User ticketing system with automatic resolution suggestions

UNIVERSITAT POLITÉCNICA DE CATALUNYA (UPC) Barcelona Tech

Author

Aqib Hameed

Tutor

Lluís Padró

Director

Josep María Mirats

MASTER IN INNOVATION AND RESEARCH IN INFORMATICS

FACULTAT D'INFORMÁTICA DE BARCELONA (FIB)

Date of defense

26 April 2022

Acknowledgements

I would like to take the chance of using this section to thanks my tutor Lluís Padró for his guidance, ideas and useful advice to carryout this project.
Professor also guided me very well how to implement the research based project.

Abstract

In the recent years, neural networks are very popular in the field of the artificial intelligence (AI). We decided to design the project on the basis of it. Neural network is the branch of the machine learning that uses the different layers to represent the data. Data are transformed to different or multi layers and generate the output.
Our project is User ticketing system with automatic resolution suggestions, in which data is text based and with the help of Natural language processing (NLP) we can solve the problem. NLP is a sub-field of artificial intelligence which creates interactions between computers and human language. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them.[1]
In this project the main goal is measuring the similarity between the texts. We design different models and use different layers and hyper-parameters later we will discuss in details. Our models are based on the Siamese neural network (SNN). SNN (sometimes called a twin neural network) is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors.[2]

In this project, three main approaches have been used. Firstly, on the basis of the pre-trained model we get the output. Secondly, we follow the transfer learning approach. The transfer learning consists of taking features learned on one problem, and leveraging them on a new, similar problem.[3] Thirdly, the model is fine tunned and designed on the basis of mutilingual (Spanish and English).

[1] https://en.wikipedia.org/wiki/Natural_language_processing
[2] https://en.wikipedia.org/wiki/Siamese_neural_network
[3] https://keras.io/guides/transfer_learning/

Contents

1 Introduction **11**
- 1.1 Objectives of the thesis . . . 12
- 1.2 Structure overview . . . 14

2 Backgrounds **15**
- 2.1 Neural networks . . . 15
  - 2.1.1 Activation Function . . . 19
  - 2.1.2 Training phase . . . 20
  - 2.1.3 Transfer learning & Fine-tuning: . . . 23
- 2.2 Siamese network . . . 24
- 2.3 Bag of words (BOW) . . . 24
- 2.4 Semantic similarity . . . 24
- 2.5 Encoders . . . 24
- 2.6 Similarity measurements . . . 25

3 State of art **27**
- 3.1 Word embedding . . . 28
  - 3.1.1 Word2vec . . . 28
  - 3.1.2 Glove . . . 29
  - 3.1.3 BERT . . . 29
- 3.2 Sentence embedding . . . 29
  - 3.2.1 DAN (Deep averaging network) . . . 29
  - 3.2.2 CNN . . . 30

4 Technology **33**

5 Functional Requirements **35**
- 5.1 Use case UC01: Login . . . 35
- 5.2 Use case UC02: Tickets already exist . . . 36
- 5.3 Use case UC03: Create ticket . . . 37
- 5.4 Use case UC04: View assigned tickets . . . 38
- 5.5 Use case UC05: View charts . . . 38
- 5.6 Use case UC06: View all tickets . . . 39

5.7 Use case UC07: Change status, ticket_type and priority of the ticket . . . 39
5.8 Use case UC08: Tickets assign to other team member . . . 40
5.9 Use case UC09: Create comment on the ticket . . . 40
5.10 Use case UC10: Receiving incoming emails . . . 41
5.11 Use case UC11: System create ticket or comments on the ticket . . . 42
5.12 Use case UC12: System assign the tickets to the agent . . . 42

6 Architecture and project files structure **45**
6.1 Files Structure . . . 46

7 Methodology **49**
7.1 Strategy . . . 49
7.1.1 Gathering Data . . . 50
7.1.2 Data Preparation . . . 50
7.1.3 Data pre-processing . . . 50
7.1.4 Training, validation and testing data set on the model . . . 50
7.1.5 Deploy Model . . . 51

8 Evaluation (experiment and results) **53**
8.1 Architecture and Design of baseline Model#1 . . . 55
8.1.1 Results of baseline Model#1 . . . 56
8.2 Architecture and Design of baseline Model#2 . . . 58
8.2.1 Results of baseline Model#2 . . . 59
8.3 Architecture and Design of baseline Model#3 . . . 60
8.3.1 Results of baseline Model#3 . . . 60
8.4 Architecture and Design of baseline Model#4 . . . 61
8.4.1 Results of baseline Model#4 . . . 63
8.5 Architecture and Design of baseline Model#5 . . . 65
8.5.1 Results of baseline Model#5 . . . 65
8.6 Architecture and Design of baseline Model#6 . . . 66
8.6.1 Results of baseline Model#6 . . . 67
8.7 Architecture and Design of baseline Model#7 . . . 67
8.7.1 Results of baseline Model#7 . . . 68
8.8 Architecture and Design of baseline Model#8 . . . 70
8.8.1 Results of baseline Model#8 . . . 71
8.9 Architecture and Design of baseline Model#9 . . . 73
8.9.1 Results of baseline Model#9 . . . 73
8.10 Baseline Model#10 . . . 73
8.10.1 Results of baseline Model#10 . . . 75
8.11 Development of the Web . . . 75
8.11.1 Login page: . . . 75
8.11.2 Similar tickets shown: . . . 75
8.11.3 Create ticket . . . 76
8.11.4 View assigned tickets . . . 77
8.11.5 Ticket details show . . . 78

8.11.6 Charts 78
8.11.7 View all tickets 79

9 Conclusion and Future work **81**

Chapter 1

Introduction

INLOC ROBOTICS is an innovative service SME offering solutions in the field of robotics and generally in the area of automation and control including industry 4.0 and IoT paradigms.
The company design the SEWDEF product for the detection and classification of sewer defects, which allows infrastructure management companies to obtain objective data for planning and prioritizing preventive, cleaning and maintenance works in sewer networks to avoid collapses, breaks or floods in time of rain.
The clients can access the control interface where they can upload the videos and add the inspection data. The system automatically finds the defects and makes the reports and charts about the state of the sewerage network.
If the clients face issues about the system related to upload the videos, adding the inspection data, reports are not working properly, charts are not displaying properly, video defects are not shown, if the order of defects is not according to the time-span and mages or videos are not working properly etc. These are queries the clients may have.
The current system of the company does not handle to response these types of queries.

The user ticketing system come into play to resolved this problem. With the help of this ticketing system the clients get the response very quickly, which improve the customer satisfaction. It handle high volume of the request. Preview the context of previous customer communication, no need to ask the same questions multiple times. It is easy and improve the team collaboration work, the agent can assigns the tickets to the team member.
It is automated process to manage the workload. When the ticket is created the ticket automatically assigns to team members (agents) based on workload. It helps to increase efficiency and productivity of agent. It also tracks the tickets volume on the basis of ticket type, status and priority.

The User ticketing system can be defined as if the users face an issue or have

any doubt about the system. They can create the tickets accordingly. When the user type questions, similar tickets will be shown accordingly. The system suggests the most similar tickets which already exists in the database.
The traditional approach to solve this problem, is based on matching the questions on the basis of common words without taking care of the context of the sentence or semantic similarity. Semantic similarity is a metric defined over a set of documents or terms, where the idea of distance between items is based on the likeness of their meaning or semantic content as opposed to lexicographical similarity.[1] Semantic similarity methods usually give a ranking or percentage of similarity between texts, rather than a binary decision as similar or not similar. Semantic similarity is often used synonymously with semantic relatedness[4].
This problem is solved with the help of NLP techniques (deep neural network-based methods). The Siamese neural network (SNN) model was designed with the help of keras library. To design the model, different layers, inputs layers, embedding layers, dense layer and the output layer have been used. For embedding, the tensor flow pre-trained model (universal sentence encoder) is used. The models take as input English strings and produce as output a fixed dimensional embedding representation of the string[3].
In this project, three approaches have used. Firstly, the per-trained model is used to calculate the MRR value (Mean reciprocal rank).[2]

Secondly, the transfer learning approach is used by taking the layers from the previous trained model which freeze them and add new trainable layers on the top of the frozen layers. For this, different hyper-parameters are used in different layers.
Thirdly, fine-tunning approach is used in which firstly unfrozen the layers and retrained the model by using the low learning rate. It can be seen in more details in the experimental section.
The models can be trained with the help of Kaggle data set [3] and when the models are over-fitting, the training is stopped and the MRR (Mean reciprocal rank) value can be calculated.
On the basis of the MRR value, the best per-trained model was chosen for this project. When user type the question, the chosen model fetch the top 5 most similar questions from the database which will be shown to the users.

1.1 Objectives of the thesis

The project is web based. It has the following functionality.

1. Detection of similar tickets work-flow works.

   Client Side:

[1] https://en.wikipedia.org/wiki/Semantic_similarity
[2] https://en.wikipedia.org/wiki/Mean_reciprocal_rank
[3] https://www.kaggle.com/code/trottefox/train-data-augmentation-quora/data

- In the client side when user creates the tickets and type the questions or doubts, similar tickets will be shown automatically which are created in the past. In this way duplicate tickets will not be created.
- During creation of ticket it has text fields (title, description) and drop downs menu (priority of ticket and ticket_type)
- When ticket is created, the system automatically assign it to the support team member (Agent) which has low tickets.
- Ticket will be in multiple languages (English and Spanish).

Agent Side:

- The agent can see only those tickets which which are assign to him.
- He can change the status of the tickets from Open to In_process or Resolved.
- When ticket is resolved, the system automatically send the email to the client (no_reply@inlocrobotics.com)
- Agent can create the tickets.

Admin Side:

- Admin can see all the tickets which are assign to different agents.
- He can assign the tickets (which status are Open) to other agents.
- He can also change the status of the tickets.
- Admin can also create the tickets.

2. Communication through Email chanel work-flow works.

- When an end-user sends an email to the support email address, the system creates a ticket. Which will assign to the agent which has low number of tickets.
- Auto response to the User through email that your ticket has been created successfully (support@inlocrobotics.com inlocrobotics.com).
- If the person replies to the notification email, the reply creates a comment on the ticket.
- The conversation between the user and the agent continues until the issue is resolved.
- When ticket is resolved, the system automatically send the email to the client (no_reply@inlocrobotics.com).

1.2 Structure overview

This document is divided into 9 chapters. The second one is related to backgrounds in which it is introduced the different concepts of artificial neural network. The third chapter introduced the state of art in which it is discussed what similar problems have been already addressed by other people, and which solutions they used. The fourth chapter is related to the different technologies which are used for this project.
The fifth chapter introduced the functional requirement, it specifies the detailed definition of the project in terms of functionality to develop. The sixth chapter explain the architecture of the system.The seventh chapter introduced the methodology, which is used during training the model.
The eighth chapter introduced the experiments and analysis of results, in which different architectures of models have been discussed. Also explained the development of the project, the user interface of the web. The ninth chapter included the conclusion and future work.

Chapter 2

Backgrounds

2.1 Neural networks

Neural networks is also called artificial neural network. It is the form of the machine learning that uses the layers to represent the data. Artificial neural network is composed by collection of artificial neurons or nodes which are fully connected with each other by different layers. First is the input layer of the data and then some hidden layers are connected together and at the end there are output layers. The data is transformed to these layers and generate the output. These layers are discussed in details as follows.

Layers:

As it is mentioned above neurons are organized by different layers which are fully connected together. In this section we will discuss different layers which are used in the models. The three common types of the layers which all neural networks have are the following ones.

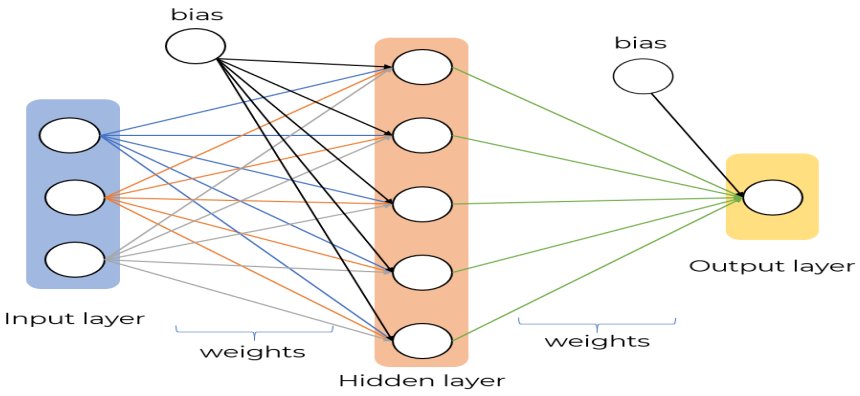
Input Layers: It is first layer of the neural network. It processes the raw or textual data. In this project input layer take the batch of sentences in a 1-D tensor of strings as input.

Hidden Layers: It is most important layer. These layers are situated between the input and output layers. In neural network there are many hidden layers. Why these layers are called hidden because they are not observed when using the neural networks. They take the set of weighted input and produce the output by using the activation function. Later, it will be discussed in detailed how they work.

Output layer: It is the last layer of the neural networks. It represents the outcome of the neural network. For example, the net that decide where the pairs

of the question is similar or not. It classify the value by using the calculation of the previous layers. It must be used one neuron or node for classification. It represent the two classes. In this project its value ranges between 0 and 1. If the value is above 0.5, it means the pairs of questions are similar otherwise pair of sentences are opposite.

Fully Connected:
These three layers are fully connected to each other. Every single layer connected to the other layer with weights. Each neuron in the layers densely connected with other neurons of other layers. Densely connected layer means that it is connected to every node from the previous layer. In this case every single node in the input layer is connected to every single node of the hidden layer as you can see in the figure below. And these connection we call weights.



These weights are actually what the neural network is going to change and optimize to determine the mapping from our input to output. There are some kinds of activation functions, when some input is given, in return it gives some output. How to get input and output from activation function ?? Later it will be discussed in details.

From the above figure you can see the different number of lines which connected the layers. Every single one of these lines are the number or some numeric value. These numeric values are between zero and one but they can be large or they can be negative. It really depends on what kind of network is using and how they are design.

The numbers for every single one of these lines are called trainable parameters, on the basis of this our neural network will actually tweak and change as we train the model to get the best possible result. From the above figure you can see our hidden layers densely connected to our output layer as well.

Apart from the weights we also have biases. The biases are little bit different than other nodes. There is only one bias and it exists in the previous layer that it affects. From the above figure it can be seen that the bias is densely connected to the next layer. The bias does not take any input information. This is another trainable parameter for the network. It has some constant numeric value that is going to connect the hidden layer. The weight of the connected lines is always 1. When bias is connected to the hidden layers or output layers. The bias are not connected to each other because they are just constant value.

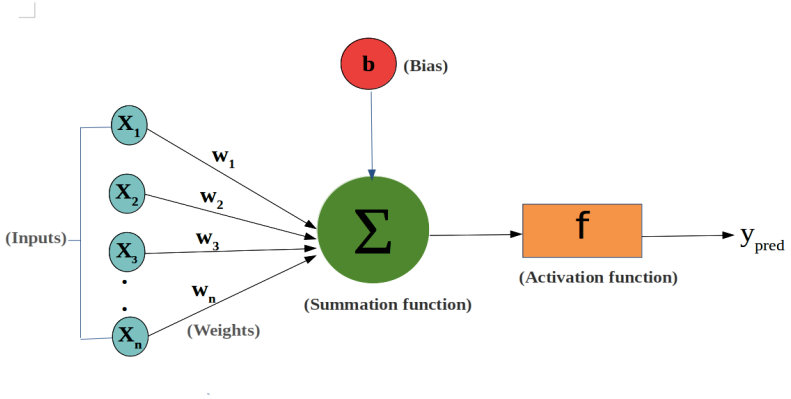
How to calculate the output value of the neuron in artificial neuron network.

The following equation shows how we calculate the output value of the neuron.

$$output = \sum_{i=0}^{n}(Weight_i.X_i) + bias \tag{2.1}$$

In the hidden layers or the output layers the input values come from the previous layer. e.g in first layer the value of the neuron or node are the following. Node 1 = 1, Node 2 = 0, Node 3 = 1, Node 4 = 0. The Node 5 which belong to the next layer takes the input from the previous layers. The first layer nodes which are fully connected to next layer nodes like node 5 their weights values are between -1 and 1. So, node 5 value calculated as follows

Node5 $= w_1(1) + w_2(0) + w_3(1) + w_4(0)$
It can also be see in the below diagram



Batch Normalization:
Normalization is the process which changes the value of numerical variable to a typical scale. While batch normalization is a technique which is used in artificial neural network and make the training process faster and stable through

normalization of layers input[7].
This effect stabilizes the learning process and reduces the number of training epochs which required to train the deep network.
Batch normalization is applied to the layers within the network. It, normalize the output from activation function. It keeps the weights of network balanced. It reduces the ability of outlying large weights that will influence the training process.

