is around line 200

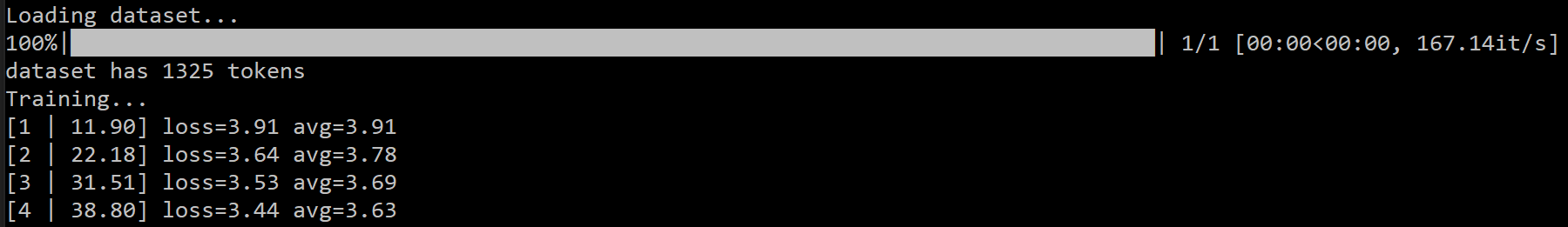




Now the last step is to execute the training command, it is the following:

python train.py --dataset \*Name of dataset\*

And that’s it. If you get the following you have done everything correctly:



You will most likely have lots of errors in your command window but as long as you get to what is displayed above you can ignore any errors.

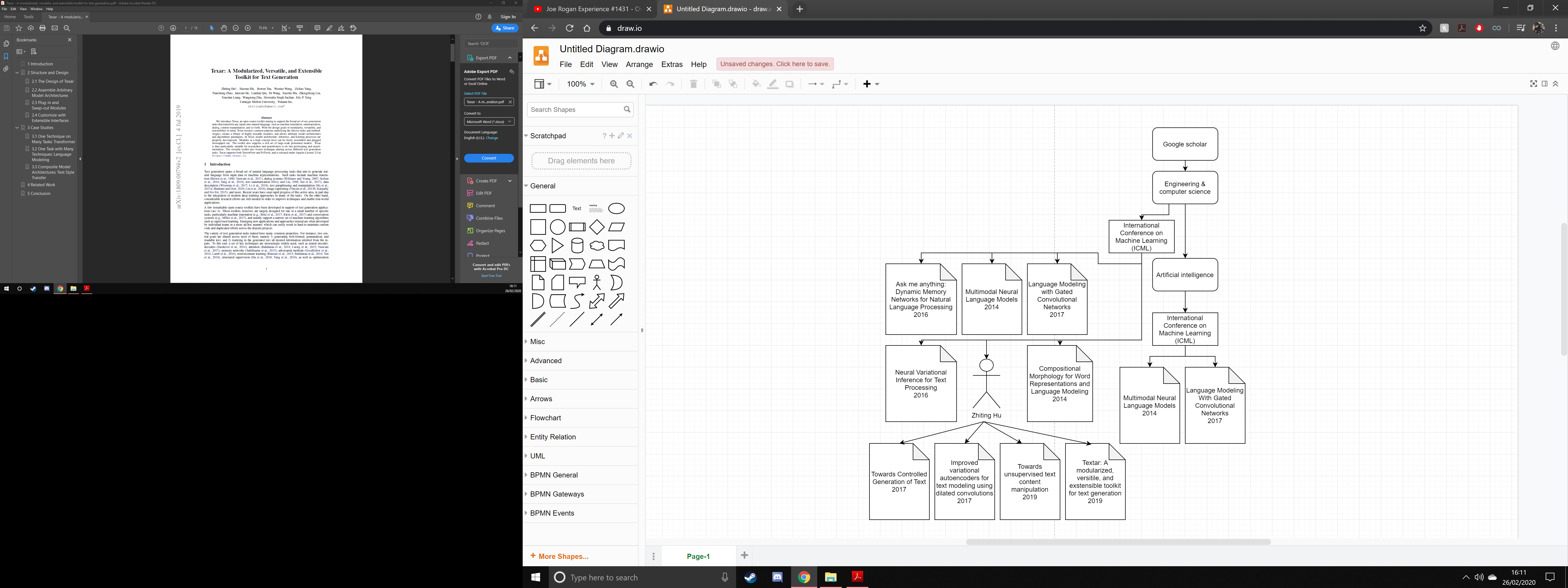
To stop training you must have the window selected and press Ctrl+C. (Don’t worry stopping training won’t make you loose any progress on your model you can resume where you left off)

