SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

FINAL YEAR PROJECT

SCSE19-0004

FAQ CHATBOT WEB FRAMEWORK FOR RESPONSE COMPARISONS AND PERFORMANCE ANALYSIS

Russell Chua Kui Hon, U1720526F

Supervisor: Assoc. Professor Chng Eng Siong

Examiner: Dr Smitha Kavallur Pisharath Gopi

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Associate Professor Chng Eng Siong for the continuous support throughout the project research and development, for his patience, motivation and enthusiasm. His guidance helped me make informed decision and help path a clear objective for me to achieve the final product. He also took his own time to guide me on report writing and presentation organization. For that I could not have imagined having a better supervisor for the duration of my Final Year Project.

I would also like to specially thank Multimedia and Interactive Computing Lab researchers Ms Vu Thy Ly and Mr Mai Trung Duc for taking their precious time off to advise me this project. My sincere thanks to Ms Ly for the reviewing my Chatbot Framework and providing me with the API tokens. And Mr Duc for guiding me with the speech to text development and Docker deployment process.

I would like to thank PHD student Mr Andrew Koh for facilitating the weekly meeting sessions, answering my queries and providing me with the data sets to work with. At last Computer Science Student Yap Boon Peng for providing me his FAQ question permutations.

List of Figures

List of Tables

Abstract

The Objective of this implementation is to develop a Chatbot framework to provide a modular platform for a given set of Question Answer Matching Engine. This framework aims to provide an all rounded input solution to assess response accuracy and performance comparison with existing Chatbot services. Chatbot serves various purposes ranging from FAQs, helpdesk to interactive smart home systems. There are currently no available web platforms that provide Chatbot analysis as different Chatbots serves various purposes and each service serves its own unique purpose, therefore there are currently no benchmark Chatbot model to make comparisons.

The challenge in this implementation is to be able to deliver a proof of concept of a modular Chatbot comparison service where Chatbot solutions can be segregated to the respective categories and tested with one another to determine the level of correctness with optimal query matching time. This platform hopes to help Chatbot engineers improve the quality of services and better understand the needs of the end users.

For the context of this implementation, the Chatbot category will be targeting FAQs. This Project will work closely with the Ministry of Social and Family Development's Baby Bonus as a basis for the FAQ Chatbot service. Using a production grade Chatbot service Ask Jamie developed by Govtech Singapore, the implemented framework will be using Ask Jamie as a benchmark to make comparisons with Nanyang Technological University MICL Lab's QA matching engine and various other open source Chatbot engines.

This report contains my approach to develop a full stack application with a ReactJS frontend and a NodeJS backend. Frontend includes a single dashboard with three components. The first component consists of three input methods, text, speech and audio file. The second component will include a collection of four Chatbot services, Ask Jamie, Dialogflow, MICL and a self-implemented text classification model. The last component will showcase a batch query feature for analysis comparison.

The backend will consist of five Application programming interfaces that connects the following Chatbot services, MICL QA Model, ask Jamie API, a Dialogflow API Client, Text Classification model and AISG Speech transcription API.

This report will also include the miscellaneous tasks of data pre-processing on raw data sets, construction of training and testing inputs with Natural Language Processing(NLP) tools. To support the functionality of each application components, system architecture designs and dialog maps will be included for better visualization.

Table of Contents

1 Introduction

A Chatbot is a software that specializes in simulating conversions with human users. Chatbots are designed with Artificial intelligence with access to Natural Language Processing tools and services. With the integration of Natural Language Processing capabilities and a large database of words, Artificial intelligence is able to serialize and decipher intentions and entities in a given sentence and understand a human's natural language [1].

Chatbots are regarded as one of the most sophisticated and powerful tool to converse with humans, and widely used to automate and simplify interactions and most often facilitate FAQ services for industries such as banks, which are required to handle many requests per minute.

The project will be making use of MSF's Baby bonus services as a basis for the development of a FAQ chat bot for NTU. MSF's data has been extremely critical and essential to the active development of natural language processing and machine learning solutions.

