Developing an Intelligent Chatbot: the First Interim Report

Group xx: Daniel Willis, Charlotte Anderson and Brandon Gous

January 14, 2022

Abstract

This interim report presents (1) the outline of our coursework report, (2) some initial descriptions of the requirements of the coursework, the methods, programming languages, packages, tools that have been identified so far, and (3) an initial work plan.

1 Introduction

For this coursework we are developing a chatbot to help customers in finding the cheapest available ticket for their chosen journey also to improve customer service satisfaction by applying some appropriate AI techniques.

1.1 Background and Motivation

A bit background information on chatbot in general and the coursework specification(Wang 2018).

1.2 Aim and Objectives of this coursework

You may rephrase the the aim and objectives from your point of view.

1.3 Difficulties and Risks

List as many as you can identify.

1.4 Work Plan

2 Related Work

Review some similar chatbot systems. (Write as much as you have now.)

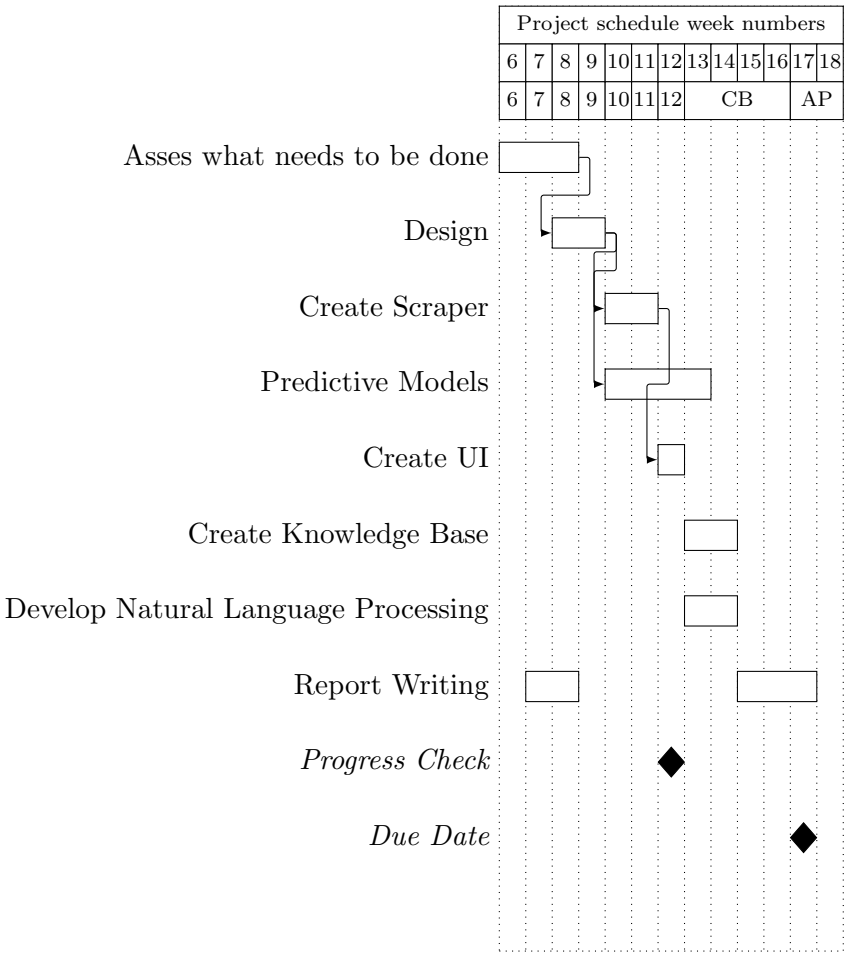
3 Methods, Tools and Frameworks

In this section, you should describe the methods, programming languages, packages, tools and framework you plan to use. for this report, you can list some you have identified and intend to use. No need to give any details.

3.1 Methods

You may list some methods you will use for developing your chatbot, including

Such as what type of user interface (graphical, text, or voice, etc) you intend to use.
What Natural Language Processing and understanding methods you intend use,
What referring or reasoning methods
What prediction methods, such as kNN, neural networks etc.



Project Gantt chart

3.2 Languages, Packages, Tools

On programming language: using Python or Java, or others.
Packages: for NLP, use NLTK(Bird & Klein 2009), or others,
For KnowledgeBase and Engine: PyKE or PyKnow, or others.
For Database: e.g, Postgres, or MongoDB

3.3 Development Framework

4 Design of the Chatbot

4.1 The Architecture of the chatbot

You may draw a functional diagram if you like.
You can describe your design for each key module or component of your chatbot, in a subsection. E.g.