Dropout:
Deep learning neural network are quickly over fit a training data set after few epochs. When there are huge amounts of the weight and bias parameters, this leads to the network over fit. This layer takes the parameter ratio which rages between 0 and 1. With the help of rate, the node can be deactivated.
In this layer, some activations (nodes) are dropped out or emitting or deactivating randomly as it can be seen in the Figure 2.1. It significantly helps also in speeding up the training phase because it reduces the complexity of the model [21]. The over fitting problem can be solved with the help of this layer.

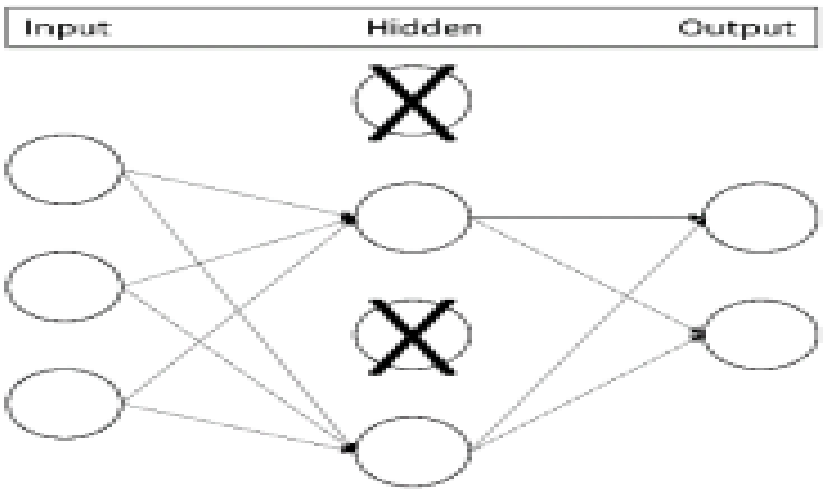


Figure 2.1: Dropout Layer

Concatenation:
This layer concatenate the list of inputs. It takes the input list of tensors, all of them have same shape and it returns the single tensor that is concatenation of all inputs.

Dot:
This layer computes the dot product between the sample of two tensors. This layer takes the different parameters; one of them is normalize and its value is

Boolean. If it is true it means the output of the dot product is the cosine similarity between two samples.

LSTM:
These networks are special kind of the Recurrent neural network. During training the text data, it is necessary for the networks to remember the previous word, to capture the context and LSTM have capacity to do so.
e.g consider the following piece of text "Alex is from Spain. He is fluent in Spanish. He loves to travel." While we reach the second sentence of text. It is essential to remember the words "Alex" and "Spain". Many researchers use LSTM architecture to measure the semantic similarity between the documents[4].

Hub keras layer (embedding layers):
This layer wraps the callable object for use as a keras layer. It is a repository of trained machine learning models which is ready for fine-tuning and deployable anywhere. It integrated with Tensorflow. Due to this, it loads all the tensorflow pre-trained model. With the help of it, the customize model can be trained. It improves the generalization and speeds up the training process.

2.1.1 Activation Function

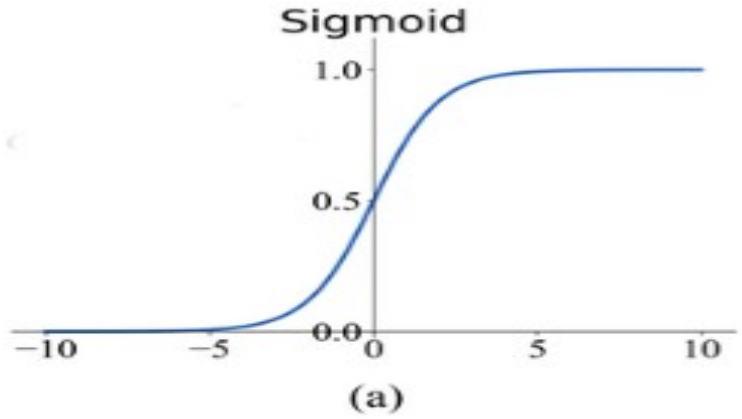
After obtaining the results from the previous formula as mentioned in the equation, then apply the activation function on it. The purpose of the activation function is to introduce the non-linearity into output of a neuron. The non-linearity solves the complex problems which build the complex relationship between inputs and outputs.
It also determines the output of the neural network between 0 and 1. It maps the value between 0 and 1. The new equation with activation function is as below.

$$output = F(\sum_{i=0}^{n}(Weight_i.X_i) + bias) \tag{2.2}$$

Following are the most common activation functions which are used in the neural network.

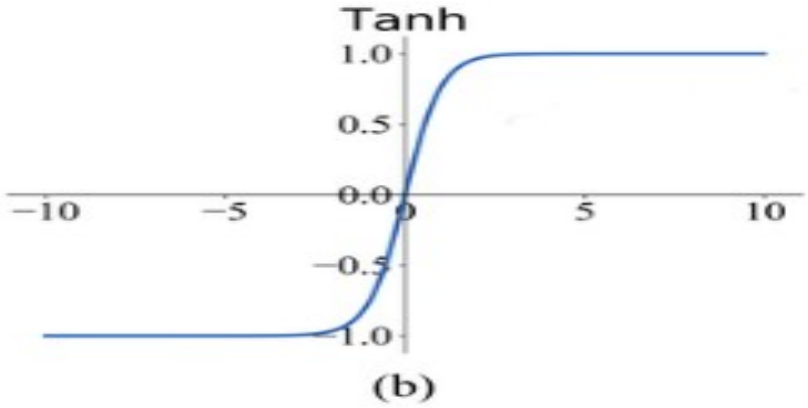
- **Sigmoid**.
  It takes the real value as input and output value ranges between 0 and 1. If the input value is larger (more positive) it will be closer to output value 1, whereas if the input value is smaller (more negative) it will be closer to the output value 0. As it can be seen in the image below

- **Tanh (Hyperbolic Tangent)**.
  It is very similar to the sigmoid activation function, this function takes the real value as input and output value ranges between -1 and 1. If the input value is larger (more positive) it will be closer to output value 1,



(a)

whereas if the input value is smaller (more negative) it will be closer to the output value -1. As it can be seen in the below Figure(b)



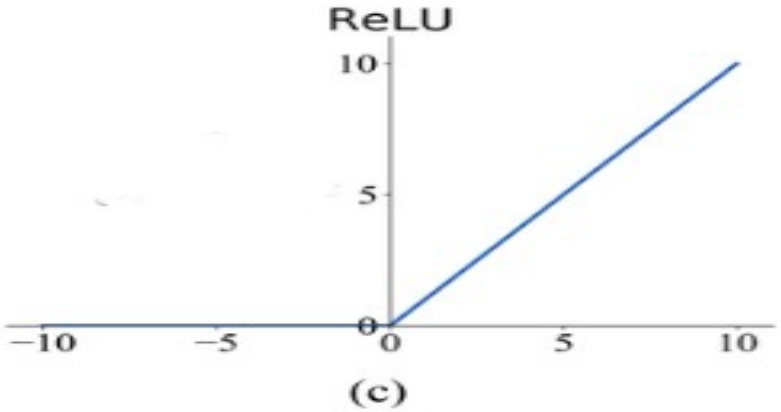
(b)

- **Relu (Rectified Linear Unit)**.
  It takes any value and makes the value zero which is less than zero. Any value which is positive, it remains the same as it is. It can be seen in the Figure(c)

2.1.2 Training phase

Before introducing the training phase of neural network, it is important to understand some terms which are used.

- **Sample:** It represent the row of data. It contains the inputs which feed to the neural network.



(c)

- **Batch size:** It defines the number of samples that will be propagated through the network before updating the internal weights. The training data set splits into one ore more batch
- **Epoch:** One epoch means that each sample in the training set has an opportunity to update the internal weights. An epoch consist of one or more batch. In other words number of epochs define the amount of times the whole training set is traversed through the neural network. It is used to test the values of the training data set (accuracy, loss and recall) and validate the data set (accuracy, loss and recall) during the training.
- **Example:** Assume you have a data set with 200 samples (rows of data) and you choose a batch size of 5 and 100 epochs.
  This means that the data set will be divided into 40 batches, each with five samples. The model weights will be updated after each batch of five samples.
  This also means that one epoch will involve 40 batches. With 100 epochs, the model will be exposed to or pass through the whole data set 100 times. That is a total of 4,000 batches during the entire training process.

The training process is divided into the two stages forward propagation and backward propagation. The Figure 2.2 shows how the training algorithm working.
In forward propagation the network receives the batch of samples from training data set and it moves the data from input layer (left) to the output layer (right). The output of the neuron is calculated as discussed in the above equation. The ANN generate an output for each batch which will be used in loss function to calculate the error between the expected and the obtained output.

It is known that the output layer stores the results which are predicted from the network. In the training phase the network will make many mistakes and poor predictions. In fact at the start of the training the network does not know

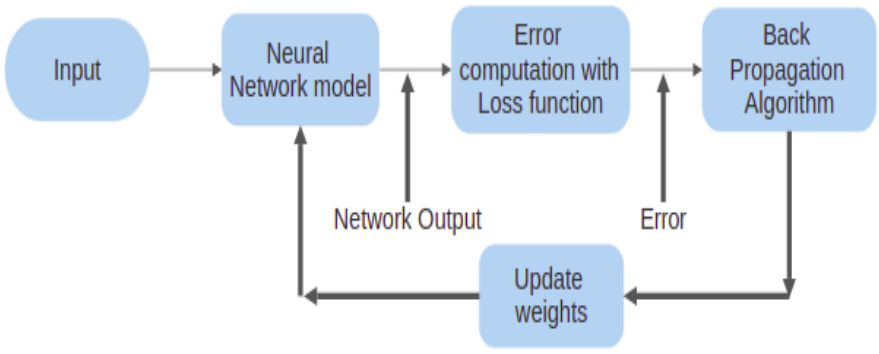


Figure 2.2: ANN training Diagram

anything as it has random weights and biases.
It is necessary to evaluate if the network is doing well and how well it is doing. From the training data there are inputs and the expected output, it is possible to compare the output from the network to the expected output. Based on the difference between these values, it can be determined if the network has a good job or poor job. This where the loss function comes in. The loss function calculates how far away output is from the expected output.
E.g if network output is 0 and the expected output is 0.7 it means this network is really bad because it gives really high loss value.

In backward propagation update the weights of all neuron in the network after calculating the error from loss function. This process is moving from right to left i.e backward from the output to input layer that's why this process is called backward propagation.

The main idea is to minimize the loss function. The lower the value of the loss function it means there is less difference between the obtained output and the real output. There are different techniques which are used to optimize the loss function e.g stochastic gradient descent [18] one of the most popular ones.

The gradient calculates the error gradient or the slop of the error whereas descent refer to the moving down along the slop towards the minimum value of the error. This process is iterative after each batch training which slightly improves its value.

The global minimum of the loss function are calculated by using the learning rate. The too higher learning rate and too smaller learning rate never reach to minimum of the cost function. That's why chose the hyper parameter very carefully.

It is generally advisable that the learning rate value at start of the training phase will be high and then it decreases over the time. Because at the start, the weights are too far from the global minimum of the loss function therefore, bigger steps can obtain faster and better solution. In contrary, sometimes high learning rate leads the ANN to a worst performance.

The parameter for this network are weight and bias. By changing these weights and bias after each batch makes either the network better or worse. The loss function will be determined if the network is getting better or worse to know how the network is working.

In the Figure 2.3 it can be seen how the gradient descent works. The gradient represents the slop of the error. The graph shows the initial weight is too far from the global cost minimum(J_min(w)). The arrow from initial weight to global minimum represents different steps taken by the algorithm. After each step the network is making better predictions and having lower loss.

Once the backward propagation algorithm is complete, the weights are updated accordingly. Then for the next batch again it starts with forward propagation stage and repeats the process again and again as shown in the Figure2.2

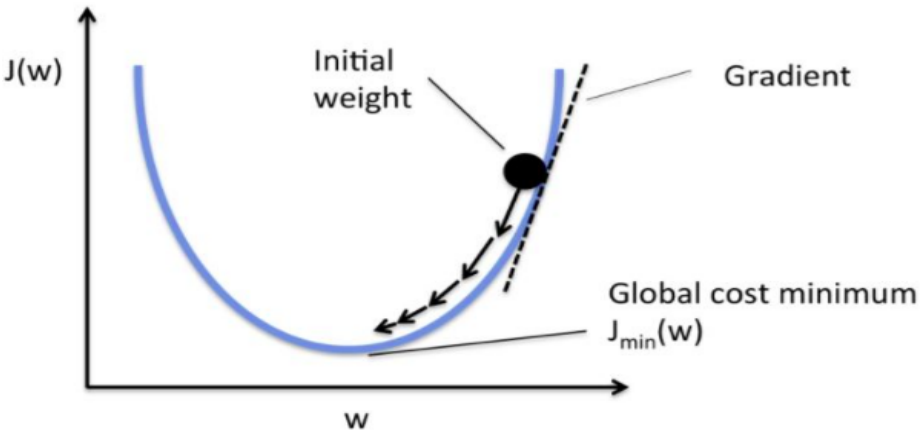


Figure 2.3:

2.1.3 Transfer learning & Fine-tuning:

It is a machine learning technique where model is trained for one problem and re-use on second similar problem. It is an optimization that improves the performance when model is trained on the new similar problem.
In transfer learning first train a base model, then reuses the learned features

or weights and transfer them on the another target model to be trained on the target data set [25]. The most common type of the transfer learning in the context of deep learning is the following.

- Reuse the layer of the pretrained model.
- Freeze the layers. It means these layers weights will not be updated.
- Add few new trainable layers on the top of the frozen layers.
- Trained the new layers on new data set.

The last step is the fin-tuning which unfreezes the whole model or unfreezes the pre-trained model layers and re-trains again the model on the new data set with low learning rate. This will also increase the performance of the model.

2.2 Siamese network

It is the class of neural network architecture that has two or more identical sub-networks. Identical means they have same configuration with same parameters and weights. Parameter updating is mirror across both sub-networks.
If the weight in one sub-network is updated then the weight in another network is updated as well. It is used to find similarity between inputs by comparing the feature vectors.

2.3 Bag of words (BOW)

It is the representation of the text which describes the occurrence of the words within the document. The information about order, grammar and structure of words in the documents is discarded.

2.4 Semantic similarity

Semantic similarity methods usually give a ranking or percentage of similarity between texts, rather than a binary decision as similar or not similar. Semantic similarity is often used synonymously with semantic relatedness[4]. It measures the likeness that two words or documents have same meaning.
e.g the words ‘coffee’ and ‘tea’ are semantically similar and the word ‘car’ and ‘bus’ also semantically similar. The sentences “What is your age” and “How old are you” are also semantically similar. It is opposite to the Lexical similarity.

2.5 Encoders

The data set is text based to convert the data into digits for that it requires to embedding the text data. In the present project, the sentence embedding is

used which is the techniques of the natural language processing NLP where the sentence maps to the vectors of the real number.
In the project, the sentence used for embedding is the universal sentence encoding which takes as input English strings and produces as output a fixed dimensional embedding representation of the string [3]. The output is a 512 dimensional vector.

The universal sentence encoders are design with two encoding model. The two encoders have different design and architecture. One based on the transformer architecture targets high accuracy at the cost of greater model complexity and resource consumption. The other based on Deep Averaging Network (DAN) architecture targets efficient inference with slightly reduced accuracy [3].

The computation and memory usage of the transformer based architecture is higher than the DAN based architecture. The transformer model time complexity is $O(n^2)$ in the length of sentence, while the DAN model is O(n). The transformer model space complexity also scales quadratically $O(n^2)$ in sentence length, while the DAN model space complexity is constant in the length of the sentence [3].

Our web based project is multilingual, so for English language we choose the DAN based architecture. Because on CPU it works more efficiently.

There is an another model called universal sentence multilingual, it takes the input sentences in 16 languages and produces an output a fixed dimensional embedding representation of the string[24].The output is a 512 dimensional vector.
This model is also designed with two encoding models. One is based on transformer architecture and another is based on CNN(Convolutional Neural Network).

As we mentioned above that our web based project supports the multilingual, so for Spanish language we choose the CNN based architecture.
Because on CPU it also works more efficiently as compare to the transformer based architecture. With increase the length of the sentence, the memory of the Transformer based architecture increases more than twice as fast as the CNN model[24].

2.6 Similarity measurements

Semantic textual similarity (STS) between sentence pair can be measured by the cosine similarity. The cosine similarity is calculated by the cosine of angle between two vectors. As can be seen in the equations below.

The cosine of two non-zero vectors can be derived by using the Euclidean

dot product formula:

$$\mathbf{A}.\mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

Given two vectors of attributes, A and B, the cosine similarity, $\cos(\theta)$, is represented using a dot product and magnitude as

$$cosin\_similarity = \cos(\theta) = \frac{A.B}{||A||||B||}$$

Basically it is cosine of angel between two n-dimensional vectors in a n- dimensional space. As it can be seen in the above equation in the numerator it is the dot product of the two vectors divided by the product of the two vectors length or magnitudes.