To cater for FAQs and queries, Baby Bonus Online provides a virtual assistant named Ask Jamie [2] to cater for quick automated responses for consumer's needs. And because of the broad services offered by Baby Bonus Online, it is a platform for most people in their adulthood and with diverse background, varied English competency levels and linguistic capabilities, Baby Bonus must be able to handle improper contextual use of English language or the common Singlish that most Singaporean are used to communicating with today.

1.1 Motivation

The aim of the project is to build a Client Server Chatbot Web framework that can store a set of open sources or experimental Chatbot services for live comparisons. Three main input methods are used in the implementation so as to simulate different types of real life scenarios, the input methods are text input, Speech input and file uploads; this includes audio files as well as text files containing batches of query inputs.

The focus to this application to use the framework as a test bed to determine experimental Chatbot performance against existing Government developed Virtual Assistant solution, Ask Jamie. With sophisticated Question Answer matching techniques developed by the MICL lab and Google's open source Chatbot frameworks like Dialogflow, this Chatbot platform is developed to be more versatile and scalable to accommodate for different FAQ context.

The Chatbot will consist of text interface and speech input interface to accommodate a broader user base. Application will also include a comparison of response between various QA engines to assess competency and reliability when dealing with a variety of queries of varying contexts.

The development of the application would be a deployment of an accurate and interactive FAQ Chatbot Framework for use by Nanyang Technological University. For this Proof of Concept, frequently asked questions of Baby Bonus retrieved from the Ministry of Social and Family development will be used to assess the capabilities of Multimedia Interactive Computing Lab's Question Answer matching system.

1.2 Implementation Considerations

In this report, some considerations were carefully planned over the period of the project to prepare the development of the Chatbot framework. These considerations are divided into three components, the view, controller and model.

The first consideration to this development was to build a web framework that could handle a considerable number of Chatbot services and also be able to prevent wasteful DOM manipulations since page renders will occur very often due to the multi display architecture. This will represent the view component of the application

Second, a light-weight API middleware service must be used to handle data intensive and simultaneous requests and as well as for persistent and real-time data streaming since the implementation will attempt to simulate a large chatroom environment. The service has to be able to scale easily as data overhead increases. This will represent the controller component.

Third, to handle response comparisons and miscellaneous pre-processing of query text permutations used for training of open source Chatbot frameworks, a programming language that offers extensive Natural Language Processing libraries. And since this language will be handling request for response processing and comparisons.

Lastly, the model would be the source of data. That would be the Chatbot API services that will be contacted. Therefore, extensive research has to be done to find suitable Chatbot services that can be contacted by the controller.

2 Chatbot Framework Research and Experiments

2.1 Overview

A Web Framework is a software created to support the development of web services and web applications. They involve the creation, development and the deployment of a functional website. Web frameworks also leverage on resources such as web services and API to deliver a better client experience in terms of feature adds, speed and availability to the mass audiences.

What makes a web application framework useful is the fact that it provides a coding library that makes for faster by giving the basic patterns for building up a reliable and maintainable applications. Web frameworks also provide functionalities that aids in everyday operations required to run web applications. These operations include functions like URL routing and web security against Cross-site request forgery.

2.2 Chatbot Web Framework

For this project, a full stack web application will be built as a framework that is capable of tying up interfaces of other Chatbot frameworks and API services. The decided stack to develop this framework would be an application consisting of four technologies; Express JS, Node JS, React JS, Flask. The combination of React JS, Express JS and Node JS is an increasing popular and powerful technological stack to work with to build and deploy a reliable web application.

2.2.1 React JS

As the acting Front-end component, React JS [3] is the anchor of the stack and a defining component of the overall application. React JS is an open source JavaScript library maintained by Facebook that is primarily used for creating views that is rendered in HTML. React is unique for its virtual Document Object Model(DOM), as compared to typical server rendered view which uses the standard DOM manipulation. The DOM is essentially the heart of all modern interactive web application, as it is being able to constantly respond to user interface changes.

The DOM is represented by a tree data structure. Because of such Data structure, DOM updates can be very fast. However, after the change the updated element and the element of its children has to be re-rendered to update the user interface, this makes the DOM manipulation performance hindering. Furthermore, given the interface requires heavy component usage, DOM updates become more computationally intensive.

