Justin Brown

EPQ Discussion

Contents

[Researching Initial Concepts and Mathematics 2](#_Toc121093412)

[Best Programming Language To Use 3](#_Toc121093413)

[Multilayer Perceptron (Sanderson, 2017) 4](#_Toc121093414)

[Generative Adversarial Networks 6](#_Toc121093415)

[Recurrent Neural Networks – Long Short-Term Memory (Rohrer, 2017) 7](#_Toc121093416)

[Cross Validation (Starmer, 2018) 8](#_Toc121093417)

[Bias and Variance (Starmer, 2018) 9](#_Toc121093418)

[Support Vector Machines 10](#_Toc121093419)

[Testing Code 11](#_Toc121093420)

[Microsoft Model Conversational Generation 12](#_Toc121093421)

[Intent Detection – Test 14](#_Toc121093422)

[Facebook Model Conversation Generation - Test 19](#_Toc121093423)

[Deciphering Maths Calculations 21](#_Toc121093424)

[Theory of Implementation 21](#_Toc121093425)

[Problems Encountered 22](#_Toc121093426)

[Word Classification 22](#_Toc121093427)

[Building The GUI 23](#_Toc121093428)

[Deciding what library to use 24](#_Toc121093429)

[Creating The GUI 26](#_Toc121093430)

[Implementing the GUI 28](#_Toc121093431)

[Detecting Context Change 29](#_Toc121093432)

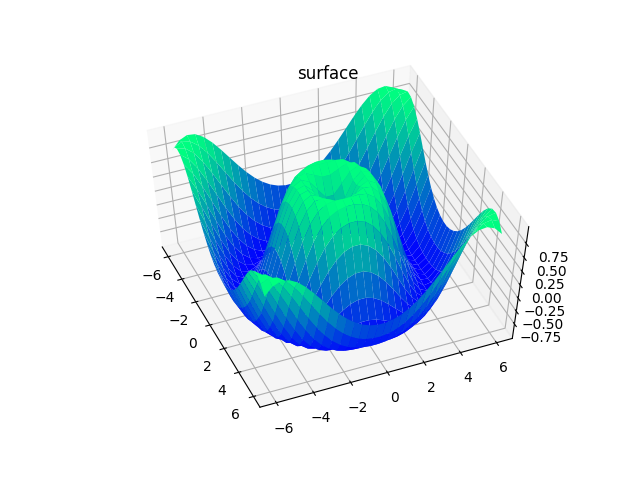
[Bibliography 31](#_Toc121093433)

[Links 32](#_Toc121093434)

# Researching Initial Concepts and Mathematics

## Best Programming Language To Use

Python is the best programming language to program AI programs for many reasons: it’s really easy to read and understand, it has a massive community with people who dedicate their lives to libraries and APIs which will be very useful, for example the google TensorFlow API and the scikit library, it is also a very powerful language.



* The majority of AI libraries were coded in or with python in mind. It also has very well documented libraries
* Libraries such as matplotlib make reading data very easy, this may be beneficial when trying to visualise the error/ success rate of an algorithm
* It is also an interpreted language meaning it can be run on virtually any machine, whereas a language such as java is machine specific and can only be run on machines similar to the machine that it was compiled on. (Arora, 2022)
* The C family (C++, C#, etc) is a very powerful language, in some cases such as the efficiency and the processing rate of the program, is better, however, I do not know how to code and the C family has a very steep learning curve, so it would be very hard to learn enough to be able to create an AI program

Evaluation:

These sources were very useful, it allowed me to decide that I am going to use python since it has the largest active community of programmers surrounding AI which will be very useful since I can easily get help or find API’s. In terms of validity of the source, I have compared their findings compared to other sources (Tirendaz Academy, 2022) and they all give the same results (python is the best).

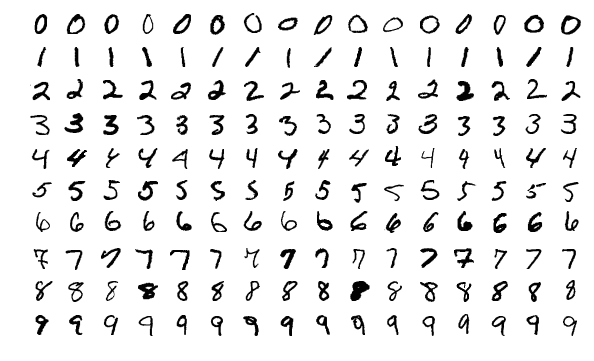
## Multilayer Perceptron (Sanderson, 2017)

I learnt this information by (Sanderson, 2017)

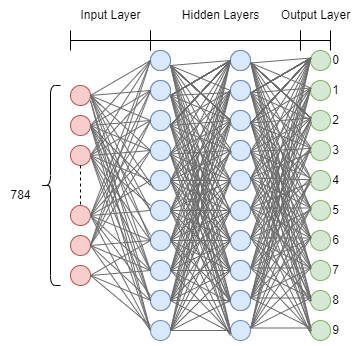
Multilayer perceptron neural networks are the foundations of machine learning. First developed by American psychologist Frank Rosenblatt, it aimed to combat the limitations of single perceptron algorithms, which was “that it couldn’t represent the XOR gate, exclusive OR, where the gate only returns 1 if the inputs are different.” (Bento, 2021). Although this type of algorithm is very old, it is a prerequisite for being able to understand the plethora of other algorithms.

*All my diagrams were based of diagrams and information by* (Sanderson, 2017)*.*

The Handwritten Digits Example



This example takes a series of differently drawn handwritten digits. The neural network bellow is designed to categorise the digits with the correct value. An example digit (NIST, 2016) is shown on the left.



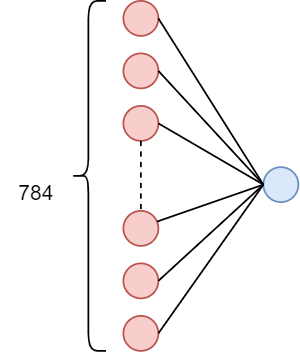
The term neuron can be thought of as a function, it takes an input, does processing on it and outputs it.

A 28x28 image is entered into a neural network. In the first layer, each pixel is assigned to a neuron, the activation number in these neurons corresponds to the greyscale value / brightness of that particular pixel.

This means that there will be 28x28 (784) neurons.

Understanding The Activation Function

I learnt this information by (Sanderson, 2017)



To calculate the activation number, we must combine the weights and activation numbers (of the neurons in the previous layer) to get the new activation number.