The cosine similarity always belongs to the interval [-1,1]. If the cosine similarity scores closer to 1, it means it is similar sentence. If the score is more closer to the 0 it means it is opposite sentence. For this project we set the threshold value 0.5. If the score is above the 0.5 it means it has same context of the sentence and vice versa.

Chapter 3

State of art

Similar questions or tickets can be seen as natural extension of information retrieval (IR) system. In the formal IR system the user information need is expressed through a query, usually it consists on set of keywords.
The query is used to retrieve the information from the data set. The output of the IR consists of set of documents or higher ranked texts which are extracted from the database. If the query is well formatted, the good system retrieves the higher ranked documents that satisfies the user information need.
In similar ticket system, the query consists of NL (natural language) question and the output of system is the set of likely relevant documents.
The similar problems are already solved by using different techniques, the first approach is **filtering** it matches the exact sub-string from the database. A better approach could be case-insensitive match but this is only marginally better[1].

The second approach is the **search query** where the order of the search does not matter. e.g query 'login issue' and 'issue login' the results of both queries are same[2]. If the query has 'facing login issue' it will not match with string 'login issue' because 'facing' word is missing.
The third approach is **SearchRank**, it takes care how often the query terms appear in the document. It is based on occurrence of the words. The better the match is the higher the value of the rank[3]. e.g if we search 'Stevens' with the help of the search rank the database returns the following results 'Fisher Stevens', 'W. Richard Stevens', 'Robert Louis Stevenson'
From the results, it can be seen that it fetches the result from DB in an ascending order.
The fourth approach is **Trigram similarity**, it compares the three consecutive characters shared between the search query and database queries, on the basis of which it returns the results[4].

[1] https://docs.djangoproject.com/en/4.0/topics/db/search/
[2] https://docs.djangoproject.com/en/4.0/ref/contrib/postgres/search/#searchquery
[3] https://docs.djangoproject.com/en/4.0/ref/contrib/postgres/search/#searchrank
[4] https://docs.djangoproject.com/en/4.0/ref/contrib/postgres/search/#trigram-similarity

The Django web framework used the above techniques. The drawback of the above techniques they could not measure the semantic similarity between the texts.

In this project, if users have any doubts about the system, they can create the ticket. So the data is increase over time in the database. The exponentially increase in text data generated over the time. The NLP (Natural language processing) has gained significant attention from AI[4].

Measuring the semantic similarity between text components like sentences or document play vital role in wide range of NLP tasks like information retrieval[9].

Now, we will discuss the various semantic similarity techniques using the deep neural network-based method which are using now a days.

3.1 Word embedding

It is corpus-based semantic similarity method which measures the semantic similarity between the documents using the information retrieval from large corpora. It follows the principle of 'distributional hypothesis'.

The distributional hypothesis is that in which the word occurs in the same contexts tend to have similar meaning[6]. Text is represented as a vector, several semantic distance measures based on the distributional hypothesis were proposed to estimate the similarity between the vector[4].
Among the different measurement units, the cosine similarity gained significant role and it is widely used among the NLP researchers[14].

Word embedding provide the vector representation of the words, where these vector have understanding the relationship between the words[19]. These vectors are computed using different approaches like neural networks[13], word co-occurrence matrix [16], or representations in terms of the context in which the word appears [11] Most widely used pre-trained word embedding are the following.

3.1.1 Word2vec

It is developed from the Google news data set which containing approximately 3 million vector representation of words and phrases. It is neural network model which is used to produce distributed vector representation of the words. There are two different models of word2vec proposed one is the Continuous Bag of Words (CBOW) and another one is the Skip-gram model[13].
The architecture of the network is simple it contains an input layer, one hidden layer and an output layer. The input layer is fed with large text corpus, and

output of the model is the vector representation of the word.
The CBOW model predicts the current word using the context of the neighbor word. While the Skip-gram model predicts the neighboring words given the target words. It also retains the contextual similarity between the words.
Many researchers extended the word2vec model context vectors [12], dictionary vectors [22], sentence vectors [15] and paragraph vectors [10].

3.1.2 Glove

It is developed by the Stanford university, it estimates the similarity which is based on the word similar to each other occur together. The co-occurrence matrix tells how often the particular word pair occur together. This model is trained with five different corpora mostly Wikipedia dumps[16].

3.1.3 BERT

It uses the transformer architecture proposed by the Vaswani et al.[23] Which produces attention based word vector using the bi-directional encoder. It involves two main process pre-training and fine-tuning.
The model is pretraining using the corpus of 3300M words from both Book corpus and English Wikipedia[4].
It reads the input text in both direction (left to right or right to left) and trains the model. Also fine-tunes the model. In fine-tuning model is trained for specific NLP task.

3.2 Sentence embedding

It is a technique in natural language processing (NLP) where the sentences are mapped to vector of the real number[5].
Sentence embedding is the extension of the word embedding. One of the most popular sentence embedding techniques is the universal sentence encoder. It is neural network pre-trained model. It is developed by Google. It can be used for multi task learning like sentiment analysis, text classification, sentence similarity etc.
The most interesting part is that this encoder is based on two encoder models, one based on transformer architecture and another based on DAN (Deep averaging network). The models take an input of the English string and produce output of fixed dimensional embedding representation of the string[3].
Here we will only discuss the deep averaging network.

3.2.1 DAN (Deep averaging network)

The input embedding for words and bi-grams are first average together then pass to the feed forward deep neural network to produce the sentence embedding.

[5]https://en.wikipedia.org/wiki/Sentence_embedding

It takes lowercase string as input and generates the output of 512 dimensional sentence embedding. The time complexity of this model is O(n).

The another most popular multilingual sentence embedding technique is universal sentence encoder multilingual. It is also neural network pre-trained model.

The architecture of USE multilingual uses both features of word and feature of the n-gram to encode the sentence into real valued vectors, with the property that sentences with similar meanings have high cosine similarity in embedding space[8].

The encoder of this model is based on two encoder models(CNN and transformer). This model takes the input in 16 languages as a string and produces the output of fixed dimensional embedding representation of the string[24].
Here we will only discuss the deep averaging network convolutional neural network (CNN)

3.2.2 CNN

In this project for Spanish language, the universal sentence encoding multilingual is used and it is based on CNN architecture. So the CNN model is used for creating the general purpose sentence embedding.
Each convolution layer has number of filters which are used to extract the morphological(form of words) information from the word vector
The CNN model is trained with one layer of convolution on the top of the word vectors obtained form unsupervised neural language model. They are trained one billion words of Google news.
The convolution neural network takes the input as token sequence embedding. The pooling is used to turn the token level embedding into the fixed length. presentation[24]

As we discussed above different solutions are already exist related to sentence similarity, our project is web based and we are using Django web framework. In the Dajango we see that they are using these techniques filtering data, search query, SearchRank and Trigram similarity.
As we already mentioned above these techniques are not measuring the semantic similarity between the text. If follow the Lexical similarity approach.

Later we discussed the word and sentence embedding, both measuring the semantic similarity between the sentences. But we see that word embedding is based on the word and sentence embedding is based on combination of the words(sentences.). According to this paper[1]
USE achieve the better accuracy values as compared to the word2vec models. Because USE is specially designed to better understanding the context of the sentence or paragraph.

So, for the present project it is decided to follow the semantic similarity approach using the pre-trained neural network. For that we chose the USE (universal sentence encoder) model for showing the similar tickets to the user.

Chapter 4

Technology

The following technology is used in the present project.

- **Tensorflow**
  It is open source deep learning frame work which is developed and maintained by the Google. It is used to build the machine learning model. It supports the different programming languages such as Python, JavaScript, c++ and Java.[1]
  Directly using the TensorFlow can be challenging, the modern tf.keras API is used the keras API to the tensorflow project. tf.keras is the tensorflow API which is used for building and training the deep learning models.[2]

  Tensorflow can be run on multiple CPUs and GPUs. With the help of tensorflow hub, different ML pre-trained model can be loaded. TensorFlow Hub is an open source repository of trained machine learning models ready for fine-tuning and deployable anywhere. Reuse trained models with just a few lines of code.[3] In the present project we are using the pre-trained model universal sentence encoder

- **Keras**
  It is an open source python library which is used for developing and evaluating the deep learning model. It acts as an interface for TensorFlow library. It wraps the different numerical computation libraries such as TensorFlow which allows to define and train neural network model in just few lines of code.[4]
  The model can be designed by using the different layers (input, hidden, output , batch normalization etc) of the keras. Also compile the model by using the different loss function and optimizers. In the next sections, it can be seen that the different models are built by using this library.

[1] https://www.tensorflow.org/
[2] https://www.tensorflow.org/api_docs/python/tf/keras/
[3] https://www.tensorflow.org/hub/overview
[4] https://keras.io/api/

- **Matplotlib.pyplot** This is the python package which is used to plot the graph. Different graphs related to model accuracy and performance can be seen in the experimental section.

- **Django**
  It is the python web framework which is used to develop web applications. It follows the MVT (Model View Template) design pattern. It is very demanding because with the help of it development is very rapid. With the help of this framework, the different functionality is implemented.

- **Celery**
  Celery in the open source library which is written in Python, it is used to execute the tasks asynchronously. Basically it is the task queue which handle the tasks and distribute them to the worker. It supports the scheduling, which executes the tasks on the basis of the scheduled time.

- **Sqlite3**
  It is embedded relational database management system, it is self contained, server-less and file-based SQL database. The whole SQL database with multiple tables, triggers, indices and views is contained in a single file.
  In this database we create different tables and also create the relationship with them. All the information related to the tickets are stored in it. Later,we will discuss these things in detailed.

- **Smtplib**
  This is the python package, basically it is SMTP client session object which is used to send email to the any client Internet machine with SMTP. It is used for email transfer.

- **Imaplib**
  It is also python package, it is python client side library that is used for accessing emails over imap protocol. With the help of imap we can retrieve the data from the email.
  Basically our system create the tickets when client sent email to support team. So with the of this module we read the incoming email which will be discussed in detail.

- **Git**
  It is the platform for code hosting. It provides the version controlling system and collaboration. It also allows the developer to work as a team.[5]
  It hosts the source code of the present project.

[5]https://github.com/

Chapter 5

Functional Requirements

5.1 Use case UC01: Login

Main Actor	Client, Agent and Admin
Precondition	The users already registered in the system.
Input	The user want to access the system.
Output	1. If the user not login the system shows the sign-in form. 2. The user insert the user-name and password. 3. The user click on sign-in button. 4. The system checks if the user exists and validate the tow fields. If the user name or password is wrong then it show the alerts. 5. The system redirects the user to the main page if it login successfully.

Table 8.1: This table describe the login process. First define the main actors and precondition. Then it explains what the actions needs to be taken for login.

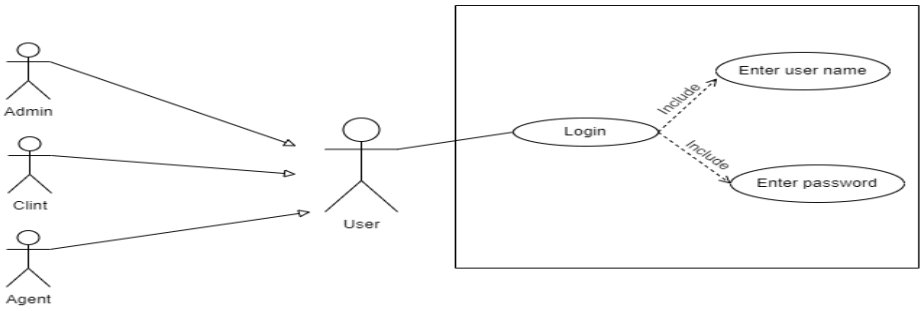


Figure 5.1: The use case describe the access process to system. The detailed description is reported in the table 8.1

5.2 Use case UC02: Tickets already exist

Main Actor	Client, Agent and Admin
Precondition	The users must login.
Input	The user want to preview the ticket.
Output	1. The system shows the create ticket form. 2. In the title field users type the question or doubts they have. 3. If the similar questions is already exist in the database, similar tickets will be shown to the user. 4. The user can click on of tickets and it shows the ticket details.

Table 8.2: This table describe the process of already exist tickets. First define the main actors and precondition. Then it explains what the actions needs to be taken for preview the already exist tickets.

5.3 Use case UC03: Create ticket

Main Actor	Client, Agent and Admin
Precondition	The users must login.
Input	The user want to create the ticket.
Output	1. The system shows the create ticket form. 2. In the title field users type the question or doubts they have. 3. If the similar questions is already exist in the database, similar tickets will be shown to the user. 4. The user can click one of tickets and it shows the ticket details. 5. In the description user write the detailed concerns about the system. 6. From the drop down choose the priority and ticket type. 7. Click on the create ticket, the system will creates the ticket.

Table 8.3: This table describe the process of ticket create. First define the main actors and precondition. Then it explains what the actions needs to be taken for create the ticket.

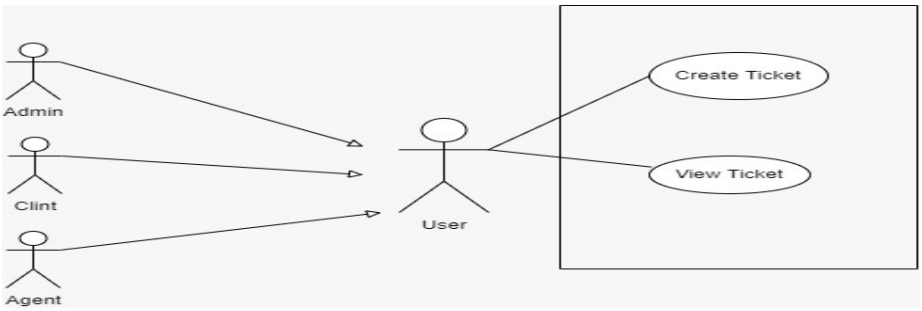


Figure 5.2: The use case describe the create tickets and preview the already exist tickets in the DB. The detailed description is reported in the table 8.2 and table 8.3

5.4 Use case UC04: View assigned tickets

Main Actor	Agent and Admin
Precondition	The users must login.
Input	The user want to access the assigned tickets.
Output	1. Go to the assigned tickets page, the user(agents or admin) can see all the tickets which assign to them. 2. In the search field they can filter the assigned tickets. 3. Click on the show drop down they can select the number of tickets will shown in the page. 4. From the pagination they can click on next and previous button to see the next and previous page tickets respectively.

Table 8.4: This table describe the process to view assigned tickets. First define the main actors and precondition. Then it explains what the actions needs to be taken for viewing the assigned tickets.

5.5 Use case UC05: View charts

Main Actor	Agent and Admin
Precondition	The users must login.
Input	The user want to access the charts about the tickets.
Output	1. After login system show the charts page. 2. The user (agent or admin) can see different charts, which shows the number of assigned tickets on the basis of priority, status and tickets type. 3. Click on the charts, it redirect to the tickets page.

Table 8.5: This table describe the process to view charts. First define the main actors and precondition. Then it explains what the actions needs to be taken for viewing the charts.

5.6 Use case UC06: View all tickets

Main Actor	Admin
Precondition	The users must login.
Input	The admin want to access the all tickets exits in database.
Output	1. Go to the All tickets tab, click on the tickets. The admin can see list of all the tickets which assign to all agents.

Table 8.6: This table describe the process to view all tickets exists in the system. First define the main actors and precondition. Then it explains what the actions needs to be taken for viewing the all tickets.

5.7 Use case UC07: Change status, ticket_type and priority of the ticket

Main Actor	Agent and Admin
Precondition	The users must login.
Input	The user(agent or admin) want to change status, ticket_type and priority of the ticket.
Output	1. From list of assigned tickets, click on one of the them, the user(agents or admin) can see details of the ticket. 2. From the 'Priority' drop down, change the priority of the ticket. 3. From the drop down of the status, change the status of the ticket. 4. From the drop down of the ticket_type change the type of the ticket.

Table 8.7: This table describe the process to change the status, ticket_type and priority of the ticket. First define the main actors and precondition. Then it explains what the actions needs to be taken to change the status, ticket_type and priority of the ticket.

5.8 Use case UC08: Tickets assign to other team member

Main Actor	Agent and Admin
Precondition	The users must login.
Input	The user(agent or admin) want to assign ticket to other team member.
Output	1. From list of assigned tickets, click on one of the them, the user(agents or admin) can see details of the ticket. 2. From the 'Assigned To' drop down choose the team member, the system assign the ticket to him.

Table 8.8: This table describe the process to view assigned tickets to other team member. First define the main actors and precondition. Then it explains what the actions needs to be taken for assigns ticket to other team member.

5.9 Use case UC09: Create comment on the ticket

Main Actor	Agent and Admin
Precondition	The users must login.
Input	The user wants to create the comment on the ticket.
Output	1. On the ticket page, create comment form is shown. 2. If the client reply the email, the system creates the comment. 3. If comment is not already exist then system disable the comments. User (admin or agent) is not able to create comment. 4. If comments is already exist then user can create the comments on the ticket. 5. When the comment is created system automatically send email to the client.