4.2 User Interface

4.3 NLP

4.4 Knowledgebase

4.5 Inferring Engine

4.6 Delay Prediction Models

For our prediction models we plan to use three. A Bayesian model that calculated the probability of the train being late to the second station given that the train is delayed to the first station. A neural network that uses the delay and the normal time it takes between the stations to predict how late the train will be. And K nearest neighbour which predicts how late the train will be to the second station.

4.6.1 Bayesian

Bayesian Theorm is that the probability of event A happened given that B happened can be calculated by the probability of event B happening given that event A happened multiplied by the probability of event A happening all divided by the probability of event B happening.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Additionally this can be expanded so that the probability of event B happening doesn't need to be known. The probability of event B happening is equal to the probability of event B happening given event A happened multiplied by the probability of event A occurring plus the probability of event B happening given that event A does not multiplied by the probability that event A does not happen.

$$P(B) = P(B|A) * P(A) + P(B|'A) * P('A)$$

This means that you can calculate the probability of event A happening given that B happened from the probability of event A happening, the probability of event B happening given event A happened and the probability of event B happening given that event A didnt happen.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B|A) * P(A) + P(B|'A) * (1 - P(A))}$$

I applied this to the prediction of trains being late by making A the event of the train being late to the second station and B the event of the train being late by delay or more to the first station.

4.6.2 Neural Network

A neural network is a collection of layers of neurons, each neuron holds a value known as activation of that neuron. There can be an unlimited number of layers and neurons and they can be used to solve very complex problems such as object recognition.

Each neuron is connected to all the neurons of the previous layer through numbers called weights (referred to as W) and a bias (referred to as b). The weights are organized in the form of a matrix of shape (no of units in current layer, no of units in previous layer). This means that to calculate a neuron on the next layer you need all the previous values.

If this is applied to our problem we chose to have two input nodes of the delay at the first station and the normal time taken between the stations and obviously the output node for the prediction. This is shown in figure 1 it shows all the neurons in the simple neural network we designed to solve the problem with two hidden layers of three neurons each.

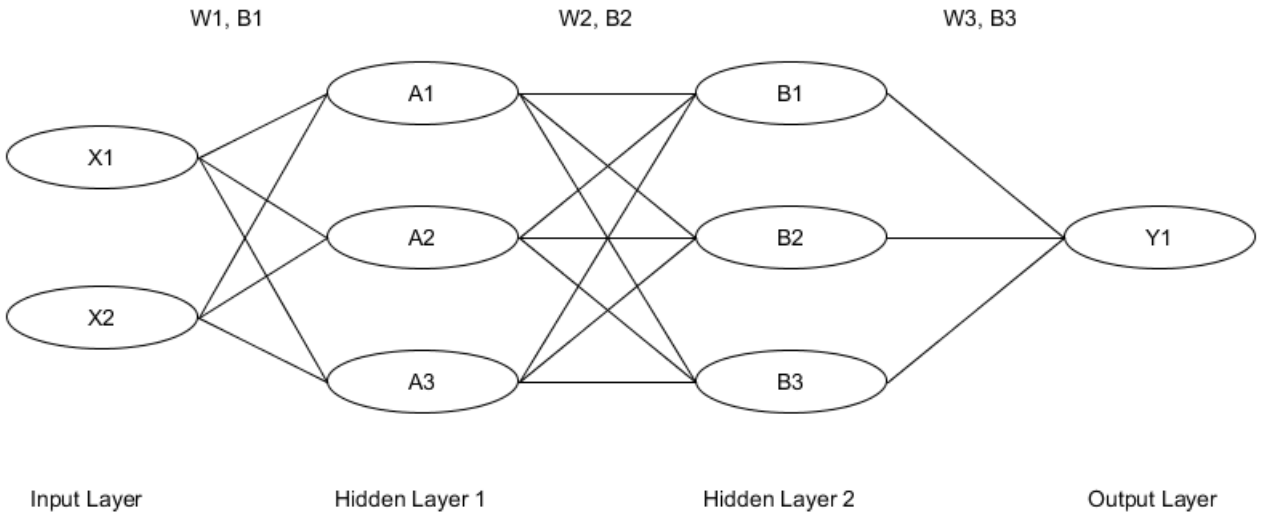


Figure 1: Diagram showing the neural network

This basically means that W_{ij} refers to the weight of the connection from i'th neuron in current layer to j'th neuron in previous layer. The biases are organized in the shape of (no of units in current layer, 1) so B_i corresponds to the bias of the i'th neuron in current layer.