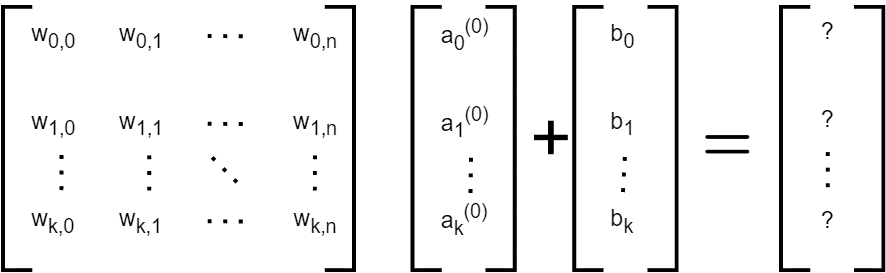
We use the sigmoid (σ) function to reduce the values to between 0-1:

σ (w1a1 + w2a2 + w3a3 + … + wnan)

However, this function does not allow for any leeway in the activation numbers of the previous, to resolve this we add need to add a bias:

σ (w1a1 + w2a2 + w3a3 + … + wnan - 20)

This only activates when the weighted sum is > 20



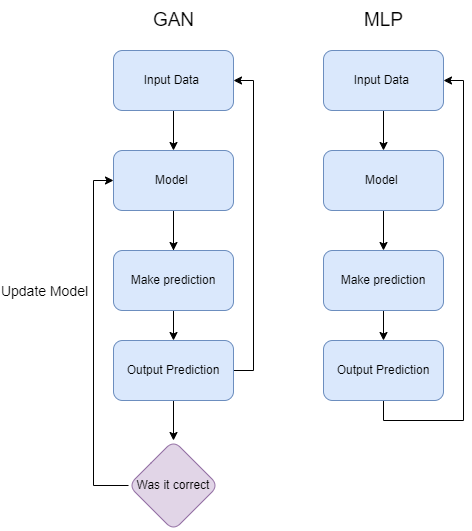
The matrix above shows a visual representation of how the activation number is being calculated. However this canal be shortened to **an  = σ (wan + b)**

Evaluation

This source was very useful, although it doesn’t explain how to code it, it does explain the concept which is a vital understanding to be able to code any program. However, I don’t think that this actual model will be useful considering it is 50 years old and technology has progressed a lot.

## Generative Adversarial Networks

I learnt this information by (Brownlee, 2019)



GANs work in a way such that the model used to make a prediction is updated every time it has an interaction. This is much better than the multilayer perceptron model because it is not static and it learns, this would be much better for an AI that has lots of interactions with humans. A useful analogy to use would be a child learning. When a child does something that is not acceptable, such as punching someone, they are told off and explained why that is wrong, this is the same thing that happens with GANs, they update their model so that next time they do not make the same mistake.

The information below was altered but gathered from (Rehman, 2020)

Benefits:

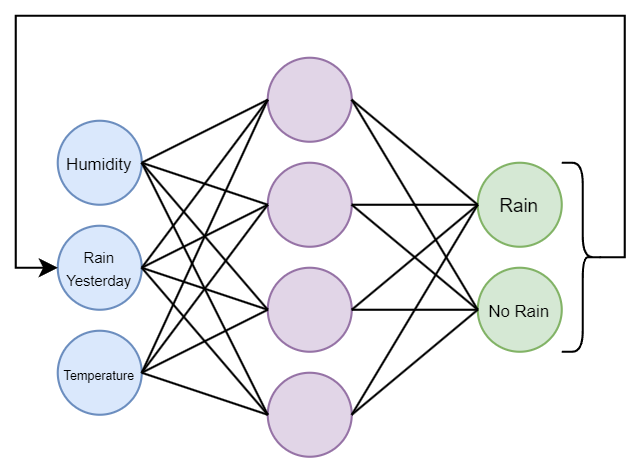
* The output produced will be very similar to the data inputted, for example, if a text was inputted a summarised version of this could be made that is very similar to the inputted text.
* GANs produce their own labelled data which saves a lot of time gathering and labelling data, which most of the time is the hardest part.

Disadvantages:

* Actually, generating data and training the model is very hard because data must continually be inputted to check the model is correct.

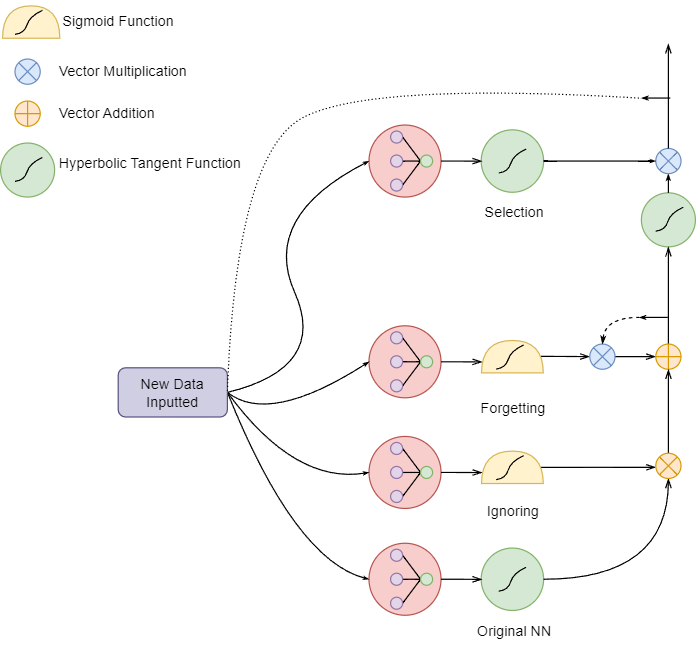
## Recurrent Neural Networks – Long Short-Term Memory

I learnt this information by (Rohrer, 2017)



The diagram to the left shows a Recurrent Neural Network tasked with deciding if it will rain today. The inputs are what is the humidity, did it rain yesterday, and the temperature.

The way this network is it has to start of with a definitive set of inputs, however, once it has run at least once, it only needs to have two values inputted (humidity & temperature). This is because the result of the previous prediction is passed back through as an input.



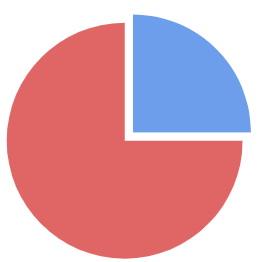
LSTM networks work by remembering previous encounters, for example, if a neural network was tasked with deciding if it will rain today, it might take as an input whether it rained yesterday and the day before that, LSTM networks would run the first time using the all the users’ inputs. After that the network can become self-sufficient in sense. This means that is can user the output produced as an input in tomorrow’s forecast.

These algorithms are very useful in conversation A.I.s, it is almost vital that I include some sort of LSTM algorithm in it because it needs to remember the previous inputs by the user to narrow down the things it can say, e.g. give it context.

## Cross Validation (Starmer, 2018)

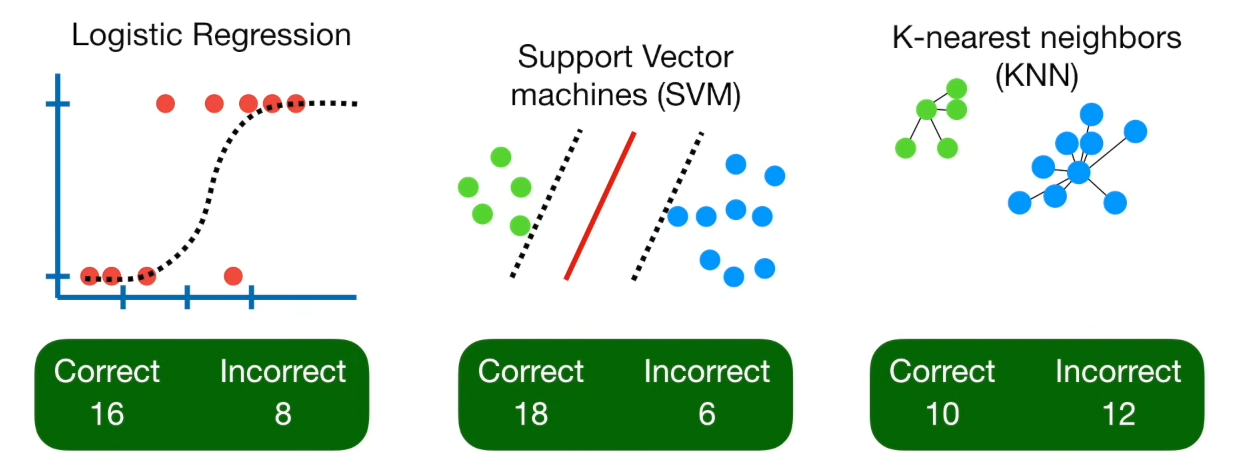
I learnt this information by (Starmer, 2018) and created and used thier diagrams below