Table 8.9: This table describe the process of create comment on the ticket. First define the main actors and precondition. Then it

explains what the actions needs to be taken for comment creation.

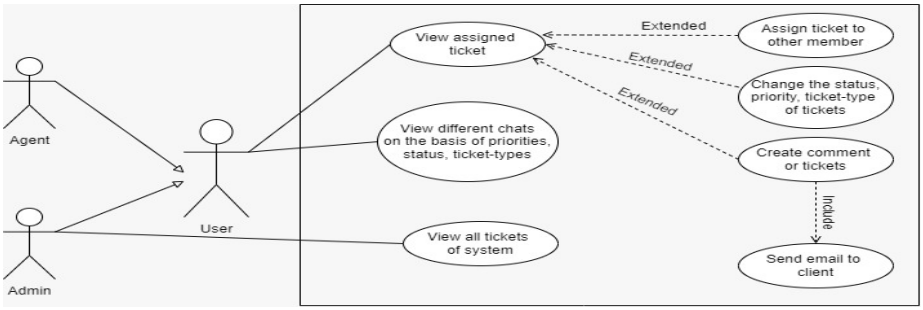


Figure 5.3: The use case describe the view assigned tickets, view different charts of the ticket and view all ticket of the system. It also explain the assign tickets to other member, change the status, priority, ticket_types of the ticket, create comment on tickets. The detailed description is reported in the table 8.4,8.5,8.6,8.7,8.8 and table 8.9

5.10 Use case UC10: Receiving incoming emails

Main Actor	System
Precondition	Celery server must be running.
Input	The system wants to receive the incoming emails.
Output	1. When client send an email to the support team email address. 2. After every 30 seconds, system read the incoming emails.

Table 8.10: This table describe the process of receiving email. First define the main actors and precondition. Then it explains what the actions needs to be taken by the system to receiving the emails.

5.11 Use case UC11: System create ticket or comments on the ticket

Main Actor	System
Precondition	Celery server must be running.
Input	The system wants to create tickets or comments on the ticket.
Output	1. After every 30 seconds, system read the incoming emails. 2. First time,if client send an email system create the ticket. 3. The support team try to resolve the issue. 4. If the issue is resolve by the user (admin or agent). 5. System send an email to the client your issue is resolved. 6. If client reply the email, then system automatically create the comment on the ticket.

Table 8.11: This table describe the process of create ticket or comment on the ticket by the system. First define the main actors and precondition. Then it explains what the actions needs to be taken by the system to create the tickets or comments on the ticket.

5.12 Use case UC12: System assign the tickets to the agent

Main Actor	System
Precondition	Celery server must be running.
Input	The system wants to assign the tickets to the agent.
Output	1. After every 30 seconds, system read the incoming emails. 2. First time,if client send an email system create the ticket. 3. This ticket is automatically assigns to the agent which has low number of tickets.

Table 8.12: This table describe the process of assign the tickets to the agent by the system. First define the main actors and precondition. Then it explains what the actions needs to be taken by the system to assign the tickets to the agent.

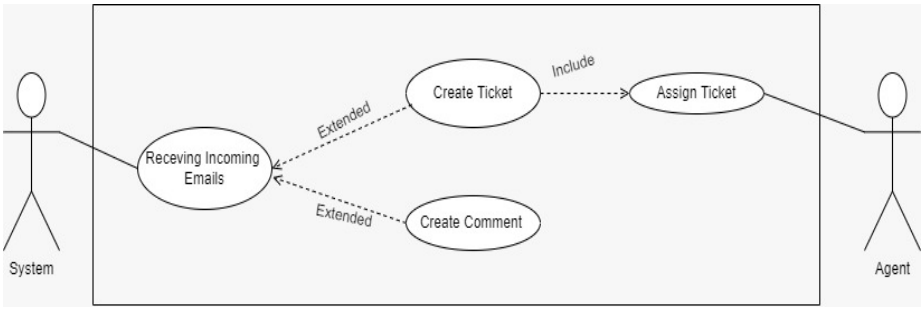


Figure 5.4: The use case describe the system receive the incoming emails, create tickets or comment on the ticket. It also explain the system assign tickets to the agent. The detailed description is reported in the table 8.10,8.11 and table 8.12

Chapter 6

Architecture and project files structure

This project is web based by using Django as already mentioned above. Basically Dajango is web application framework written in Python. It is based on Model View Template (MVT).
The MVT is the software design pattern, Model helps to handle the database in which database schema (Table structure) is defined. It is data access layer which handles the data of the database.

While View is used to handle the business logic, interact with the model which can perform the CRUD (Create, Read, Update and Delete) operation in the database. It also renders a template.

The Template is the presentation layer which handles the User interface. It contains the following files (HTML, CSS, JavaScript)
The Figure 6.1 shows the work flow of MVT. User requests the resources to

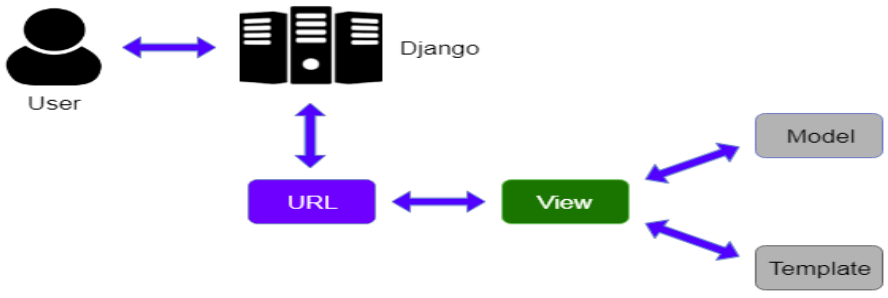


Figure 6.1: Work flow of the system[1]

[1]https://www.javatpoint.com/django-mvt

the Django, the Django works as a controller. It handled all the requests and checked if the available resources in the url.py file. If the URL match, then the view method is called which interact with model and template and return the template accordingly. This template will be shown to the user as a response.

6.1 Files Structure

- **manage.py:**
  This file is used to interact with the project, while using this file we can debugging, deployment, migrations and running the web server locally.
- **requirements.txt:**
  It contain the packages which are using in the project.
- **db.sqlit3:**
  It is sqlit3 database. It has different table which store the data. Also handle the relationships between the table. As you can see in the Figure6.2.

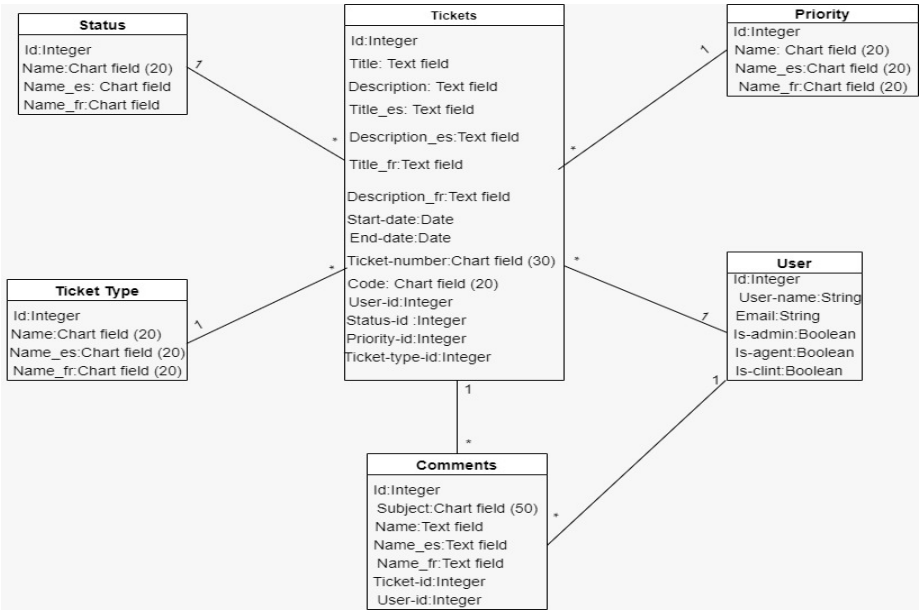


Figure 6.2: shows the database model created using the migration.The Ticket table has five foreign keys referring to the status,priority,ticket type, user and comments table. The comments table has one foreign keys referring to the user table.

- **ticketing_system:**
  Djngo working in file system to manage the web application. It creates

the project and application folder which handles different files.
The ticketing_system is the name of present project which contains the following files. The application name is Ticketing_auto_suggestion which will be discussed later.

- **_init_.py:**
  It is the python packages file, it invokes when the package or the module is imported. In the project we imported the celery app.
- **middleware.py:**
  It is lightweight plugin that processes during the request and response the execution. It is used to perform the following function security, session, authentication and crf protection etc.
- **settings.py:**
  It contains all the settings related to the project. In this file we register the app (Ticketing_auto_suggestion), location of the static file, database configuration etc.
- **urls.py:**
  It is universal resource locator, it handles all the requests from the users.
- **celery.py:**
  It is an open source library which is used to run the tasks asynchronously. It runs the job or tasks in the backgrounds. It is used for parallel executions of the task.
  In the project, it is used to read the incoming emails. The task is scheduled to read the incoming emails after 30 seconds.
- **Ticketing_auto_suggestion:**
  It is name of app which contains the following files. Migrations: The migration directory contains different migration files. It reflects any changes in the model of the database schema.
  Django creates the migrate file inside the migration folder for the each model. When the migration runs, it creates the tables into the database accordingly.
- **statics:**
  This directory handles the static resources like CSS, JavaScript and images.
- **templates:**
  This directory contains the Html pages of the projects. Django template engine also handles the python code and allows to build the dynamic web pages.
- **admin.py:**
  It is used to register the Djnago model or DB tables into the Django administration. Due to which Django model is displayed at the admin panel.

- **apps.py:**
  It is used for the configuration of app. models.py: It define the structure of the database. Database table schema are define here. Also it handles the relationship between the tables by using the foreign keys.

- **views.py:**
  This file stores the function based views. It handles the business logic and rendesr the template.

- **tasks.py:**
  It is celery based task which reads the incoming emails and creates the tickets in the system if the ticket does not exist.
  If the ticket already exists then it creates the comments against the tickets.

- **Universal-sentence-encoder_4:**
  It is universal sentence encoder pre-trained model which is used for embedding the English sentences.

- **Universal-sentence-encoder-multilingual_3:**
  It is universal sentence encoder multilingual pre-trained model which is used for embedding the Spanish sentences.

Chapter 7

Methodology

In this section we will discuss about the methodology that is going to be applied in the similar tickets shown to the user. Below we will discuss all the steps included in our workflow.

7.1 Strategy

Using the correct strategy makes the machine learning system (model) more quickly and efficiently. Also model gives the results more accurate. The general workflow of machine learning algorithms is shown in Figure 7.1

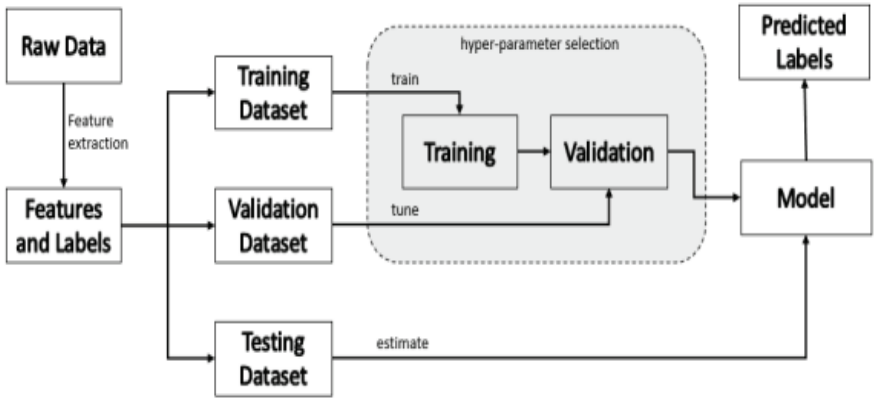


Figure 7.1: General workflow diagram of machine learning algorithms

7.1.1 Gathering Data

Gathering data is the most important task in a machine learning. We need to collect the real time data related to our problem. Which is User ticketing system with automatic resolution suggestions.
Our problem most resembles to the duplicate of the questions. So from the Kaggle[1] we get data set which contains the quora question pairs which are marked as the pair of questions either they are duplicate or not.
The collecting data cannot be used directly for performing the analysis process because it contains the missing values, extremely large values, unrecognized text or noisy data. To solve this problem data preparation is necessary.

7.1.2 Data Preparation

It is process to transform the collecting data into form that is more appropriate for the modeling. While developing the ML model, only a few few variables or column in the data set are useful for building the model while rest of the columns are redundant and irrelevant.
In our data set there are three main fields question1, question2 and is_duplicated. Question 1 and Question 2 are used as features column it means these are inputs of our model and is_duplicate is used as label column on the basis of it predicts the output values.

7.1.3 Data pre-processing

It helps to build the machine learning model more accurate. Data pre-processing is a process of the cleaning the raw data or selected features columns. So that will be used to train the model.
This includes dealing with artifacts such as missing values, normalization, augmentation, and quality control[20]. The following steps will be taken to clean the data.

- Firstly, we convert to lowercase and remove trailing and leading spaces.
- Replace certain special characters with their string equivalents.
- Replacing some extremely large numbers with string equivalents (not perfect, can be done better to account for more cases)
- Decontracting words(to expand from contracted)
- Remove the missing data

7.1.4 Training, validation and testing data set on the model

To train the model we split the data set into three chunks which are the training data, validation data and testing data, the percentage of the data are

[1] https://www.kaggle.com/code/trottefox/train-data-augmentation-quora/data

80%,10%,10% respectively.

In training data set the samples of the data used to fit the model.The training set is used to train the algorithm and develop the classifier model [2]. Weights and bias values are updated during the training. The model is learnt from this data.

The validation data set is used to evaluate the model frequently. The validation set used to estimate the value of accuracy, precision and recall. On the basis of these values we can fine-tune the model hypermeters. Model is never learnt from this data. From the validation set results update the hyperparameters accordingly. It is also called development set.

The test data set will be used when the model is completely trained using the train and validation set.To test the model on new data, the testing data set is used as input to the final model to predict output data labels. [2] The test set is used to assess the accuracy, precision, recall and performance of the model. This set helps to find out any issue or mistraining in the model.

7.1.5 Deploy Model

Once the model is trained and tested. If the model gives good results as per requirements then the model will be deployed in the present project.

Chapter 8

Evaluation (experiment and results)

In this section, different architectures and layers of the models have been discussed. Also, the results from training phase of each model will be discussed. Before explaining the experiments and result, it is important to understand some terms.

1. **Recall:**
   It measures the model ability to identifying the True positives. The higher the value of recall means more positive samples detected.

   $$R = TP/(TP + FN)$$

   $$TP = TruePositive$$

   $$FN = Falsenegative$$

   If the model classifies all positive samples as positive then the recall value will be 1.

2. **Overfitting:**
   We face the issue of the over fitting when the model fits perfectly to the training data set, as it can be seen in Figure 8.1. When testing on the validation data set the results are incorrect. This is because the error in the model has low bias and high variance.
   When the model is over trained it can start to learn the noise or irrelevant information within the data set. When the model memorizes the noise and fit closely to the training set, it falls into over fitting.

3. **Sentence Embedding:**
   It is the technique of natural language process (NLP), which represents the entire sentences and their semantic information as vector of real numbers. It helps the machine to understanding the context and intention of the entire sentence.

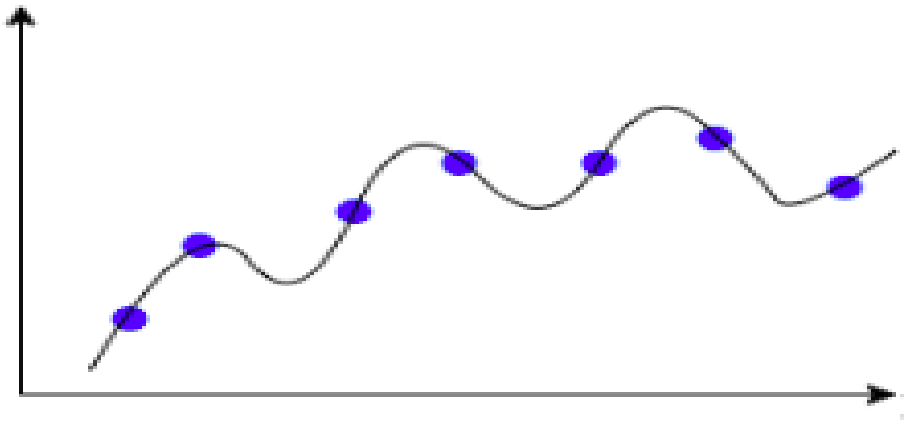


Figure 8.1: Overfitting

4. **Tensor:**
   It is mathematical object which can be used to describe the physical probabilities like scalars and vectors. Basically it is multi-dimensional array of real numbers. It is collection of the vectors.