Each neurons value can be calculated by:

$$A1 = g(f(X1, X2)) = g(W1_{11} * X1 + W1_{12} * X2 + b1_1)$$

$$A2 = g(f(X1, X2)) = g(W1_{21} * X1 + W1_{22} * X2 + b1_2)$$

$$A3 = g(f(X1, X2)) = g(W1_{31} * X1 + W1_{32} * X2 + b1_3)$$

$$B1 = g(f(A1, A2, A3)) = g(W2_{11} * A1 + W2_{12} * A2 + W2_{13} * A3 + b2_1)$$

$$B2 = g(f(A1, A2, A3)) = g(W2_{21} * A1 + W2_{22} * A2 + W2_{23} * A3 + b2_2)$$

$$B3 = g(f(A1, A2, A3)) = g(W2_{31} * A1 + W2_{32} * A2 + W2_{33} * A3 + b2_3)$$

$$Y = f(B1, B2, B3) = W3_{11} * B1 + W3_{12} * B2 + W3_{13} * B3 + b3_1$$

Where $g(x)$ is the activation function. Being as this is a regression problem and not classification I will use $g(x) = max(0, x)$ more commonly called the ReLU. This returns 0 when x is negative and x when positive. Due to it's lower saturation region, it is highly trainable. These equations can be vectorised using the dot product of the matrices:

$$A = g(W1.X + b1)$$

$$B = g(W2.A + b2)$$

$$Y = W3.B + b3$$

However for the neural network to become more accurate it has to learn by adjusting the weights an bias to bring the output closer to the target. This is done using a back propagation algorithm to calculate the gradients. These are the vectorized equations:

$$dY = \frac{1}{m} * (Y - T)$$

$$dW3 = \frac{1}{m} * (dY.W3^T)$$

$$db3 = \frac{1}{m} * \sum dY$$

$$dB = W3^T.dY$$

$$dW2 = \frac{1}{m} * (dB.W2^T)$$

$$db2 = \frac{1}{m} * \sum dB$$

$$dA = W2^T.dB * g'(W2.A + b2)$$

$$dW1 = \frac{1}{m} * (dA.W1^T)$$

$$db1 = \frac{1}{m} * \sum dA$$

Where T is the true value, m is the number of datapoints, $*$ is multiplication, $x.y$ is dot product of the matrices x and y and x^T is the transpose of matrix x. We do not need gradients with respect to the input layer neurons because we are not going to change them.

Now using these gradients calculated we can modify our weights and bias getting us closer to the true result.

4.6.3 K Nearest Neighbors

For the K nearest neighbour model we have to use load of data extracted from the database for each query. To do this I used a stored procedure to get the amount that all the trains was late to both stations specified.

The amount the train was late to the first station is then compared to the amount of delay that the user had at the first station using a distance function, in my case picked euclidean.

These distances are then sorted and you get the mean of the delay at the second station for the closest K nodes in the dataset. This is simple algorithm but heavily dependant on finding the correct K value.

4.7 Conversation Control

5 Implementation

5.1 Predictive Models

5.1.1 Bayes

To get these probabilities can use stored procedures to extract the frequencies from the database.

Firstly the probability of A (the train being late to the second station). This was easy enough to calculate I simply extracted the frequency count of all times it arrived at the second station. Totalled up the number of times the train was late and divided by the total pieces of data I had.

Secondly I calculated the probability of B given A. To do this I extracted the frequencies of delays at the first station if the train arrived late to the second station. From here I summed all the frequencies that the train was late to the first station by at least the delay.

Next I calculated the probability of B given not A. To do this I extracted the frequencies of delays at the first station if the train arrived on-time or early to the second station. From here I summed all the frequencies that the train was late to the first station by at least the delay.

5.1.2 Neural Network

The best way to implement all the weights and values being stored was through dictionarys with string keys eg: "W1" with multidimensional arrays as the value.

This meant that I could implement a solution which has the capacity to expand the number of layers and neurons per layer easily with simple for loops, making the solution highly customizable.

To keep track of how well the network was learning I would test every single datapoint through it every 1000 iterations and get the root mean squared error for them all. This was then plotted on a graph.

5.1.3 K Nearest Neighbour

To find the best K value a set up a test to run one thousand iterations of selecting random data points from the data. Then for each piece of data I would create a KNN object with a different value of K ranging 1 to 100 (inclusive). Get a prediction from each object and compare it to the target outcome. From this I calculated the root mean square error for each value of K over the iterations.