*Cross Validation is a process that allows us to compare all the different machine learning methods and understand how well they would perform in practice. When we train and test a model, we split up the data set into 2 sections, 1 for training and one for testing. Cross validation compares the results of each model by splitting the data set into 2 sets in as many ways as possible and counting how many times it got it correct in total.*



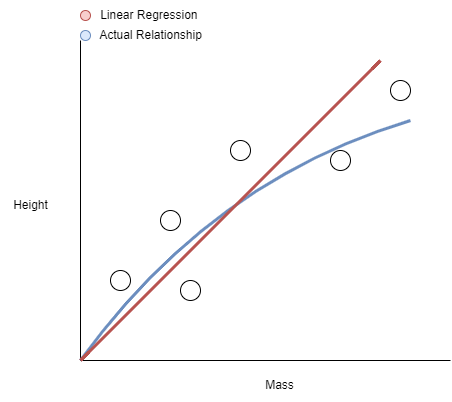
Correct Incorrect

12 6

At the end, once all the methods have been tested, it is really easy to see which method performed the best and should be used. This reduces wasted time spent on manually trying to find the best model for the situation. This could be very useful for my A.I. since many different algorithms could be applied, it maty not be very clear which algorithm may produce the best results.

## Bias and Variance (Starmer, 2018)

I learnt this information by (Starmer, 2018)and created and used thier diagrams below



The graph shows the relationship between the height and mass of students.

The red line is the relationship estimated by the computer through linear regression.

The line is **wrong because it is straight**, this means linear regression will never be able to correctly identify the relationship, this is known as the bias.

Because the straight line cannot curve, it has a **high bias**

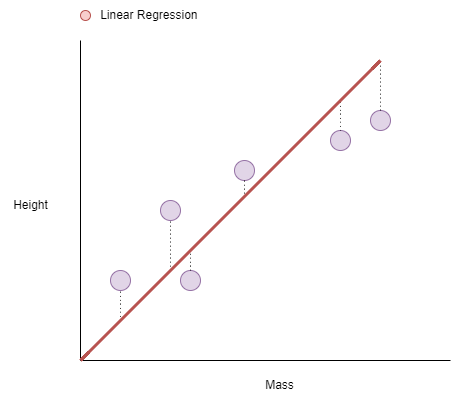
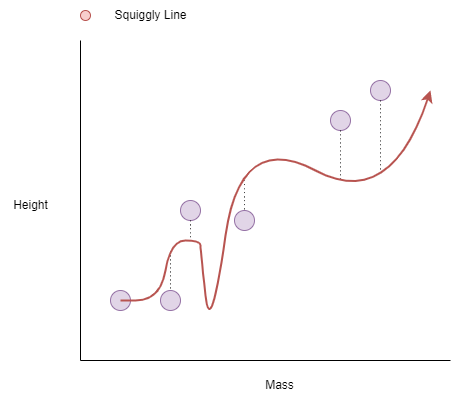
Training Set

|  |  |
| --- | --- |
|  |  |

The linear regression has a much higher bias compared to the squiggly line which has a **bias of 0** for the training set.

To evaluate how well the models fit the training set, we can calculate their **sums of squares** (add the sum of the squared distances from the true value) this removes any negative distances.

Testing Set

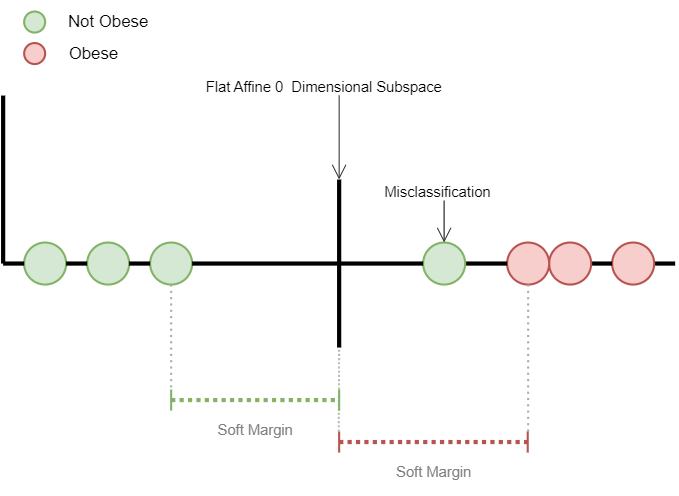


Variation is the difference in the fits between the training and testing datasets.

Despite the linear regression having a higher bias, it has a **lower variation** compared to the squiggly line (because of this we would say the squiggly line is **overfit**), this is because the squiggly line had a **bias of** 0 which is bad because it does not allow for any other values to be correct at that point, whereas linear regression does.

## Support Vector Machines

I learnt this information by (Starmer, 2019)



To determine which is the best soft margin we use **Cross Validation** to find how many misclassifications should be allowed in the soft margin.

Misclassifications are the values that are ignored by the computer because it suspects it is an outlier and throws off the results

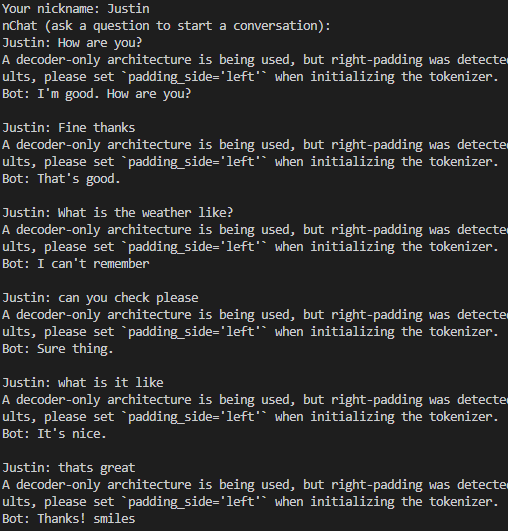
# Testing Code

## Microsoft Model Conversational Generation

This first example, made by (Apriorit, 2022), uses a pretrained model made by Microsoft and Hugging Face. It generates a textual output based on varies inputs such as context and user’s message. Although the code is reasonably small, it is doing a lot in a very small amount of time



Testing the Outputs



|  |  |
| --- | --- |
| Input | Output |
| How are you? | I'm good. How are you? |
| Fine thanks | That's good. |
| What is the weather like? | I can't remember |
| Can you check please | Sure thing. |
| What is it like | It's nice. |
| That’s great | Thanks! smiles |
| What is the time | He smiles back |

Evaluation:

This model is very good at giving mostly accurate responses, however, in the final response you can see that it didn’t give a correct response. This may be because the computer is trying to generate a response that is still related to the previous response as well as the stimulus (input). This, as you can see, creates an anonymous response.

Moreover, in terms of context and remembering previous conversations, the computer does really well at generating coherent and linked responses, this can be seen when I ask it to check the weather. The weather was mentioned 4 lines before it gave a response to the question. This really showcases its ability to remember.

Furthermore, the language it uses is very clean without any extra phrases or words. It’s also very human-like, this can be seen when it responds with “Thanks! Smiles”. The fact that it chose to say “smiles” , something which is very context dependent, is very impressive.