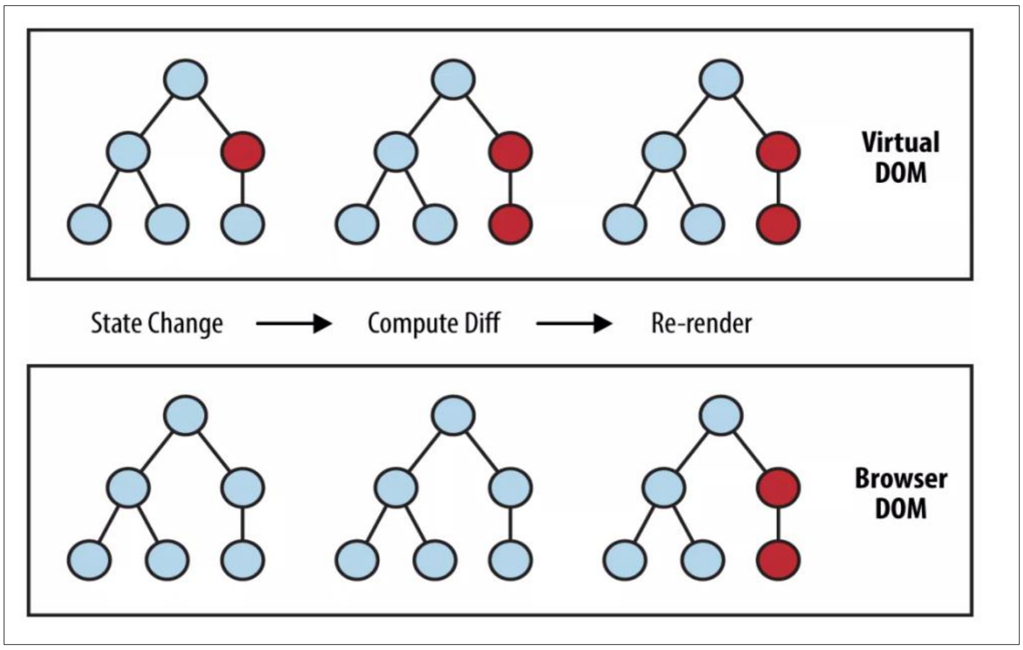


Figure 1 Virtual DOM Tree Update [4]

To solve the issue of inefficient reloading in the DOM, React JS makes use of a virtual DOM. A virtual Dom is a virtual representation of the DOM, and whenever a React application changes its state, the virtual DOM gets updated instead of the actual DOM. Although the virtual DOM is still a representation of the DOM, it exceeds the performance of the DOM in terms of speed and efficiency.

Figure 1. describes the process of a state change in the virtual DOM. Like the DOM, a virtual DOM is also represented by a tree data structure. Every node on the tree would represent an element in the DOM and if any of the element changes its state, a new virtual DOM tree will be created and compared with the previous copy of the virtual DOM. The node difference that is computed will be reflected to the real DOM and only the affected nodes will be updated instead of reloading the entire DOM. React follows a batch updating mechanism so that updates to DOM are triggered in batches instead of a per state change basis which makes React able to stand out as a high performing JavaScript library.

React fits the use case of the project because of its use of components, components are the fundamental building block of the React framework and each of them maintains and renders their own states. Each component is responsible for storing the state of its view and therefore renders independently of one another. Components can easily communicate with each other by sharing state information from parent components. This hence makes writing the application much simpler and modular.

2.2.2 Node JS

React JS is useful to the development up the front-end application, however since the application requires the integration of multiple Chatbot solutions, contact to these Chatbot requires a light weight platform to handle numerous concurrent client side requests. Data Synchronization between client and server should happen very quickly to accommodate a mass real time chat system.

This is where Node JS [5] comes in, Node JS is a platform built on a JavaScript runtime, it is an open source framework that offers an event driven architecture. Node JS has an asynchronous, event driven and non-blocking input/output model. Other language that depends on threading to perform simultaneous task where I/O events such as file system access or network request events are triggered while other code execution are blocked to await the event completion. Node JS or generally JavaScript is a single threaded engine running off a single processor.