5. **Neural Network Training Convergence:**
   During the training phases, the mail goal is to minimize the error rate as well as recall value will be very high. Also make sure the model is generalized well on the validation data.
   When we build the model, we set the epoch parameter before training. In the beginning we don't know how many epoch is good for model. In the neural network it is common approach to visualize the curve of the graph which decides the model convergence. On the basis the value of the epochs is decided. In this project, two graphs are plotted, one is loss vs epoch and another is validation recall, training recall and validation accuracy vs epoch.
   During the training it is expected that the validation loss value will be decreased and validation recall and accuracy will be increased. It is also expected that the network will be converged after training with few number of epochs. If the training is keep on, it will over fit the model and also validation of error will be increased and validation recall value will be decreased.
   As suggested, the main goal is early stop the training model when validation loss to increase or validation recall value starts to decrease.

6. **Mean Reciprocal Rank:**
   It is a statistic measure for evaluating system that return a ranked list of answers to queries, it is order by probabilities of correctness. For single query the reciprocal rank is 1/ rank, whereas rank is position of the cor-

rect answer. If there is no correct answer it returned in query then the reciprocal rank is 0 [17].
For multiple queries Q, the mean reciprocal rank is the mean of the Q reciprocal ranks.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{8.1}$$

7. **How MRR value is calculated for testing data set:**
After training the model, it requires to evaluate it on 10% testing data set. This data set exists in the database. For evaluation, 1000 unique questions took from the q1. These questions used as a queries. Then for each query, top-5 similar questions from database were searched and checked if any of the top-5 plus the query would form a pair that was positive (is_duplciate) in the original data set.

   If the question pairs match in the original set then calculate the rank and break the loop. It is based on first answer reciprocal rank. Then its again start with query 2 and then this process goes on until 1000 queries are not completed. The rank will be calculated with the following way if the results match in the original set then 1 divides with the position of the answer(index) like 1/index. If question pairs are not match in the original set then the rank value will be 0.
   At the end sum all the rank of the queries. The MRR value will be calculated as mention in the above equation 8.1.
   The same approach will be followed to calculate the value of MRR in the following experiments.

8.1 Architecture and Design of baseline Model#1

The baseline model is a 7-layers architecture model, which can be seen as in Figure 8.2. It is siamese network based model, It consists of two input layer, which takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input, in the sequential model used the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture. Which is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text.This layer is shared by both inputs.

Then concatenation layer is used to concatenate the both outputs of the sequential model. Next, normalized the data with the help of batch normalization. Then normalized data is fed to a deep neural network made of one hidden layer with 1024 neurons. This layer used ReLu activation function.
Finally the output layer consists of a single neuron with a sigmoid activation

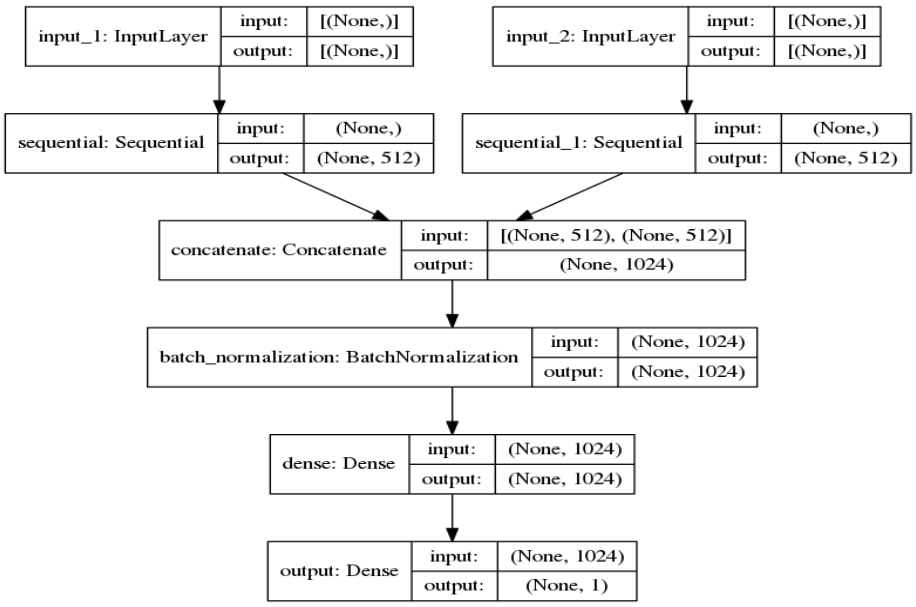


Figure 8.2: Baseline Model 1

function. The deep neural network model was trained with Adam optimizer, binary_crossentropy loss function and to analyse the results, the custom recall method is also used.

8.1.1 Results of baseline Model#1

The model was trained through 4 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some values of the epochs, convergence will start. So, the training will be stopped after 4 epoch. It can be seen in Figures 8.3 8.4 validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.

The Figure 8.3 shows that the Training recall values are increasing while validation accuracy and validation recall values raise to 84.5% in 2 epochs and nearly stop increasing after that. The training is stopped as validation accuracy and recall stop to increase, which is also a way to prevent from over fitting.

After training the model, it is required to evaluate it with testing data set which existed in the database. For that, calculate the MRR value by using the same approach as discussed in the section 7

For this experiment, the MRR value is **38%**. It means that in average the right answer is in third position in the output rank.

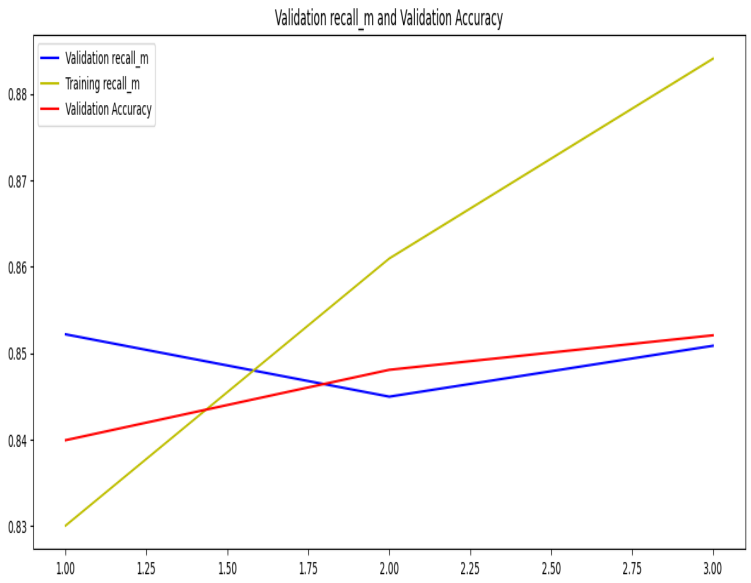


Figure 8.3: Validation recall vs accuracy vs Training recall

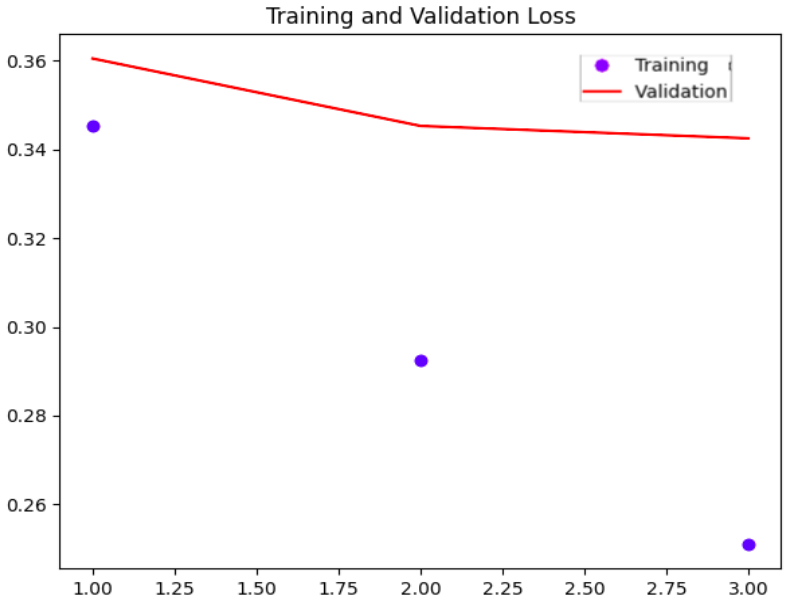


Figure 8.4: Training loss vs Validation loss

8.2 Architecture and Design of baseline Model#2

The baseline model is a 9-layers architecture model, which can be seen in Figure 8.5. It is also siamese network based model, It consists of two input layer. which takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input, in the sequential model by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture which is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text.This layer is shared by both inputs.

Then concatenation layer is used to concatenate the both outputs of the sequential model. Next, normalize the data with the help of batch normalization. Then normalized data is fed to a deep neural network made of one hidden layer with 1024 neurons. This layer uses ReLu activation function.
Next the dropout layer is used which drop or emit 0.1% of the hidden nodes. It reduces the complexity of the training data. Further data will be normalized.
Finally, the output layer consists of a single neuron with a sigmoid activation function. The deep neural network model was trained with Adam optimizer, binary_crossentropy loss function and to analyse the results, the custom recall method is also used.

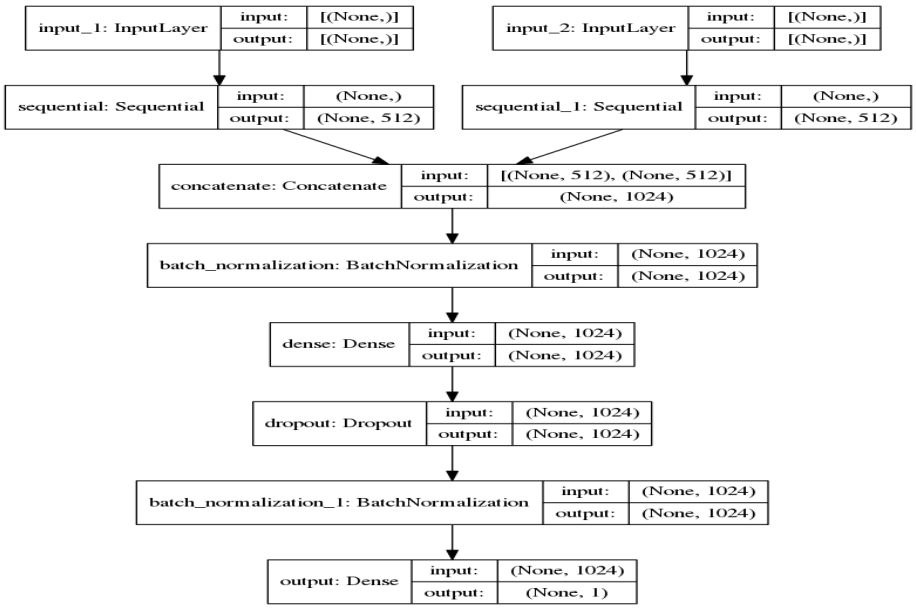


Figure 8.5: Baseline Model 2

8.2.1 Results of baseline Model#2

The model was trained through 12 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some values of the epochs, the convergence will start. So, we stop the training after 12 epoch when validation loss values start to increase. It can be seen from the Figures 8.6 8.7 the validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.
The Figure 8.6 shows that the training recall values are increasing while vali-

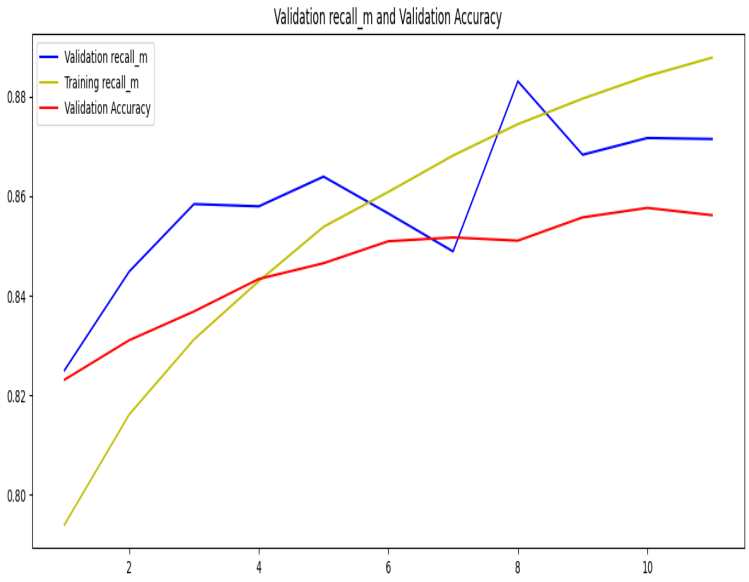


Figure 8.6: Validation recall vs accuracy vs Training recall

dation accuracy raises to 85.5% and validation recall values increase 87% in 10 epochs and nearly stop increasing after that. We stop the training as validation accuracy and recall stop to increase, which is also a way to prevent from over fitting.

After training the model, it is evaluated with testing data set which existed in the database by calculating the MRR values using the same approach as discussed above in the section 7

For this experiment the MRR value is **40%**. It means that in average the right answer is in third position in the output rank.

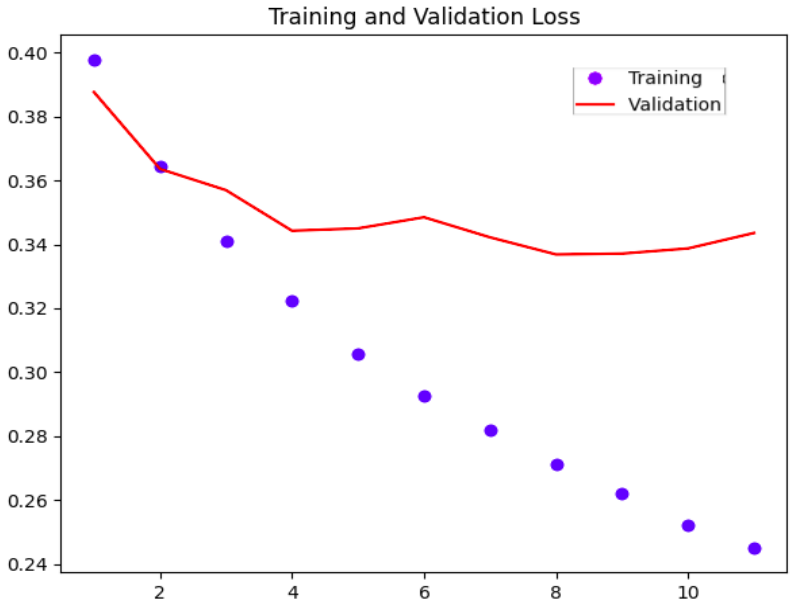


Figure 8.7: Training loss vs Validation loss

8.3 Architecture and Design of baseline Model#3

The baseline model is a 6-layers architecture model, which can be seen as in Figure 8.8. It is also siamese network based model, which consists of two input layer and takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input, in the sequential model by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture. Which is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.
Then concatenation layer is used to concatenate the both outputs of the sequential model. The output of concatenation layer is fed to a deep neural network made of one hidden layer with 1024 neurons. This layer used ReLu activation function.
Finally, the output layer consists of a single neuron with a sigmoid activation function. The deep neural network model was trained with Adam optimizer, binary_crossentropy loss function and to analyse the results, the custom recall method is also used.

8.3.1 Results of baseline Model#3

The model was trained through 4 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some

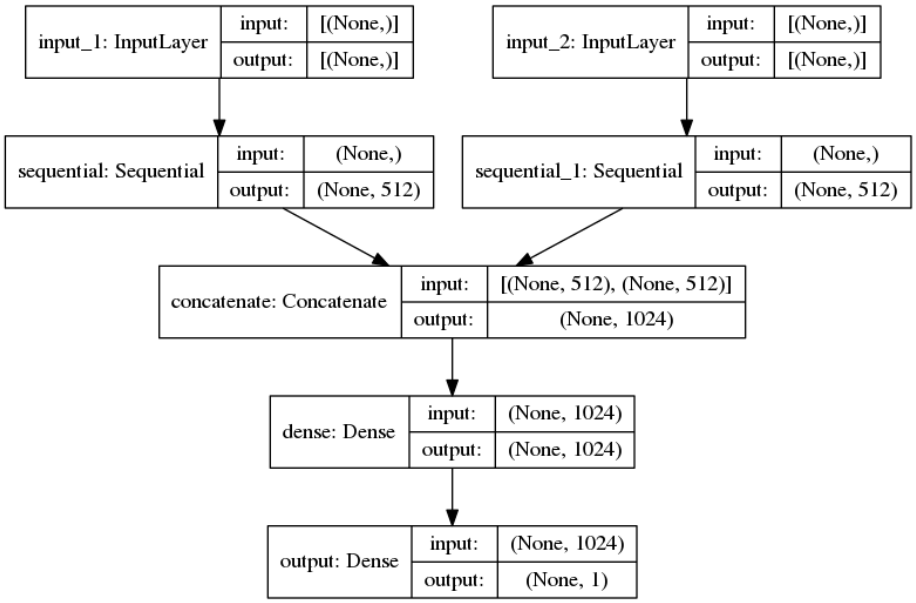


Figure 8.8: Baseline Model 3

values of the epochs, convergence will start. So, we stop the training after 4 epoch when validation loss values start to increase. it can be seen from the Figures 8.9 8.10 validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.
The Figure 8.9 shows that the training recall values are increasing while validation accuracy raises to 85.5% and validation recall values increase 84.5% in 3 epochs and nearly stop increasing after that. It has been observed that after 3 epochs the validation accuracy and recall stop to increase. So, we stop the training which is also a way to prevent from over fitting.