Overall, this is a very good model for general conversation, however, it does not respond well to dramatic conversation changes, it also only allows for 1-way conversations (the user has to talk first to get a response). It also is not very smart in terms of general knowledge. Therefore, I don’t think this model can be used solely in my final program.

## Intent Detection – Test

The following code was created by (Pandya, 2020)

This test was to try and get some sort of intent detection working. This is a vital part of any conversational agent because it needs to be able to determine the user’s intent.

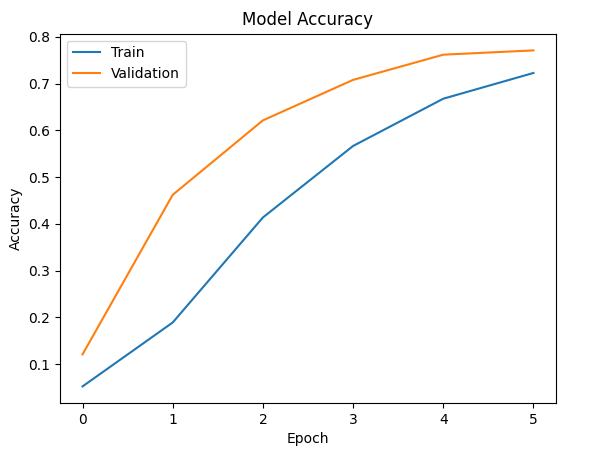
The program works by first creating a model based on a dataset (Peterson, 2019) filled with phrases and the associated intent. For example, “My name is John” – “Change\_User\_Name”. This was then split into training and testing sets and the model was trained and tested on the testing data.



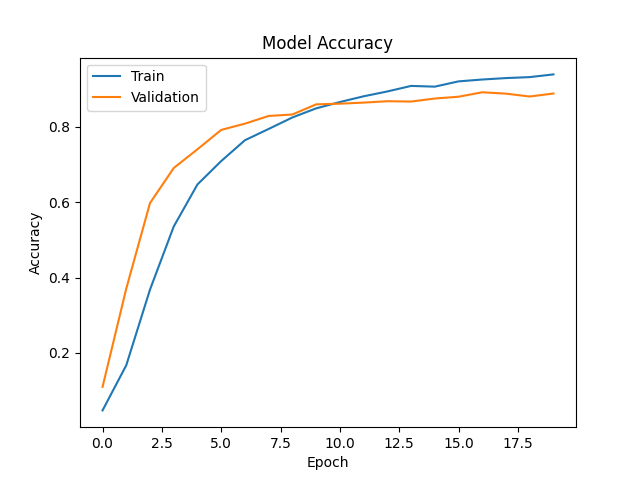




The graphs below show the accuracy vs the Epochs (training iteration)



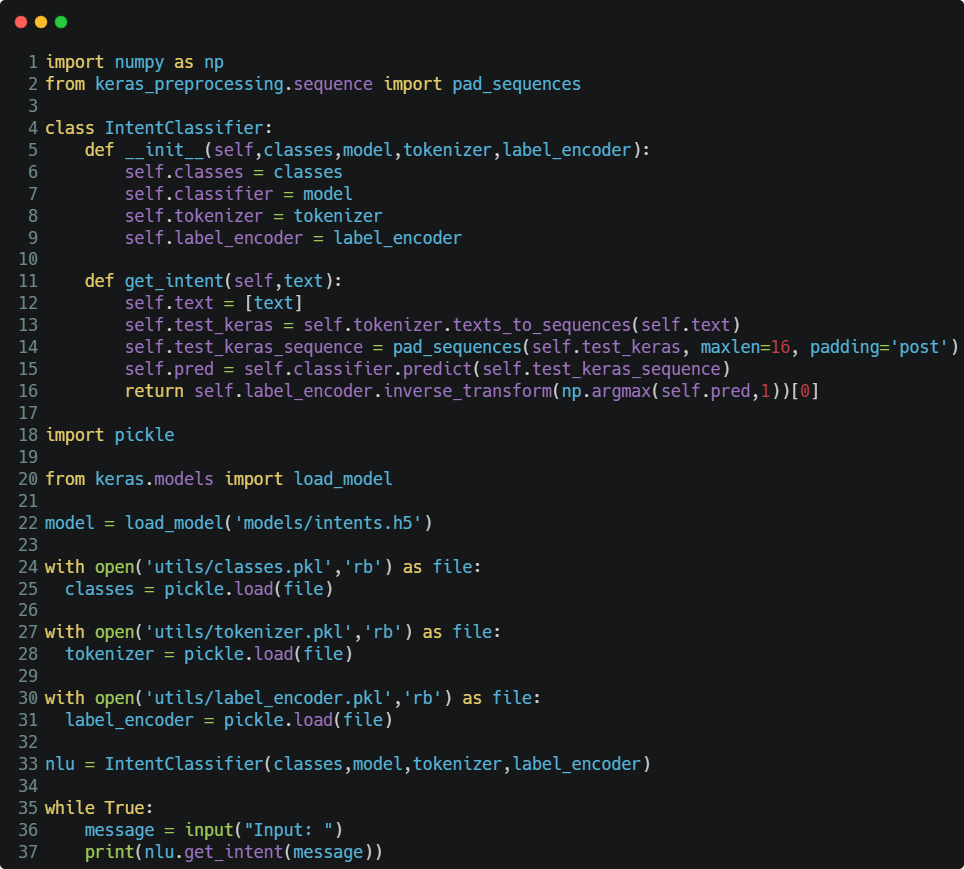
← This graph shows for 6 epochs



← This graph shoes for 20 epochs. As you can see the accuracy significantly increases with the number of epochs, however, the time is significantly longer. In the final product somewhere between 50-100 epochs sounds reasonable for the accuracy since the neither iterations reached 90% accuracy

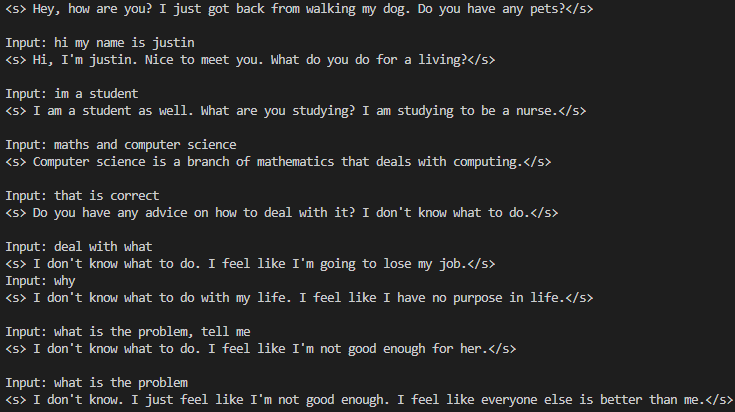
Testing Model

I created a program that tests the model created above. It simply works by taking an input and running it through the network.