# **How to find research documents**

After setting up the GPT-2 model we are nearly at the cutting edge of this area of computing. So now the next step is to heavily research the latest discovery’s in the field and see if I can apply my knowledge to try and push the field forward or make it easier for others to.

To research we are going to use google scholar.

I have created a map to show my path to finding all the relevant papers on my topic:



# **Textgenrnn**

Textgenrnn is a pretrained model I found on github made by minimaxir. I got the model up and running and made a python script to interface with the model. The following is an explanation of how to get my interface running so you can easily use the model. The model itself can be found at <https://github.com/minimaxir/textgenrnn> and the methods for interface that I made can be found here <https://github.com/minimaxir/textgenrnn/blob/master/docs/textgenrnn-demo.ipynb>

## Using the interface

The interface I created is a simple python script to interact with the model. (the interface can be downloaded from my github: <https://github.com/Savorok/Computer-Science-Chalenges--40259070>)

Step 1 – Create a venv in anaconda using python 3.7

Step 2 – install the following modules to the venv:

H5py

Scikit-learn

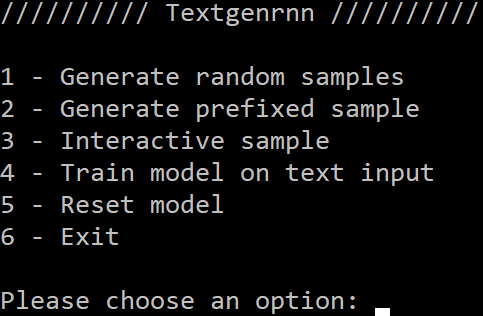
Tqdm

Tensorflow==2.1.0

Step 3 – Cd to the folder containing my script and call the following command in terminal



Step 4 – The main menu should now load and you will be greeted with this menu



## Interface options

1 – Generate random samples

This will generate a chosen number of random samples

2 - Generate prefixed samples

This will let you generate a chosen number of samples with a prefix of your choice

3 - interactive sample

This will let you generate samples yourself by giving you the top N choices the model has for the next part of the sample

4 - train model on text input

Will let you train the model on your own dataset

5 - reset model

Will reset the model back to its pretrained state

6 – Exit

Will close the script and return you to the console

# **References**

* Why we use python for machine learning: <https://www.netguru.com/blog/why-is-python-so-good-for-ai-machine-learning-and-deep-learning>
* Python code examples and documentation: <https://www.w3schools.com/python/python_tuples.asp>
* What is a neural network:<https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf>
* The mathematics of neural networks: <https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05>
* Venv help setup video (python 3.8): <https://www.youtube.com/watch?v=APOPm01BVrk>
* Setting up tensor flow: <https://www.tensorflow.org/install/pip>
* Poetry tutorial - <https://www.gwern.net/GPT-2>
* GPT-2 tutorial: <https://medium.com/@ngwaifoong92/beginners-guide-to-retrain-gpt-2-117m-to-generate-custom-text-content-8bb5363d8b7f>
* How to clone a github project with vis studio code: <https://www.youtube.com/watch?v=pVQCJ6sY8AQ>
* Python not recognised as a command fix: <https://www.youtube.com/watch?v=sRRFgF4Us68>
* How to make HTML websites: <https://www.w3schools.com/howto/default.asp>
* Texar: <https://github.com/asyml/texar>
* Bert model explained: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>
* Textgenrnn - <https://github.com/minimaxir/textgenrnn>