For this experiment the MRR value is **44%**. It means that in average the right answer is almost in second position in the output rank.

8.4 Architecture and Design of baseline Model#4

The baseline model is a 6-layers architecture model as shown in Figure 8.11. It is also siamese network based model, It consists of two input layer. Which takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input, in the sequential model by using the hub keras layer which load the universal sentence encoder pre-trained model. It is based on DAN architecture which is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text.

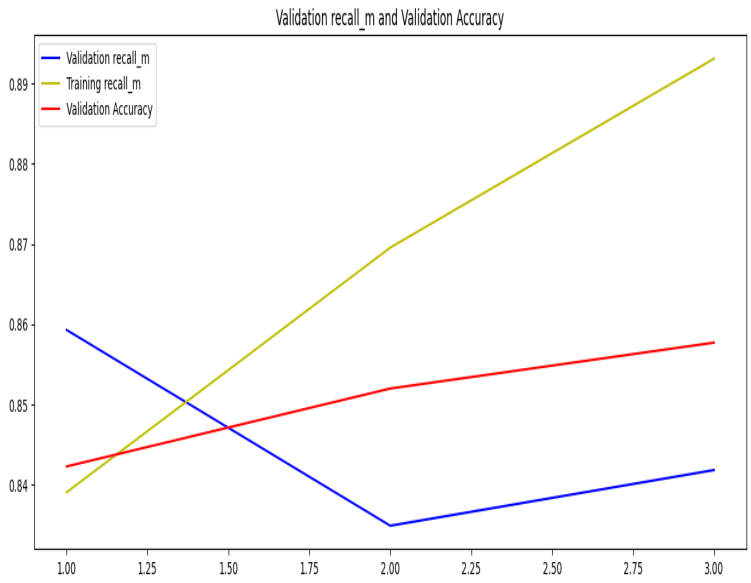


Figure 8.9: Validation recall vs accuracy vs Training recall

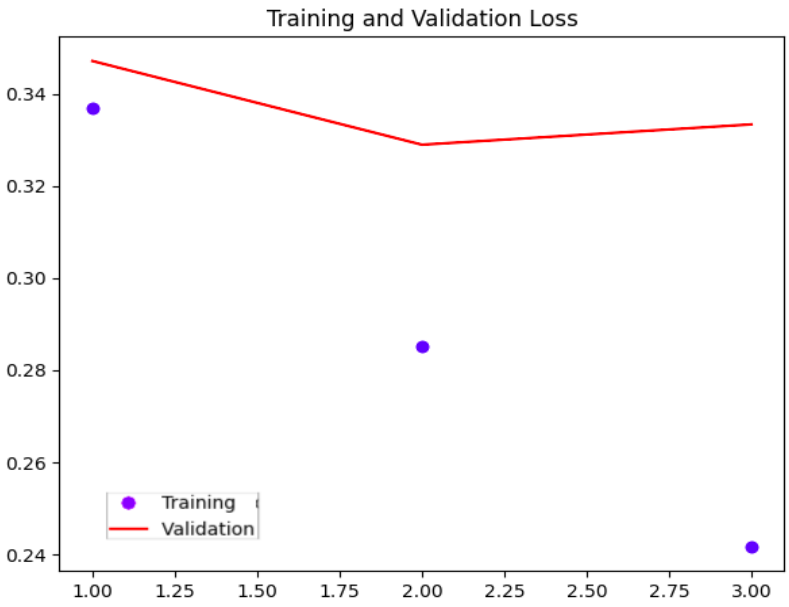


Figure 8.10: Training loss vs Validation loss

This layer is shared by both inputs.

The output of each sequential model is fed to a deep neural network, each

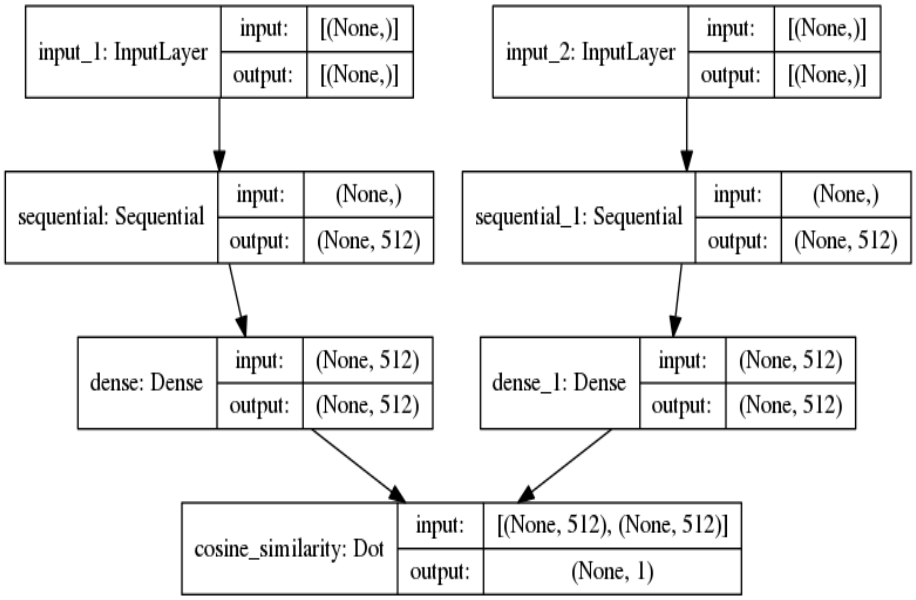


Figure 8.11: Baseline Model 4

hidden layer has 512 neurons. These layers use ReLu activation function. Finally, the Dot layers which take the inputs from both dense layers and then calculate the cosine similarity between them.

8.4.1 Results of baseline Model#4

The model was trained through 3 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some values of the epochs, convergence will start. So, we stop the training after 3 epoch when validation loss values start to increase. It can be seen in Figures 8.12 8.13 validation loss values are decreasing and validation recall and accuracy value are increasing. It means the training model is not over fitting.

From the Figure 8.12 it can be seen that the training recall values are increasing while validation accuracy raises to 83% and validation recall values increased to 81% in 2 epochs and nearly stop increasing after that. It observed that after 3 epochs the validation accuracy and recall stop to increase. So, we stop the training to prevent from over fitting.

For this experiment the MRR value is **25%**. It means that in average the right answer is in fourth position in the output rank.

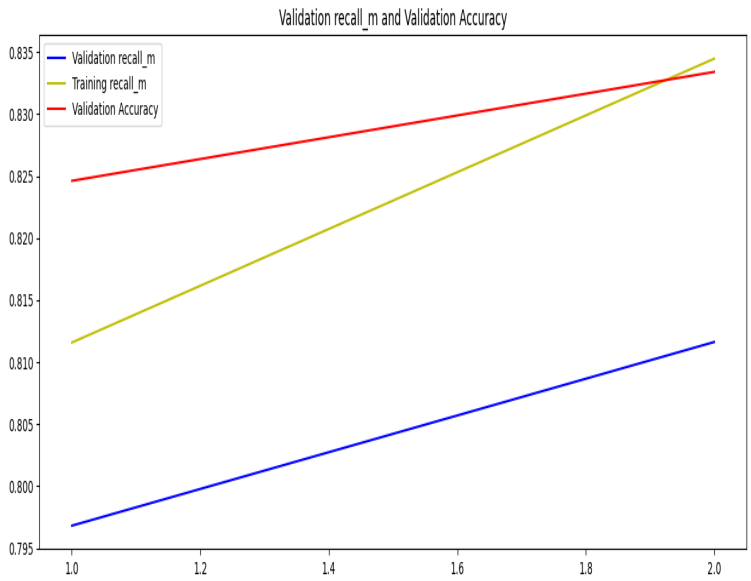


Figure 8.12: Validation recall vs accuracy vs Training recall

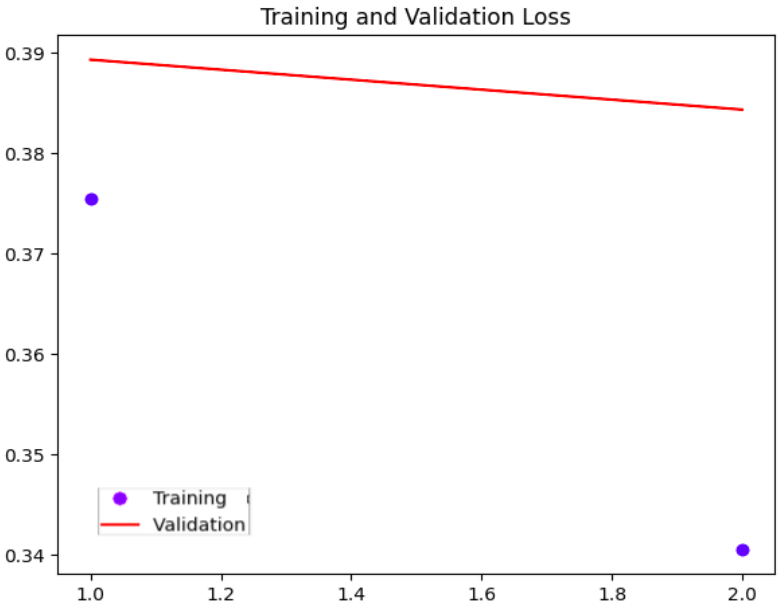


Figure 8.13: Training loss vs Validation loss

8.5 Architecture and Design of baseline Model#5

The baseline model is a 4-layers architecture model as shown in Figure 8.14. It is also siamese network based model, it consists of two input layers which take the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input by using the hub keras layer which loads the universal sentence encoder pre-trained model. It is based on DAN architecture.
The hub keras layer takes the parameter trainable. If we pass the value True to it. Then it unfreeze the layers of the base model. So, during training, the base model weights will also be updated. This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.
Finally, the Dot layer computes the cosine similarity between sentence embed-

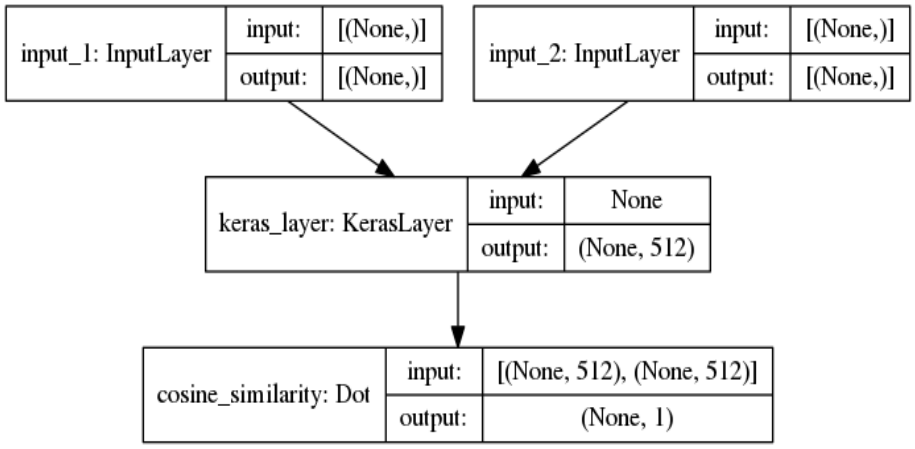


Figure 8.14: Baseline Model 5

ding vectors.

8.5.1 Results of baseline Model#5

The model was trained through 1 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model.
This one epoch takes 16h for completing the training. Because the model is fine tuned with low learning rate 1e-5. Also, in our machine GPU is not installed, for that reason in this experiment it is not possible to train it with more epochs values.
The results of one epoch are very impressive, The training recall value is 94% and validation recall value is 92%. It showed that these values are very high as compared to the validation accuracy which is 68%.

For this experiment the MRR value is **58%**. It means that in average the right answer is in second position in the output rank.

8.6 Architecture and Design of baseline Model#6

The baseline model is a 7-layers architecture model, which can be seen in Figure 8.15. It is also siamese network based model, it consists of two input layer. which takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture.
This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.
The output of each sequential model is fed to LSTM layers, each lstm layer

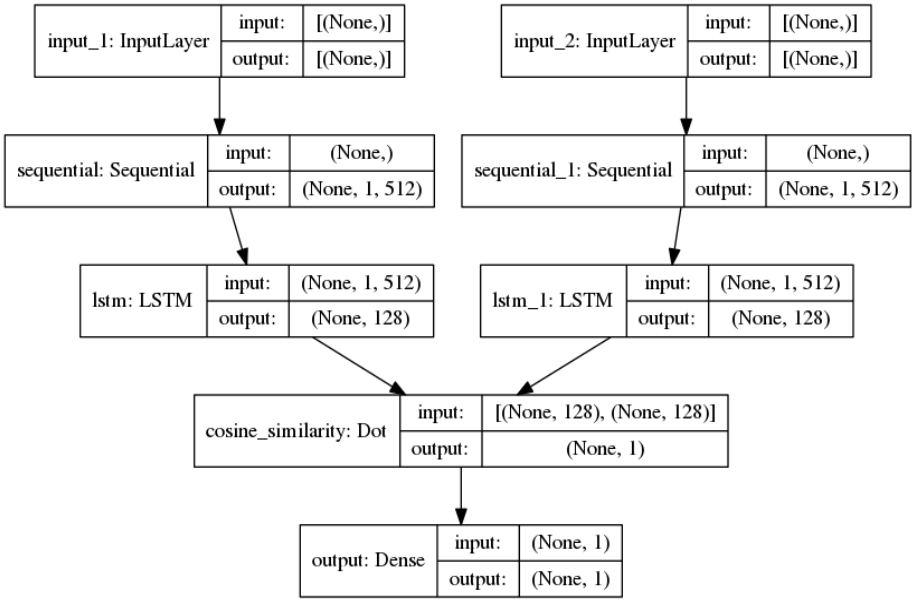


Figure 8.15: Baseline Model 6

has 128 neurons.
Then the Dot layers take the inputs from both lstm layers and then calculate the cosine similarity between them.
Finally the output layers consist of a single neuron with a sigmoid activation function. The deep neural network model was trained with Adam optimizer, binary_crossentropy loss function and to analyse the results, the custom recall method is also used.

8.6.1 Results of baseline Model#6

The model was trained through 9 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some values of the epochs, convergence will start. So, we stop the training after 9 epoch when validation loss values start to increase. It can be seen from the Figures 8.16 8.17 that the validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.

From the Figure 8.16 it can be seen that the training recall values are increas-

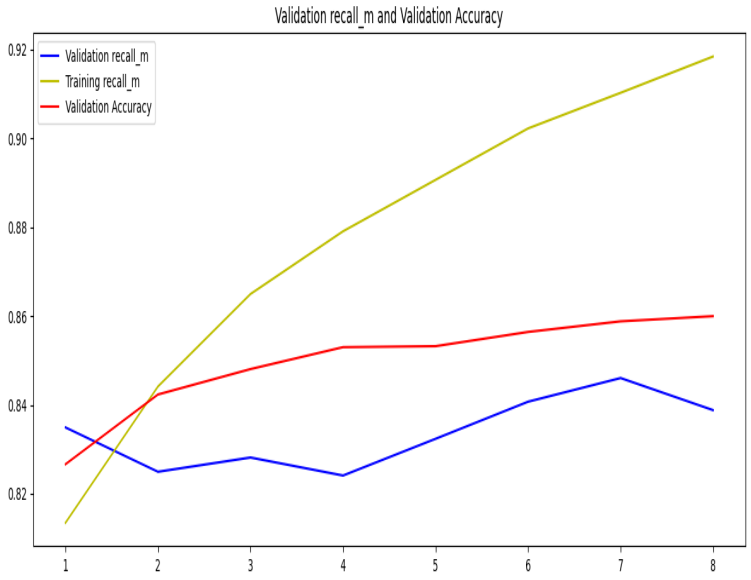


Figure 8.16: Validation recall vs accuracy vs Training recall

ing while validation accuracy raises to 86% and validation recall values increase to 84% in 7 epochs and nearly stop increasing after that. It observed that after 7 epochs the validation accuracy and recall stop to increase. Even validation recall values start to decrease. So, we stop the training to prevent from over fitting.

For this experiment the MRR value is **34%**. It means that in average the right answer is in third position in the output rank.

8.7 Architecture and Design of baseline Model#7

The baseline model is a 5-layers architecture model, which can be seen as in Figure 8.18. It is also siamese network based model which consists of two input layers and takes the batch of sentences in a 1-D tensor of strings as input.