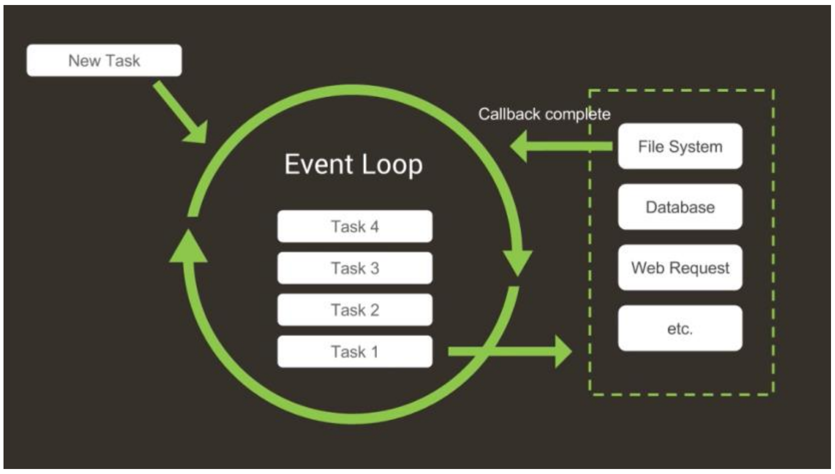


Figure 2 JavaScript Event Loop [6]

Node JS achieves a multi-threaded task handling with the help of event loops, an event loop is a set of tasks that needs to be processed and are triggered when an I/O event is completed through Callbacks. For example, if a code is written to make an API request, a Callback function is supplied to receive and manipulate the results on a successful connection. In the stated example, the API request will be an I/O event that is stored in an event queue and executed when the thread is free, and while the event is executing other tasks are allowed to run until the event is completed. Event loops essentially makes up the asynchronous properties of the engine which makes Node JS application fast and capable of handling multiple client requests, with event loops there is no need for threads which enables minimal memory usage [7].

2.2.3 Express JS

Node JS provides a lightweight JavaScript run-time environment. However, it does not serve the purpose of handling web request from the client application. Since the objective of the project is to create a client server framework, a web server must be considered to provide REST API middleware service to contact the various Chatbot solutions.

To achieve the above-mentioned task, the Express framework [8] is used to help simplify the task of writing server codes. The express framework allows for the definition of routes and it provides specifications of what happens when a HTTP request matches a pattern arrives. Express matching specifications are based on regular expressions which makes route determination very flexible.

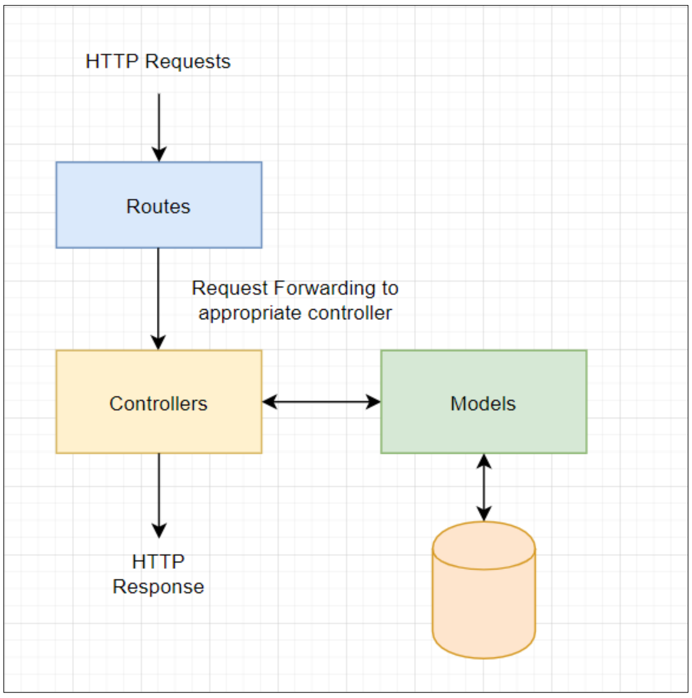


Figure 3 Express Routing Diagram [9]

Figure 3. describes the main data flow that will be requires when handling a HTTP request and response. With Express JS handling the main routing, controllers will be needed to separate the route requests from the codes that processes the requests. In this case controllers can represent the individual Chatbot APIs and would only be triggered when the route requests are received.