The results plotted on a graph is in figure 3 and the top ranking results are included in the table in figure 4. The shape of the graph is exactly what I expected to see with very few neighbours having a very high RMSE and then having a dip where the ideal K value is with the RMSE gradually increasing again as the more neighbours are used.

From these results I concluded that 9 was the best value for K not only because it ranks at the top but because 10 and 8 rank 3rd and 4th in the list. However any value from 5 - 15 would be very accurate still.

5.1.4 Comparison

I didn't think that the Bayesian model was very good it is more of a classification model not a regression therefore not particularly useful.

In figure 5 below I compare the neural network and k nearest neighbours over two million iterations of training the neural network.

This was accomplished by randomly selecting a thousand data sets, running these datasets through both the models and comparing the predictions to the targets. Then train the neural network on the same dataset for a hundred thousand iterations and repeat.

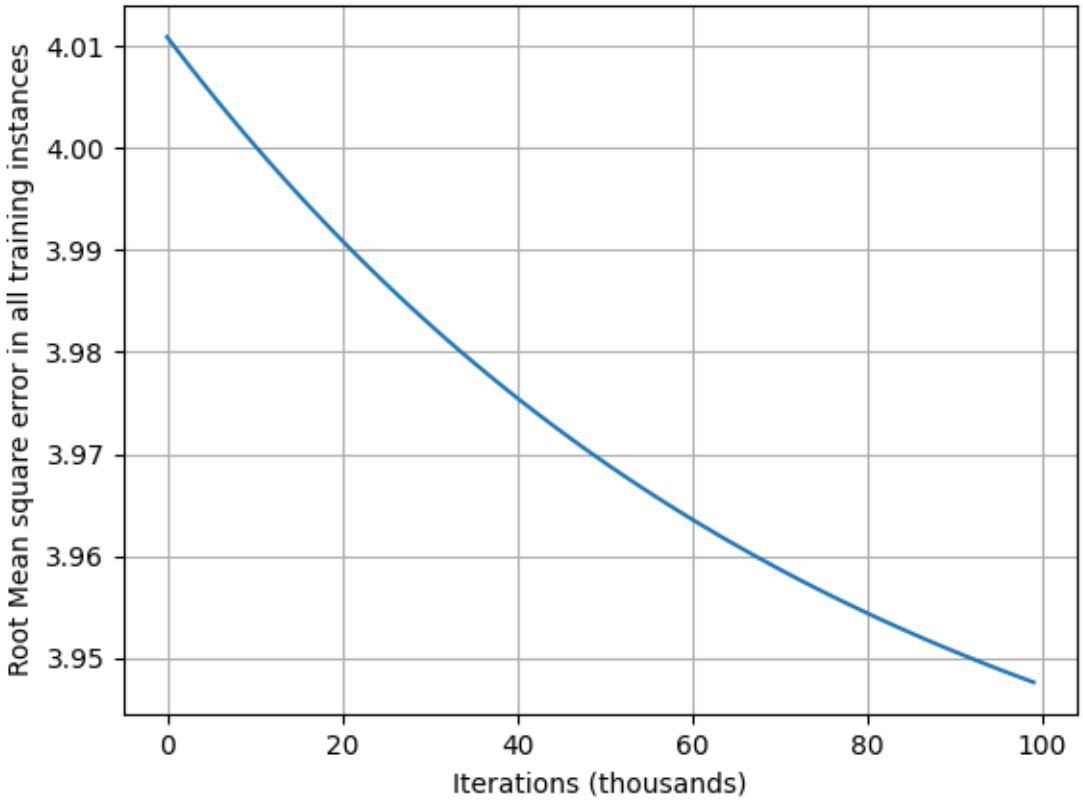


Figure 2: Comparison of neural network and K nearest neighbour

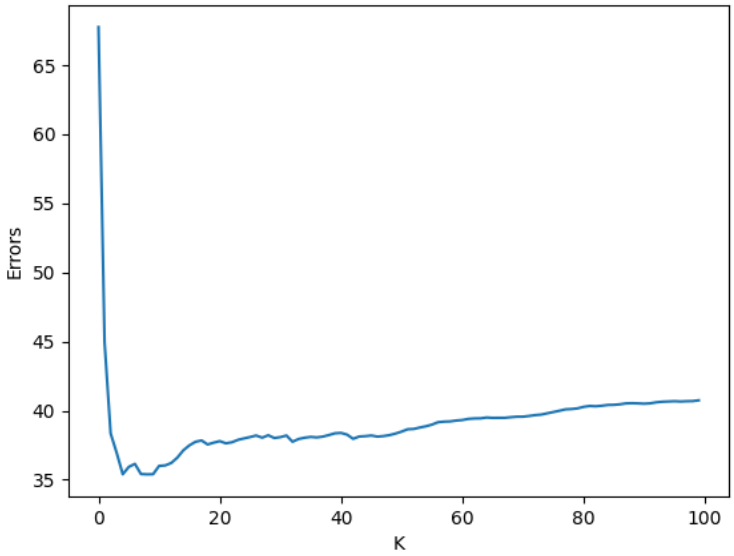


Figure 3: Searching for the correct K value

K	RMSE
9	3.538776543
5	3.539336
10	3.539625
8	3.54108125
6	3.592727778
11	3.600159504
12	3.60323125
7	3.614691837
13	3.620826627
14	3.659637245
4	3.6941
15	3.712949778
16	3.748832031
19	3.75489446
22	3.763205165

Figure 4: Sorted RMSE of K values

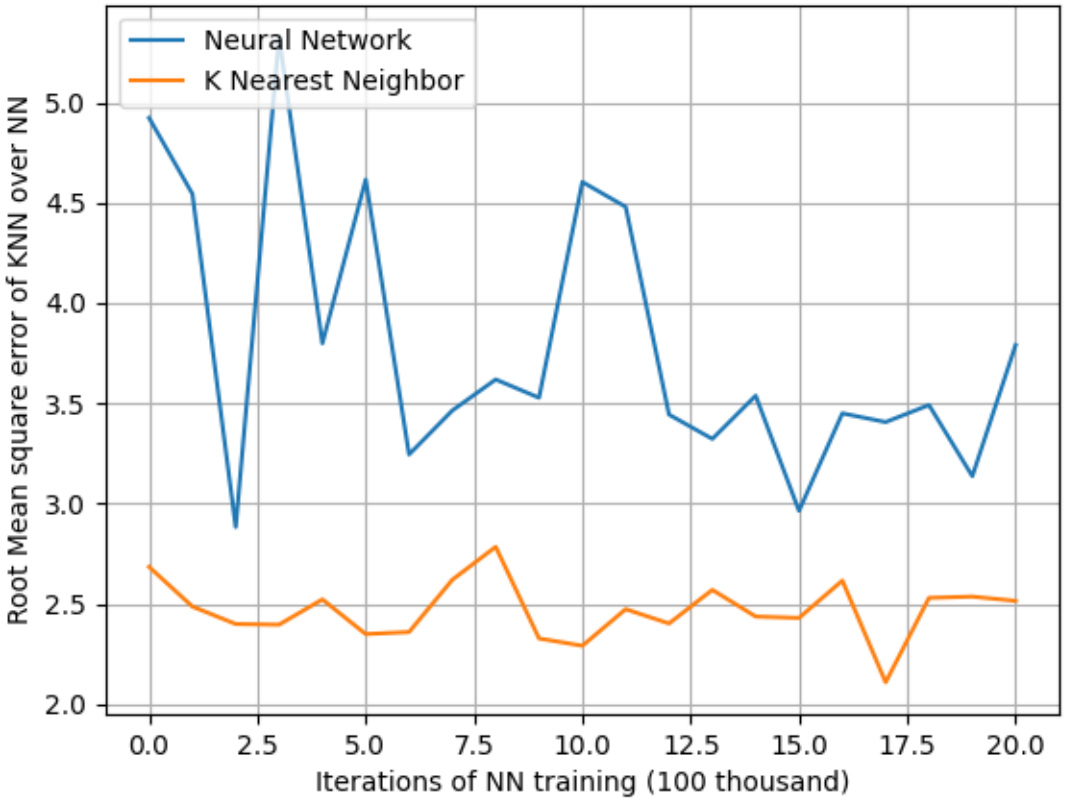


Figure 5: Comparison of neural network and K nearest neighbour

As you can see the K nearest neighbour algorithm is consistently better than the neural network despite the improvements made so we decided to use that.

6 Testing

6.1 Unit Testing

6.2 Integration Testing

6.3 System Testing

6.4 Userbility Testing

7 Evaluation and Discussion

8 Conclusion or Summary

References

Bird, Steven, E. L. & Klein, E. (2009), *Natural Language Processing with Python.*

Wang, W. (2018), 'Artificial intelligence modules (cmp6040, 7028) coursework specification'.