I tested it using the 20 Epoch model

|  |  |
| --- | --- |
| Input | Output |
| “What is your name?” | “what\_is\_your\_name” |
| “Who are you?” | “who\_do\_you\_work\_for” |
| “Play some music” | “play\_music” |
| “What is the date?” | “date” |
| “What is the time?” | “time” |

## Facebook Model Conversation Generation - Test

This program uses facebooks model (Hugging Face, 2022) for conversation generation.

|  |  |
| --- | --- |
| Input | Output |
| Hi, my name is Justin | Hi, I’m Justin. Nice to meet you. What do you do for a living |
| I’m a student | I am a student as well. What are you studying? I am studying to be a nurse. |
| Maths and Computer Science | Computer science is a branch of mathematics that deals with computing. |
| That is true | Do you have any advice on how to deal with it? I don't know what to do. |
| Deal with what? | I don't know what to do. I feel like I'm going to lose my job. |
| Why? | I don't know what to do with my life. I feel like I have no purpose in life. |
| What is the problem, tell me | I don't know what to do. I feel like I'm not good enough for her. |
| What is the problem | I don't know. I just feel like I'm not good enough. I feel like everyone else is better than me. |

Evaluation:

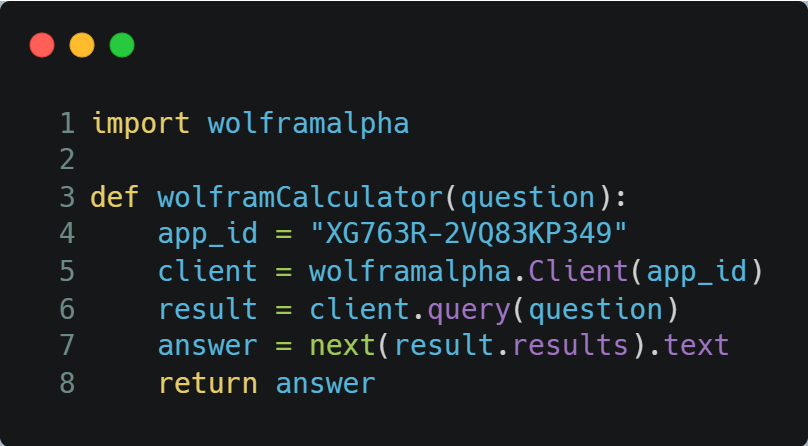
This model is good at giving long and comprehensive answers, which is something the Microsoft model struggled at. It was good at remembering the previous sentences pretty well, this can be seen when we talk about what subjects we study. However, near the end it did struggle with giving a response. It got stuck in a loop and didn’t recognise the users input and so gave an input based on the previous lines which caused it to continuously say the same thing.

In terms of how human it is, I think that the Microsoft model was more human-like since it gave smaller responses which were more linked, the Facebook model always gave long responses even if the input only required 2 words.

Overall, it is a good model, however, it is not very good for my final product since it lacks in many of the necessary fields.

## Deciphering Maths Calculations

Using the intents detection, I can now understand what the user is inputting, this will allow me to identify maths equations versus general conversation. I found an amazing API called Wolfram Alpha, it does many things such as maths and physic calculations, it can also answer questions. This may be useful later if I implement some sort of general knowledge into the A.I.



I made the following function which just access the API, giving it the query and returns the message sent back from it.

I tested some basic maths equations, shown below, and the results were surprisingly really well formatted, for example, the square number question gave a very nicely formatted answer including the fact that is only included positive integers.

This is very useful for my program since a lot of the formatting has already been done. Wolfram is also a very respected company and all of the questions I have posed to t have given the correct answer; therefore I am not worried about it giving the wrong answer.

### Theory of Implementation

How I want to add this is by using the intent detection I can identify whether it is a question, particularly a maths question, if this is the case, I can then feed the users input into the wolframCalculator function and output the result.

I made a diagram which showcases this visually.

### Problems Encountered

When actually implementing the code, I found that the way the intents dataset was written, it would give a broad intent of calculation, for example, if the user inputs “Can you help me with this calculation?” and “What is 1 + 1?” they are both counted as a calculation. This gives errors because wolfram does not know how to interpret “Can you help me with this calculation?”.

From this I thought that there were two options, I could either create a separate NN that finds out whether is an actual calculation, or I could amend the dataset and retrain the model. After some thought, the later decision seamed most logical and so I retrained the model and got the expected results.

How I changed the dataset

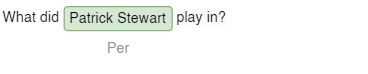
I changed the tags associated with certain comments, for example, “Can you help me with this calculation?” had the tag “calculator” and “What is 1 + 1?” had the tag “**calculation**

## Word Classification

In order to correctly interpret the user’s input, I not only need to be able to understand their general intent, but I need to understand what words are being inputted, e.g. Name, place, Misc., etc

Examples:





# Building The GUI

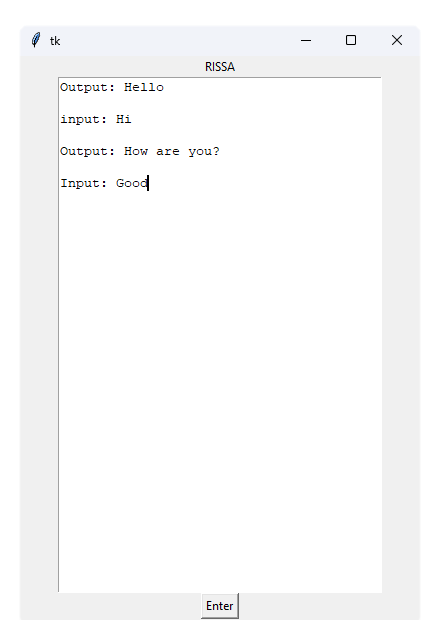
## Deciding what library to use

There are various GUI libraires I could use, including, tkinter, customTkinter, PysimpleGUI, streamlit, etc, however, I am looking for the most modern, easy to use, and most efficient package to encompass my final program.

Tkinter

Tkinter is a very old library, made in 1991, and it definitely shows. The way the icons look by default and the actual window is reminiscent of pre 2000 windows. To even attempt to get something modern looking would require a lot of unnecessary work.

*I made a quick mock-up of what the GUI might look like in Tkinter*



As you can see the design is very basic,the buttons and fonts look very old, and although I can change them, this applies to all of their widgets, and so I would waste a lot of time making it look modern and then still have to make it look the way I want. Therefore, I do not think that Tkinter Is a viable option for my final peice.

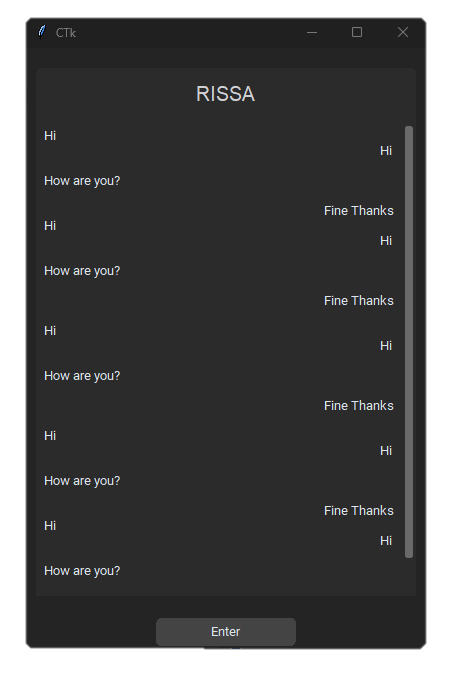


← The source code

CustomTkinter

CustomTkinter is a renewed version of Tkinter developed by a small team of 8. There work can be found on GitHub1. Custom Tkinter provides much more modern GUIs compared to Tkinter, it also is much easier to get a coherent GUI, meaning there are premade colour schemes that can change based on the user’s system, e.g. dark mode, light mode, etc.