2.2.4 Flask vs Django

With React JS handling the user interface and Node JS handling the controller logic, both frontend and backend service are enough in laying out the overall infrastructure of the Chatbot framework. The only feature that could not be easily implemented on React and Node is the response comparison feature.

Response comparison would require Natural Language Processing tools, python is a good language consideration for the development. And to be able to serve python application as API REST service, two frameworks Django and Flask web frameworks comes into consideration.

Flask is a Python micro framework is built with a small core and an extensible philosophy in mind. Flask provides flexibility and fined-grained control and is also easy to get started as there a boilerplate code available for getting a simple app up and running [10].

Django on the other hand follows a more "Battery included" idea, which provides a user with an all in one service, this includes features like an administrative panel and a database interface [11].

Overall, Django presents a heavier framework compared to Flask, a steeper learning curve and more tuned to developing complete web application prototypes. Flask on the other hand is more focused in offering simplicity and minimalism, it allows developers to build what they want with external libraries giving it a more extensible structure.

For this project, feature comparison would only be functioning as micro service to receive and return response scores rather than a full-scale web application which has been implemented with React and Node. Therefore, a more light-weight and customizable option like Flask would be preferred for more specific solutions.

2.3 Chatbot Platforms (Dialogflow and Rasa)

Through the implementation of the Chatbot framework, there are integrations of Chatbot services such as Ask Jamie a government develop FAQ Chatbot service and a student produced QA matching model. However, it is not enough to only compare only two possible QA service, rather given the many Chatbot platforms readily available in the market, there is a need to investigate the possible integration of public Chatbot services into the Chatbot framework with hopes to analyse and understand the limitations or benefits of using such platforms. This section will discuss two possible Chatbot platform considerations that could possibly be a good contender for building a FAQ Chatbot.

The two Chatbot platforms that came out on the top of the list are Dialogflow and Rasa. Both platforms are notably the better platforms for building Chatbots in the service industry. Therefore, this section of the report will discuss features and functionality of each of the platforms to understand the final platform integration decision.

Starting off with Dialogflow Platform [12]., Dialogflow is a build-once deploy everywhere development suite for the development of website conversational interfaces and messaging platforms. With Dialogflow, installation is not necessary as Dialog provides GUI interface for the building of Chatbots hence its intuitiveness and highly customizable web interface makes it very easy to on board new users and it also makes Chatbot building relatively easier. Furthermore, Dialogflow also provides Chatbot integration to popular messaging services as well as voice assistant systems. The out of the box integration service makes Dialogflow a more versatile platform that can serve a wider demand.

Unlike Dialogflow’s closed system, Rasa [13] is an open source conversational AI platform specializing in the building of contextual assistant, enabling more code level customizations and freedom to optimize and tailor the framework to a user’s needs. Rasa however does not come with a web interface and its usage would hence require the installation of multiple software component and technical knowledge in python and JSON to be able to unlock the full capabilities of Rasa’s framework.

Dialogflow	Rasa
Installation not required as web interface is provided for building of Chatbot	Installation required for basic usage of framework, intent creation and training
Intuitive and interactive interface for easy customization	Requires knowledge in Python and JSON objects
Closed system, client API is provided for server-side usage of service	Open source system that is available on GitHub, Made for simpler integration for personal projects
Data hosted on Google Cloud Platform	Self-hosted platform

Table 1 Dialogflow and Rasa Comparison [14]

Observing the differences in both platforms, it is a very difficult decide the appropriate platform to use for the Chatbot platform as both platforms presents its own unique attributes and benefits for developers of all skill levels. Dialogflow provides a dashboard for intent training and testing, features like entities and web hooks can be configured with button clicks, these perks put Dialogflow ahead for faster Chatbot building and easier on boarding of new users. A downside of Dialogflow is its closed system, and reliance of Google Cloud Services for external interactions makes it a tedious process to use the Dialogflow API client.

On the other hand, the Rasa platform is extremely attractive to a developer due to its open sourced option. The platform opens a lot of customization opportunities to more technical creators. Additionally, being an open source system, data training can be done solely on python rather than subjecting to API post limitations like Dialogflow. The downside of Rasa would be its steeper learning curve and code familiarity to build a Chabot with confidence.