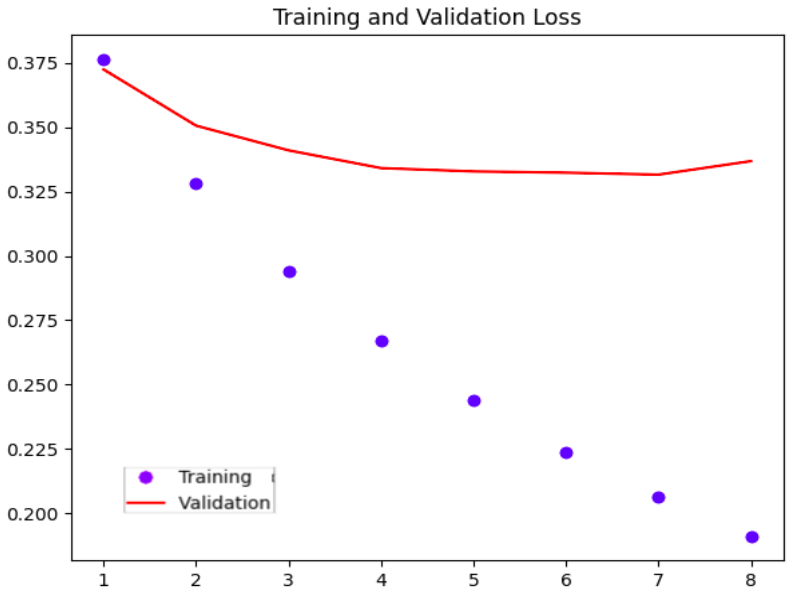


Figure 8.17: Training loss vs Validation loss

The sequential model takes the input by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture. The hub keras layer takes the parameter trainable. If we pass the value True to it. Then it unfreeze the layers of the base model. So, during training the base model weights will also be updated.
This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.

Then the Dot layers which compute the cosine similarity between sentence embedding vectors. Finally, the output layers consist of a single neuron with a sigmoid activation function. The deep neural network model was trained with SGD optimizer, binary_crossentropy loss function and to analyze the results, the custom recall method is also used.

8.7.1 Results of baseline Model#7

The model was trained through 26 epochs and batch size is 64, which indicates the number of passes of the entire training data set in the model. After some values of the epochs, convergence will start. So, we stop the training after 26 epoch when validation loss values start to increase. It can be seen in the Figures 8.19 8.20 the validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.

From the Figure 8.19 it can be seen that the training recall values are in-

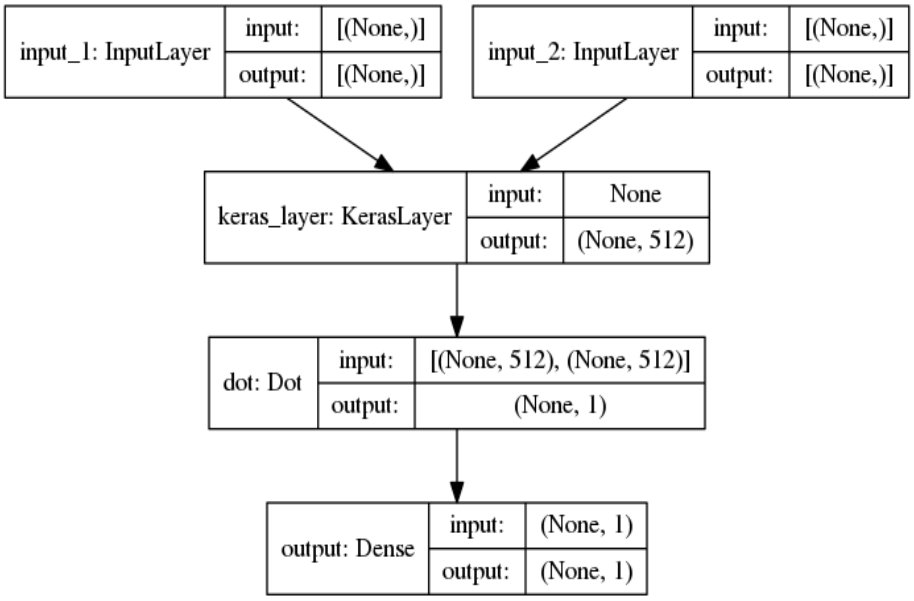


Figure 8.18: Baseline Model 7

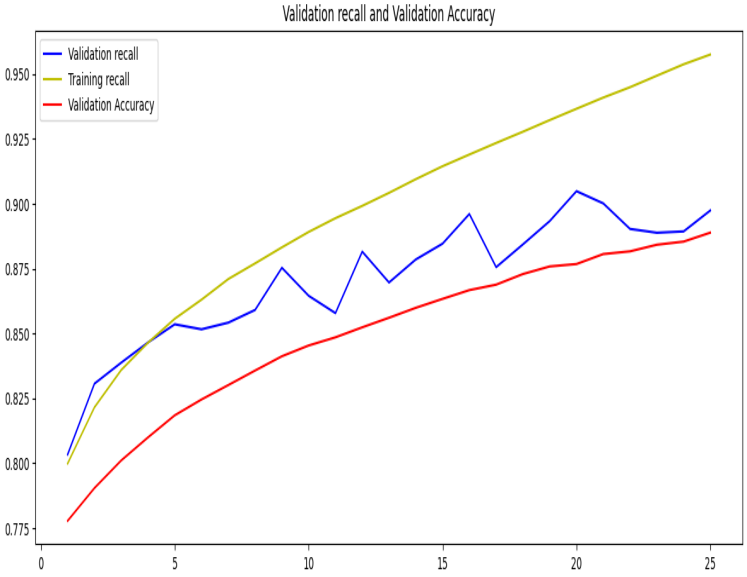


Figure 8.19: Validation recall vs accuracy vs Training recall

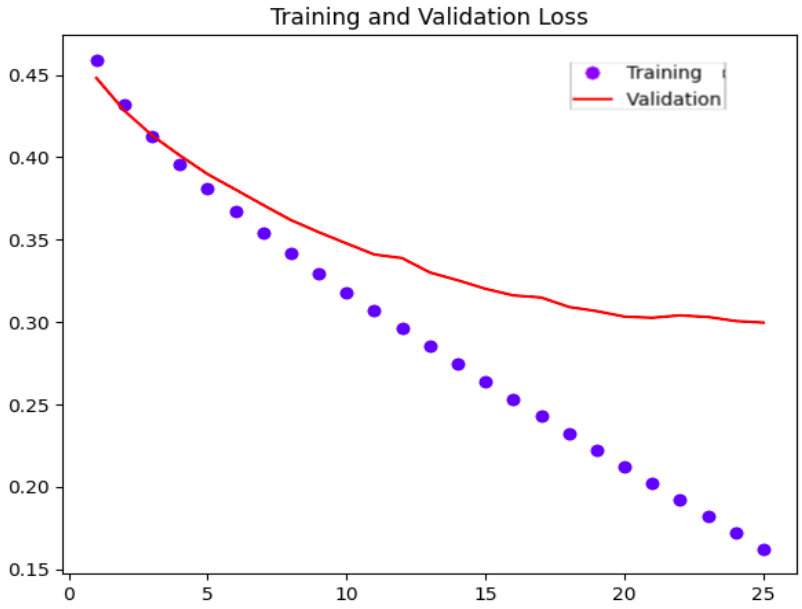


Figure 8.20: Training loss vs Validation loss

creasing while validation accuracy raises to 87% and validation recall values increase to 88% in 23 epochs and nearly stop increasing after that. It has been observed that after 23 epochs the validation accuracy and recall are increasing very slowly. So, we stop the training to prevent from over fitting.

For this experiment the MRR value is **53%**. It means that in average the right answer is in second position in the output rank.

8.8 Architecture and Design of baseline Model#8

The baseline model is a 4-layers architecture model, which can be seen as in Figure 8.21. It is also siamese network based model which consists of two input layer and takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture.
The hub keras layer takes the parameter trainable. If we pass the value True to it. Then it unfreeze the layers of the base model. So, during training the base model weights will also be updated. This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.
Finally, the Dot layers compute the cosine similarity between sentence embed-

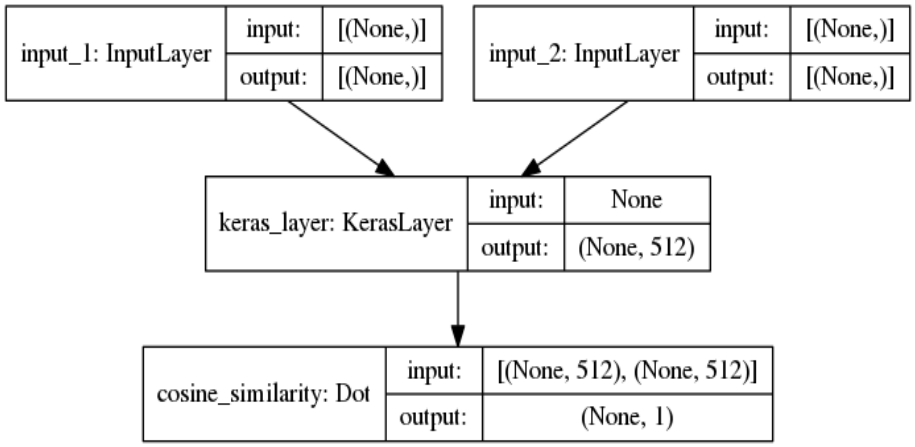


Figure 8.21: Baseline Model 8

ding vectors. The deep neural network model was trained with SGD optimizer, binary_crossentropy loss function and for analyses the results the custom recall method is also used.

8.8.1 Results of baseline Model#8

The model was trained through 6 epochs and batch size is 64, which indicated the number of passes of the entire training data set in the model. After some values of the epochs, convergence will start. So, we stop the training after 6 epoch when validation loss values start to increase. It can be seen from the Figures 8.22 8.23 that the validation loss values are decreasing and validation recall and accuracy values are increasing. It means the training model is not over fitting.

From the Figure 8.22 it can be seen that the training recall, validation recall and validation accuracy values decrease because validation loss values are increasing. After 2 epochs when validation loss value is decreasing then these values will be increased.

Validation accuracy raise to 77% and validation recall values increase to 89% in 4 epochs and nearly stop increasing after that. It has been observed that after 4 epochs, the validation accuracy and recall increasing very slowly. So, we stop the training to prevent from over fitting.

For this experiment the MRR value is **50%**. It means that in average the right answer is in second position in the output rank.

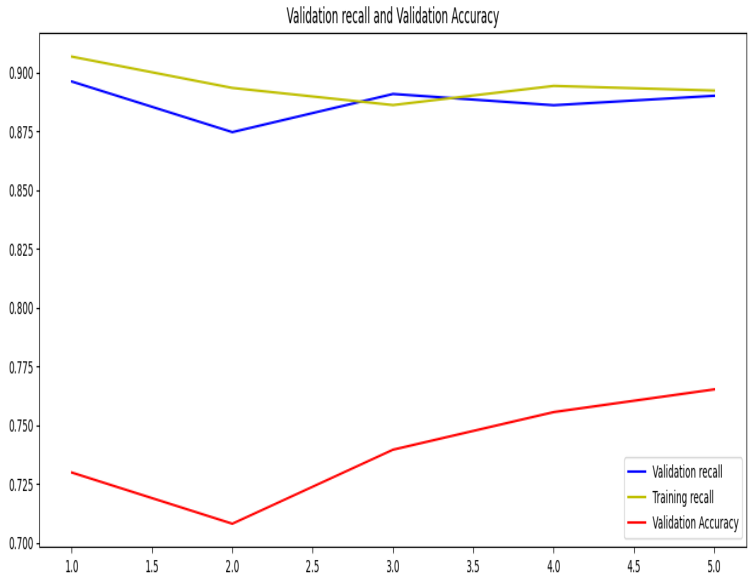


Figure 8.22: Validation recall vs accuracy vs Training recall

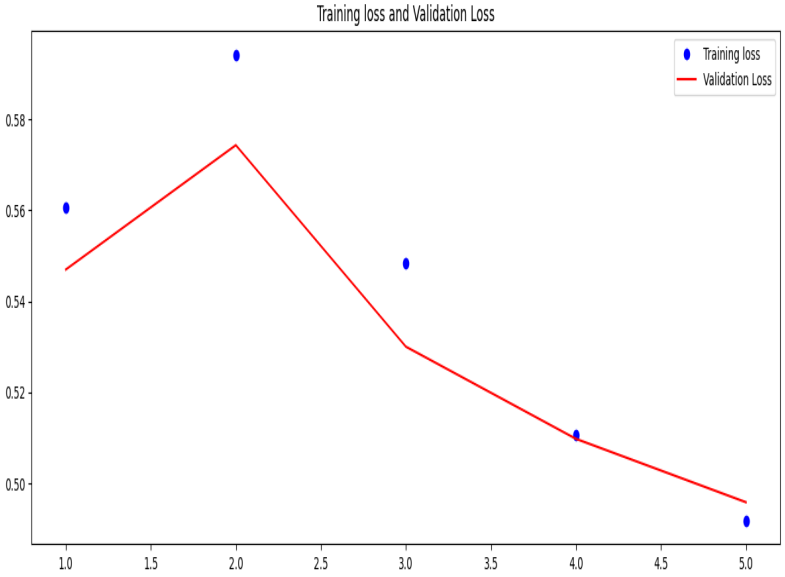


Figure 8.23: Training loss vs Validation loss

8.9 Architecture and Design of baseline Model#9

The baseline model is a 4-layers architecture model, which can be seen in Figure 8.24. It is also siamese network based model which consists of two input layer and takes the batch of sentences in a 1-D tensor of strings as input.
The sequential model takes the input by using the hub keras layer which loaded the universal sentence encoder pre-trained model. It is based on DAN architecture.
This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.

Finally the Dot layers computed the cosine similarity between sentence em-

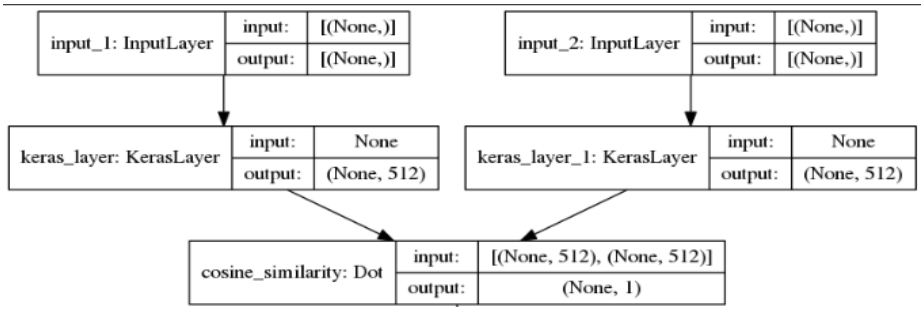


Figure 8.24: Baseline Model 9

bedding vectors.

8.9.1 Results of baseline Model#9

In this model, the deep learning technique is not used. It just embedding the text and calculate the cosine similarity between the embedding vector. No hidden layers and fine-tuning approach are used, that's why, there is no need to train the model.
In this model we only use the base model (universal sentence encoder). It is evaluated on testing data set which is existed in the database. For that the MRR value is calculated by using the same approach as we discuss in this section 7

For this experiment the MRR value is **62%**. It means that in average the right answer is in second position in the output rank.

8.10 Baseline Model#10

Universal sentence encoder multilingual
The present project is web based and it supports the multi languages like En-

glish and Spanish. If user type the question in Spanish language then the result will be also in Spanish. Due to which the pre-trained model (multilingual universal sentence encoder) is used.
It embedding the text from 16 languages into single semantic space [24].It also embedding the text from Spanish language.
It included the training on multiple tasks across language. The multiple task training based on the Multi-task Dual-Encoder Model [5]. It is based on CNN(convolutional neural network) architecture.

Google Translate service
Translate the testing data set into Spanish language and store it in the database. For translation, use the Google cloud translate API. It translates a large volume of text in asynchronous batch mode. This service is paid.
Charges on the basis of the character of the text. For example, if you send 575,000 characters for processing within a month, you are charged $1.50. The first 500,000 characters are free, and then you are charged for the additional 75,000 characters.[1]

Architecture and Design
The baseline model is a 4-layers architecture model, which can be seen as in Figure 8.25. It is also siamese network based model which consists of two input layers and takes the batch of sentences in a 1-D tensor of strings as input. These sentences are in Spanish language.
The sequential model takes the input by using the hub keras layer which loaded the universal sentence encoder multilingual pre-trained model. As mentioned above, it is based on CNN architecture.
This layer is used for encoding sentences into embedding vector. The output of this layer is 512 fixed-size dimensional vector for the text. This layer is shared by both inputs.
Finally the Dot layers compute the cosine similarity between sentence embed-

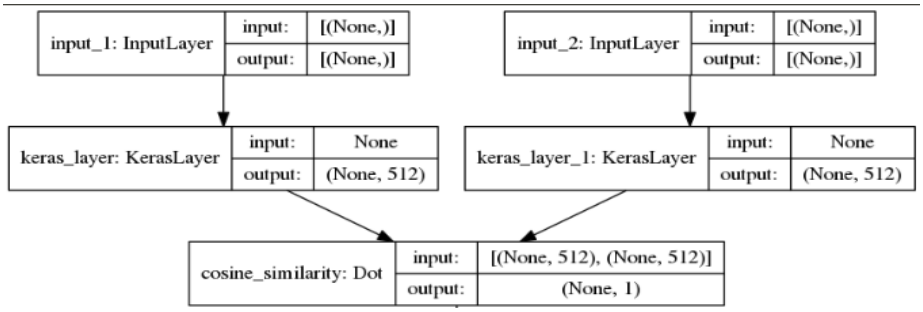


Figure 8.25: Baseline Model 10

ding vectors.