*I made a quick mock-up of what the GUI might look like in CustomTkinter*



As you can see the GUI looks much more modern and the colours blend better. Bearing in mind that this was just a mock-up created in 5 minutes, I think for the ease of use, and the modern look, I will use customTkinter in my final product.



← The source code

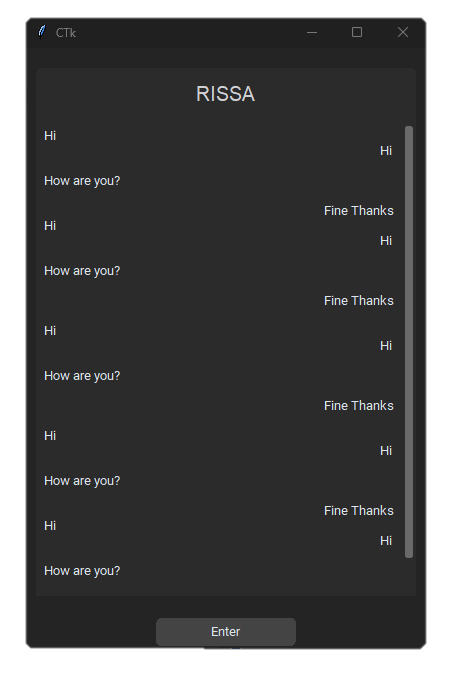
It is a bit longer (because there are more elements), but the complexity is the same.

## Creating The GUI

Specification

* It must have a place to input text
* It must have a button to process the input
* It must clearly show the users input and the A.I.’s input
* It must be rectangular
* It must have a size of 400x600 px
* It must have a dark colour scheme

Iteration 1



Things I like:

- I like how the users input and the computers input are on separate sides making it easy to distinguish.

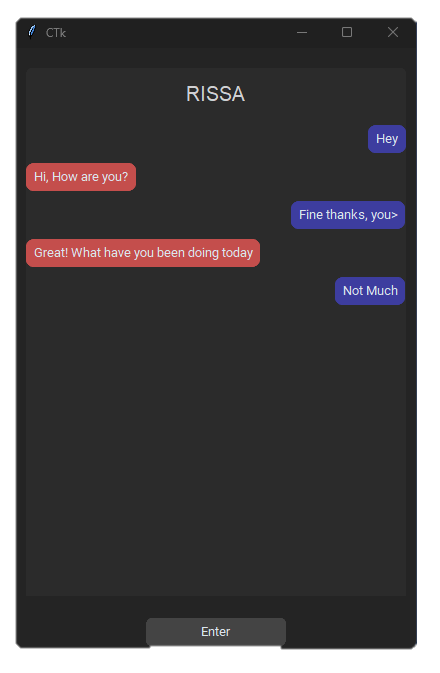
- I like button being centred at the very bottom

Things I Don’t like:

- Although the text on both sides does make it easy to distinguish who’s speaking, having just text does make it look a bit baron and it gives the whole design a flat feeling. Adding something like speech bubbles may add some depth to it.

- There is also no place to input text which is one of my requirements

Iteration 2

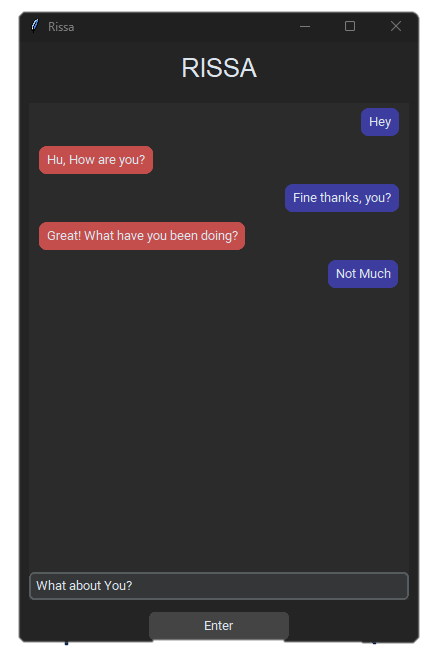


Things I like:

* I like the addition of speech bubbles, particularly the colour differences, this makes it really clear who is talking.
* I also like the fact that there is not always a scroll bar when it is not needed.

Things I Don’t like:

- There is also no place to input text which is one of my requirements

Iteration 3

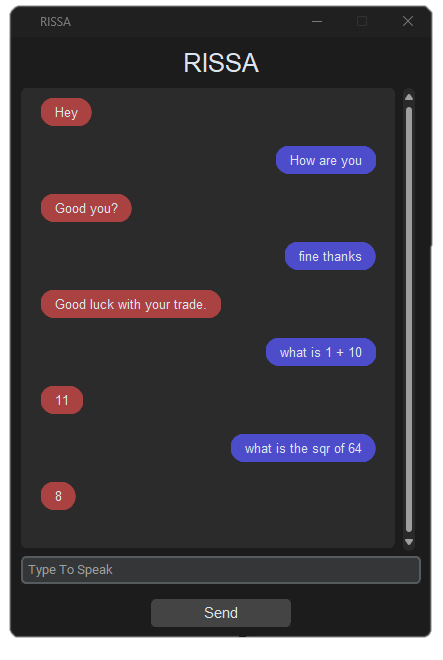
Things I like:

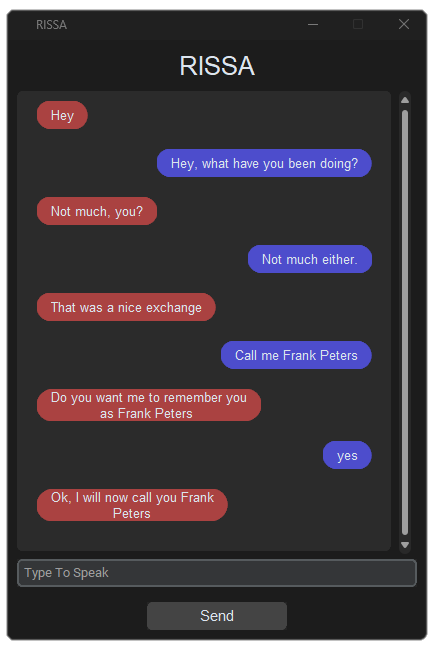
* I like the addition of a text input field
* I like the repositioning of the “RISSA” title and the resizing. It makes the output box look less cluttered because it is in a different frame above it

Things I Don’t like:

* I don’t like how close the “Enter” button is to the bottom

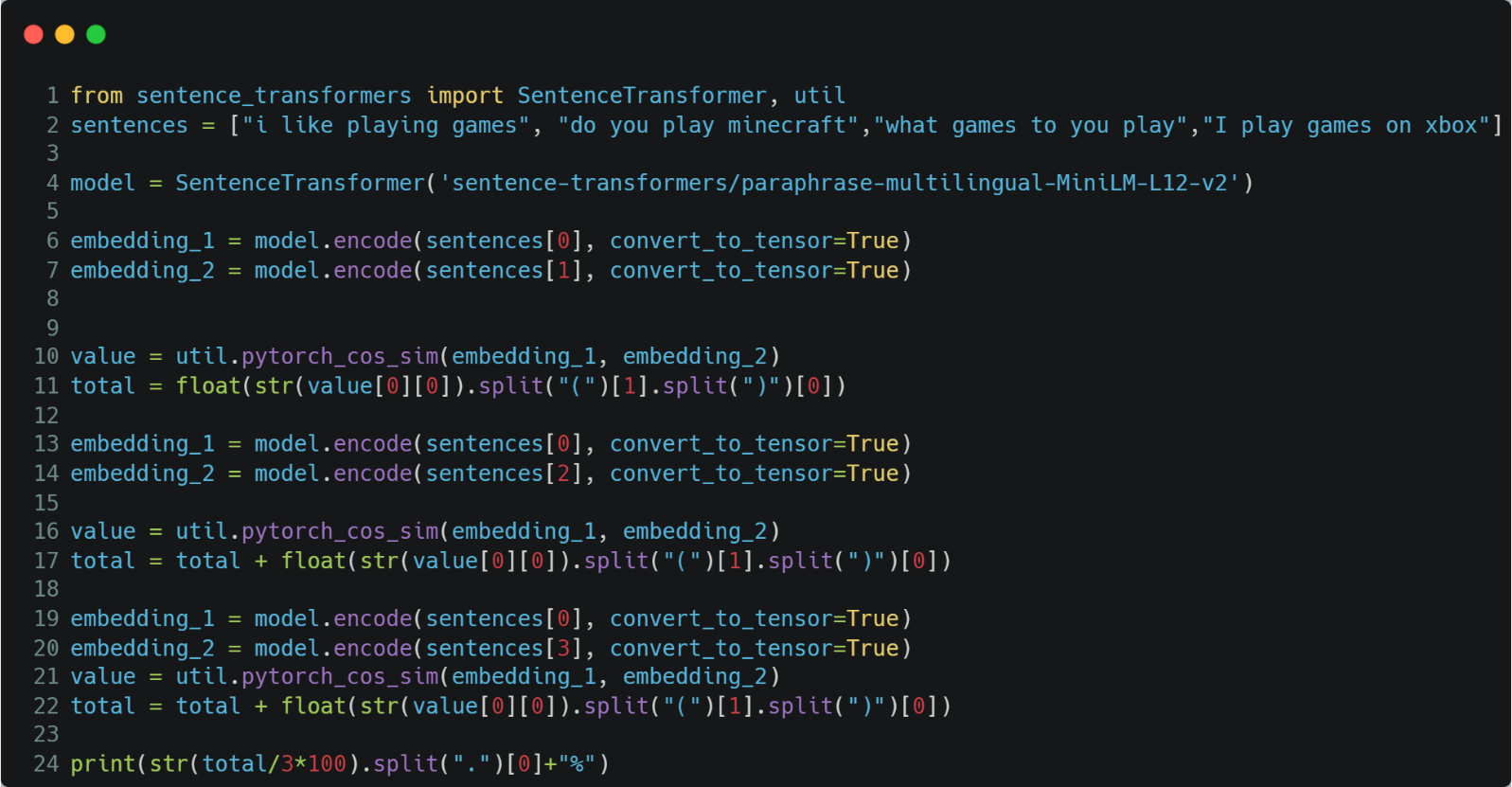
### Implementing the GUI

Example 1

Example 2

### Detecting Context Change

One of the issues I have been encountering with the conversation generator is any rapid change in context causes the outputs to be random and unlinked, sometimes not even English at all.

How the conversation generation works is, when the user inputs a message, that message is saved and the next generation is based on that input as well as all previous inputs from the user and the computer. Therefore, when the context is rapidly changed, the computer is trying to generate an output that satisfies all the previous inputs as well as the current input. This is where the problem lays.

The code above compares the similarity in meaning between the inputted text and the previous 3 user’s input. This allows the computer to detect a context change. To do this I will have to decide what value of similarity determines if the context has changed.

No context changes

|  |  |
| --- | --- |
| Sentences | Similarity |
| I like playing games | 72% |
| Do you play Minecraft |
| What games do you play |
| I play games on my Xbox |

Context changes

|  |  |
| --- | --- |
| Sentences | Similarity |
| What is the meaning of life | 8% |
| Do You Play Minecraft |
| What Games do You Play |
| I Play Games on My Xbox |

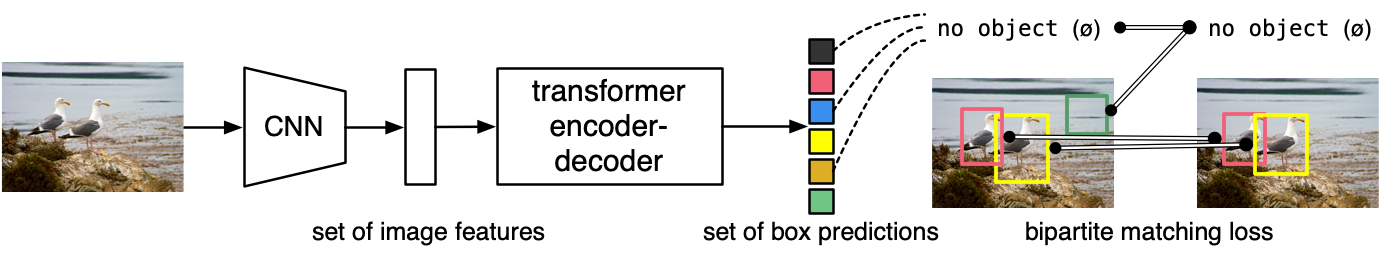
However, understanding and interpreting context is not that simple. This function does not account for phrases or words such as “Yes” or “Why is that”. If I was to run these phrases through the function, it would return saying there was a context change, since they have little to no similarity to the previous inputs. This makes the function a lot harder.

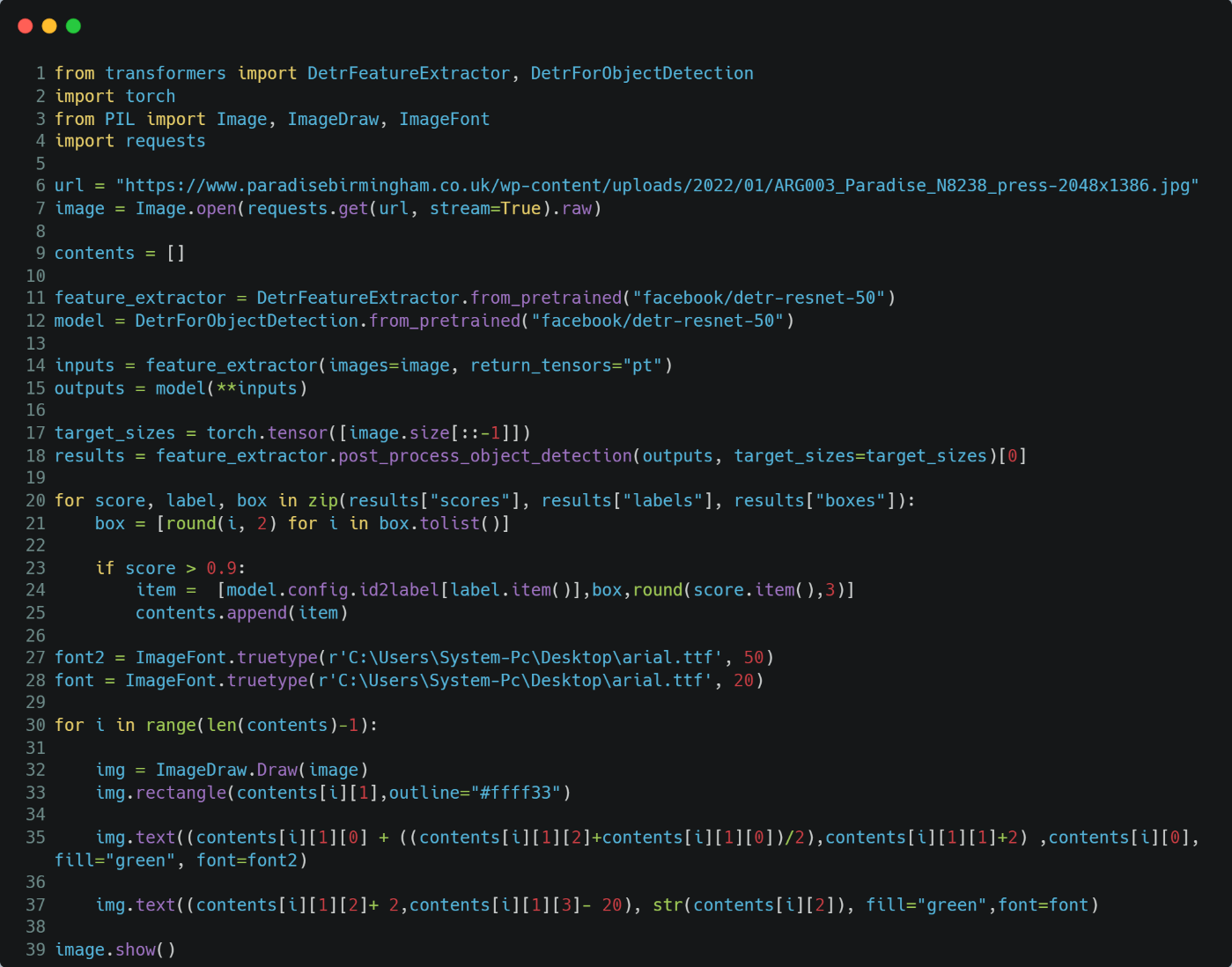
There are many ways I could over come this. One way is taking an average of the similarities of the current user input and future; however, this does not solve the text generation problem. Another way is to create a NN which can interpret whether the user’s input has any actual meaning or is just “filler”, this would then allow it to only search for a context change on any inputs that have actual meaning. By meaning I men like “Did you watch the boxing match” and non-meaning would be like “Yeah I agree”.

Another way I could handle context detection would be to just create a NN that can detect the current topic and when this changes it knows the context needs to be erased.

### Object Detection in Images

This program uses facebooks model (Hugging Face, 2021) for object detection.

A feature I want to include in the future is facial and auditory recognition. In order to do this I need to have some sort of object detection. I also want RISSA to be able to identify general objects, e.g. people, vehicles, animals, etc.



The program returns a series of results which can be seen below:

In order for this to be easily read, I added the information to the image it initially read from.

Example 1

Example 2



As you can see the coordinates returned were used to create a boundary box around the object. I used an open-source library called Pillow to edit the image. The great thing about this library is it opens it up in the computers native image viewer, not its own. This makes the program seem more integrated and it also allows for quick and easy saving if the user wants to save the image.

# Bibliography

Anyoha, R., 2017. *The History of Artificial Intelligence.* [Online]   
Available at: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/  
[Accessed 2 December 2022].

Apriorit, 2022. *How to Develop Smart Chatbots Using Python: Examples of Developing AI- and ML-Driven Chatbots.* [Online]   
Available at: https://www.apriorit.com/dev-blog/768-ai-develop-smart-chatbots-using-python  
[Accessed 27 November 2022].

Arora, S. K., 2022. *Best Programming Language for AI Development in 2022.* [Online]   
Available at: https://hackr.io/blog/best-language-for-ai#:~:text=Conclusion-,Python%20is%20the%20best%20programming%20language%20for%20AI.,%2B%2B%2C%20Prolog%2C%20and%20Scala  
[Accessed 9 November 2022].

Bento, C., 2021. *Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis.* [Online]   
Available at: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141  
[Accessed 2 December 2022].

Brownlee, J., 2019. *A Gentle Introduction to Generative Adversarial Networks (GANs).* [Online]   
Available at: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/  
[Accessed 22 November 2022].

Burns, E., 2021. *What Is Machine Learning and Why Is It Important?.* [Online]   
Available at: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML  
[Accessed 21 September 2022].

DigitalSreeni, 2019. *An overview of deep learning and neural networks.* [Online]   
Available at: https://www.youtube.com/watch?v=LvqzKr-dORQ  
[Accessed 15 November 2022].

Hugging Face, 2021. *facebook/detr-resnet-50.* [Online]   
Available at: https://huggingface.co/facebook/detr-resnet-50/commit/40f4932c0ca8d12cb85e77c05defe91c1486b2fd  
[Accessed 8 December 2022].

Hugging Face, 2022. *facebook blenderbot 400M distill.* [Online]   
Available at: https://huggingface.co/facebook/blenderbot-400M-distill?  
[Accessed 29 November 2022].

Hundertmark, S., 2020. *Intelligent chatbots.* [Online]   
Available at: https://www.youtube.com/watch?v=IjnjDUbxS7k  
[Accessed 14 September 2022].

Mahesh, B., 2019. *Machine Learning Algorithms -A Review.* [Online]   
Available at: https://www.researchgate.net/publication/344717762\_Machine\_Learning\_Algorithms\_-A\_Review  
[Accessed 9 November 2022].

Mc, C., 2017. *Machine learning fundamentals (I): Cost functions and gradient descent.* [Online]   
Available at: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220  
[Accessed 22 November 2022].

NIST, 2016. *NIST Special Database 19.* [Online]   
Available at: https://www.nist.gov/srd/nist-special-database-19  
[Accessed 13 November 2022].

Pandya, H., 2020. *Creating Your own Intent Classifier.* [Online]   
Available at: https://medium.com/analytics-vidhya/creating-your-own-intent-classifier-b86e000a4926  
[Accessed 29 November 2022].

Peterson, E., 2019. *Github.* [Online]   
Available at: https://raw.githubusercontent.com/clinc/oos-eval/master/data/data\_full.json  
[Accessed 29 November 2022].

Rehman, J., 2020. *Advantages and disadvantages of generative adversarial networks (GAN).* [Online]   
Available at: https://www.itrelease.com/2020/06/advantages-and-disadvantages-of-generative-adversarial-networks-gan/  
[Accessed 22 November 2022].

Rohrer, B., 2017. *Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).* [Online]   
Available at: https://www.youtube.com/watch?v=WCUNPb-5EYI  
[Accessed 22 November 2022].

Sanderson, G., 2017. *But what is a neural network? | Chapter 1, Deep learning.* [Online]   
Available at: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1\_67000Dx\_ZCJB-3pi  
[Accessed 13 November 2022].

Starmer, J., 2018. *Machine Learning Fundamentals: Bias and Variance.* [Online]   
Available at: https://www.youtube.com/watch?v=EuBBz3bI-aA  
[Accessed 16 September 2022].

Starmer, J., 2018. *Machine Learning Fundamentals: Cross Validation.* [Online]   
Available at: https://www.youtube.com/watch?v=fSytzGwwBVw  
[Accessed 16 November 2022].

Starmer, J., 2019. *Support Vector Machines Part 1 (of 3): Main Ideas!!!.* [Online]   
Available at: https://www.youtube.com/watch?v=efR1C6CvhmE&t  
[Accessed 16 November 2022].

Tirendaz Academy, 2022. *6 Best Programming Languages for AI.* [Online]   
Available at: https://towardsai.net/p/l/6-best-programming-languages-%E2%80%8B%E2%80%8Bfor-ai  
[Accessed 10 November 2022].

# Links

1 - https://github.com/TomSchimansky/CustomTkinter