[1] https://cloud.google.com/translate/pricing

8.10.1 Results of baseline Model#10

As in this model, the deep learning technique is not used. It just embedding the text and calculate the cosine similarity between the embedding vector. No hidden layers and fine-tuning approach have been used that's why, there is no need to train the model.
In this model, only the base model (universal sentence encoder multilingual) is used. It is also evaluated on testing data set which is existed in the database. For that, the MRR value is calculated by using the same approach as discussed in this section 7

For this experiment the MRR value is **50%**. It means that in average the right answer is in second position in the output rank.

8.11 Development of the Web

This section detailed implementation of the project, It include the snapshots of the application features.

8.11.1 Login page:

Login page is designed with the help of the HTML. You can in the Figure 8.26. Django provide the authentication to the user. When user click on the Login button, the authenticate class is used to check whether the user-name and password exist in the database user table. And with the help of LoginView class of Django, login the user by user-name and password.

8.11.2 Similar tickets shown:

When the user open the create ticket page, and in the text field type the question similar tickets will be shown
The database has total number tickets are 55376. To fetch the similar tickets from database, Baseline model#9 is used because it has good value of the MRR. From Figures 8.27 8.28 you can see that most related (similar) tickets are shown. These tickets are semantic similar with the query.

When the web is in Spanish language mode, when the user type the question similar question will be shown.
The database has Spanish language tickets are 5000. To fetch the similar tickets from database, Baseline model#10 is used because for embedding, this model use the universal sentence encoder multilingual. Also it has good value of MRR. From Figures 8.29 8.30 you can see that most related (similar) tickets are shown. These tickets are semantic similar with the query.
As we observe that in Spanish language results are not good as compare to English language. Because when translate the dataset from English to Spanish

Figure 8.26: Login page

Figure 8.27: similar tickets shown

language the results accuracy are not 100% accurate.
From the experimental results it is notice that, the MRR value with Spanish language are low as compare to the English language.

8.11.3 Create ticket

With the help of HTML, create ticket page is designed as you can see in the Figure 8.31. User (client, agent or admin) can write the description of the question in detailed. He can select the priority and ticket type of the ticket.
When the ticket is created the system automatically assigned it to the agents which has less number of tickets.

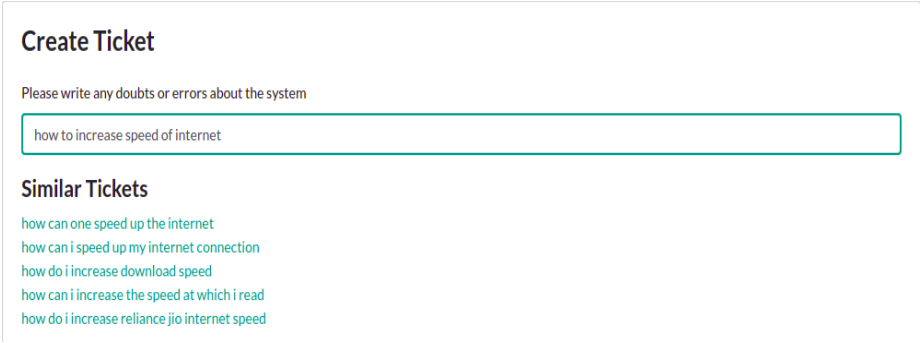


Figure 8.28: similar tickets shown

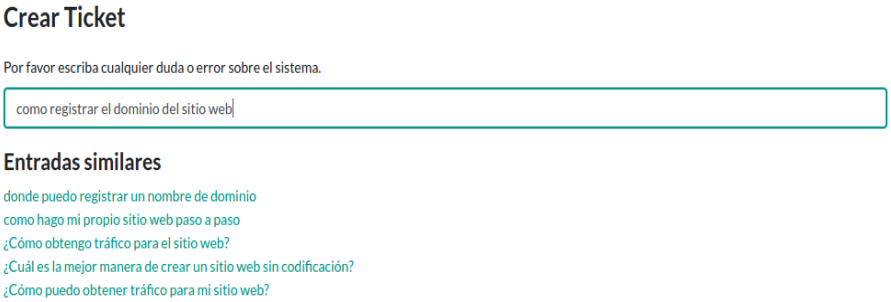


Figure 8.29: similar tickets shown

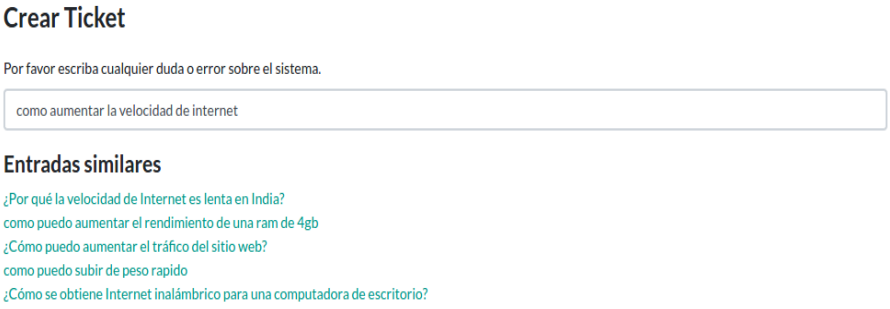


Figure 8.30: similar tickets shown

8.11.4 View assigned tickets

List of the tickets can be see in the Figure 8.32. The user (agents or admins) can see only those tickets which assigned to them.

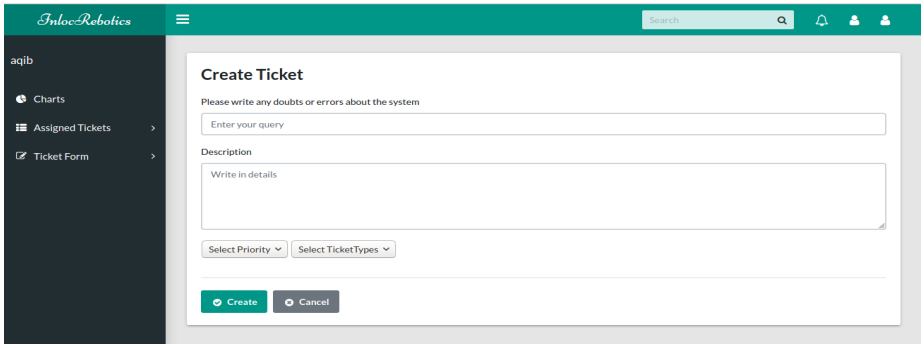


Figure 8.31: Create Ticket

From the search field user can find the particular tickets. With the help of pagination can change the page.

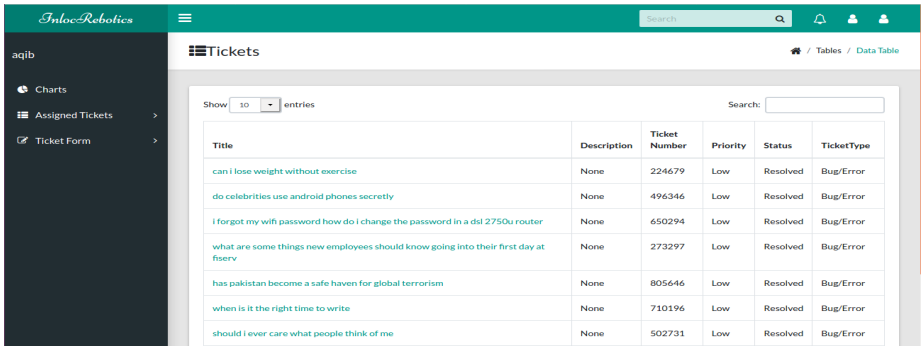


Figure 8.32: List of Tickets

8.11.5 Ticket details show

When user click on the ticket, it shows the ticket details as you can see in the Figure 8.33. In the Figure you can also observe the comments about the ticket. These comments related to the communication between the user (agent or admin) and the client through the email channel.

8.11.6 Charts

From the chart page user (admin, agent) can see the number of tickets on the basis of the status, tickets types and priorities. You can also see from the Figures 8.34 8.35.

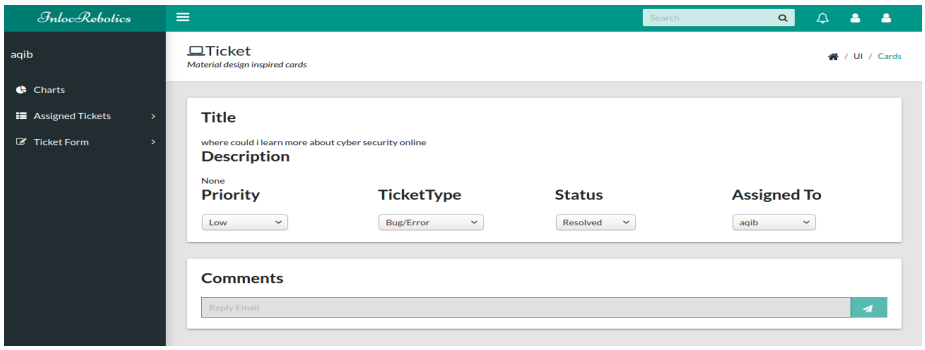


Figure 8.33: Ticket details

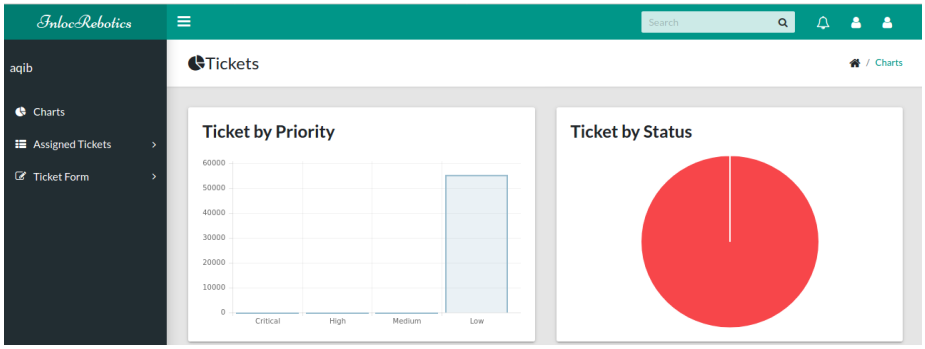


Figure 8.34: Ticket charts about ticket type and status

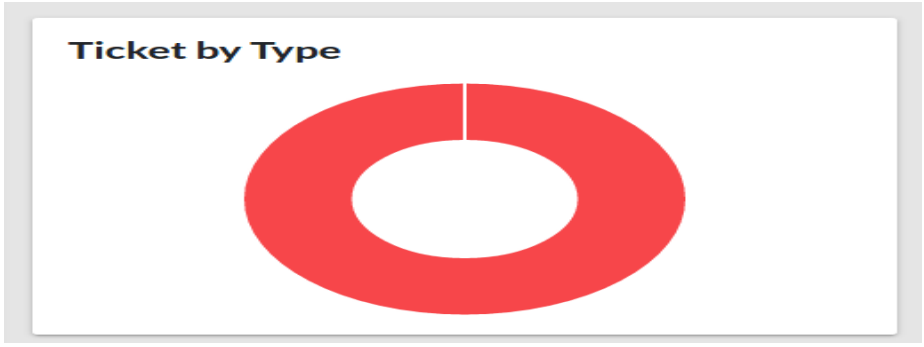


Figure 8.35: Ticket charts about priorities

8.11.7 View all tickets

Admin can see all the tickets in the system which are assigned to other team member (agents or admin) as you can see in the Figure 8.36.

Figure 8.36: Admin list of tickets

Chapter 9

Conclusion and Future work

In this project we try to solve the problem of ticketing system with automatic suggestions. For automatic suggestions of the similar tickets, the problem is solved with the help of artificial neural network.
In the neural network nutshell, different concepts of ANN (artificial neural network) were discussed. Also, different techniques were used to train the neural network.
The problem was solved with the help of NLP technique. So, computer will become capable of understanding the contents and context of the documents or texts. For this, universal sentence encoder was used for embedding. For similarity between the texts, the cosine similarity was used.
In the experiment and result sections, different architecture of the models with different hyper-parameters were used. The results of models were reviewed after training.
The following results got from the experiments.

Baseline Models	**Validation Recall**	**MRR**
Model1	85%	38%
Model2	87%	40%
Model3	84.3%	44%
Model4	81%	25%
Model5	92%	58%
Model6	84%	34%
Model7	88%	53%
Model8	89%	50%
Model9		62%
Model10		50%

From the above table, it can be seen that Baseline Model 9 and Model 10 have not validation recall values because these model are not using the hidden layers. So that's why these models will not be trained. So, in these models we only use the pre-trained model. Baseline model 9 used the pre-trained model

universal sentence encoder and Baseline model 10 used pre-trained model universal sentence encoder multilingual.
The above table shows that with baseline model 5 we get the good results even it is trained with one epochs. If we have GPU, in the future we can train this model with more epochs values. So, the MRR value will be improved. Because it will learn from the dataset.
In the experiments, it is also observed that optimizer Adam and loss function binary_crossentropy gave good results as compared to other optimizers and loss functions.
For the present project the model was chosen on the basis of the MRR values. From the above table it can be seen that baseline model 9 has high value of the MRR(62%) and validation recall (92%). It means that in average the right answer is in second position of the list. So, this model is used in the present project, when the web in English language mode.
When the web is in Spanish language mode, then the baseline model 10 was selected. Because it used the pre-trained model universal sentence encoder multilingual. It's MRR value is 50%. It means that, in average the right answer is in second position of the list.

Bibliography

[1] Pratima Rao Akinepally. *Investigating Performance of Different Models at Short Text Topic Modelling*. 2020.

[2] Akanksh Basavaraju et al. "A machine learning approach to road surface anomaly assessment using smartphone sensors". In: *IEEE Sensors Journal* 20.5 (2019), pp. 2635–2647.

[3] Daniel Cer et al. "Universal sentence encoder". In: *arXiv preprint arXiv:1803.11175* (2018).

[4] Dhivya Chandrasekaran and Vijay Mago. "Evolution of semantic similarity—a survey". In: *ACM Computing Surveys (CSUR)* 54.2 (2021), pp. 1–37.

[5] Muthuraman Chidambaram et al. "Learning cross-lingual sentence representations via a multi-task dual-encoder model". In: *arXiv preprint arXiv:1810.12836* (2018).

[6] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[7] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[8] Xiaoqi Jiao, Fang Wang, and Dan Feng. "Convolutional neural network for universal sentence embeddings". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 2470–2481.

[9] Sun Kim et al. "Bridging the gap: Incorporating a semantic similarity measure for effectively mapping PubMed queries to documents". In: *Journal of biomedical informatics* 75 (2017), pp. 122–127.

[10] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR. 2014, pp. 1188–1196.

[11] Omer Levy and Yoav Goldberg. "Dependency-based word embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2014, pp. 302–308.

[12] Oren Melamud, Jacob Goldberger, and Ido Dagan. "context2vec: Learning generic context embedding with bidirectional lstm". In: *Proceedings of the 20th SIGNLL conference on computational natural language learning*. 2016, pp. 51–61.

[13] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[14] Saif M Mohammad and Graeme Hirst. "Distributional measures of semantic distance: A survey". In: *arXiv preprint arXiv:1203.1858* (2012).

[15] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. "Unsupervised learning of sentence embeddings using compositional n-gram features". In: *arXiv preprint arXiv:1703.02507* (2017).

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[17] Dragomir R Radev et al. "Evaluating Web-based Question Answering Systems." In: *LREC*. Citeseer. 2002.

[18] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[19] Tobias Schnabel et al. "Evaluation methods for unsupervised word embeddings". In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 298–307.

[20] Ola Spjuth, Jens Frid, and Andreas Hellander. "The machine learning life cycle and the cloud: implications for drug discovery". In: *Expert opinion on drug discovery* 16.9 (2021), pp. 1071–1079.

[21] Hossam H Sultan, Nancy M Salem, and Walid Al-Atabany. "Multi-classification of brain tumor images using deep neural network". In: *IEEE Access* 7 (2019), pp. 69215–69225.

[22] Julien Tissier, Christophe Gravier, and Amaury Habrard. "Dict2vec: Learning word embeddings using lexical dictionaries". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 254–263.

[23] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[24] Yinfei Yang et al. "Multilingual universal sentence encoder for semantic retrieval". In: *arXiv preprint arXiv:1907.04307* (2019).

[25] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems* 27 (2014).