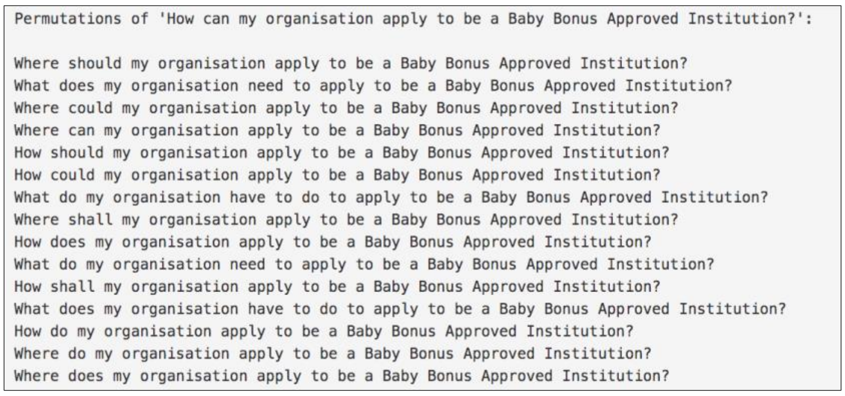
However, due to the short implementation duration and other framework building requirements, Dialogflow would be a more preferred option due to its simplicity and UI interface and automated training. Furthermore, FAQ Chatbots are more generalized and requires not much context customization which makes Dialogflow a more fitting choice.

2.4 QA Model Experiments

The QA experiments section will go through intent training on two Chatbot platforms, Dialogflow and a self-implemented Feed Forward Neural Network Model. Both training modes will require the manipulation of huge data sets. Therefore, data pre-processing is necessary to transform this raw information into mutable data objects. The following section will discuss the data pre-process steps to understand the data format that will be used for QA training.

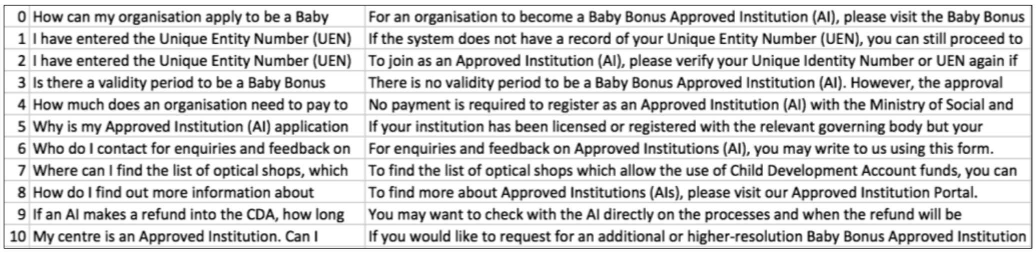
2.4.1 Data Pre-processing

For this project, two sets of data were provided, the first set of data consists of a group of text files containing total 292 Baby bonus questions and their respective permutations, each question would consist of as little as 6 to as much as 300,000 permutations. The raw data was generated by a final year student Yap Boon Peng. The second set of data is a CSV file that consist of a model question and answer pairs for Baby Bonus, this data is provided by PHD student Andrew Koh. Refer to Figure 4. for the data sample of question permutation and Figure 5. for the pair of question and answer.



```
Permutations of 'How can my organisation apply to be a Baby Bonus Approved Institution?':

Where should my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation need to apply to be a Baby Bonus Approved Institution?
Where could my organisation apply to be a Baby Bonus Approved Institution?
Where can my organisation apply to be a Baby Bonus Approved Institution?
How should my organisation apply to be a Baby Bonus Approved Institution?
How could my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation have to do to apply to be a Baby Bonus Approved Institution?
Where shall my organisation apply to be a Baby Bonus Approved Institution?
How does my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation need to apply to be a Baby Bonus Approved Institution?
How shall my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation have to do to apply to be a Baby Bonus Approved Institution?
How do my organisation apply to be a Baby Bonus Approved Institution?
Where do my organisation apply to be a Baby Bonus Approved Institution?
Where does my organisation apply to be a Baby Bonus Approved Institution?
```

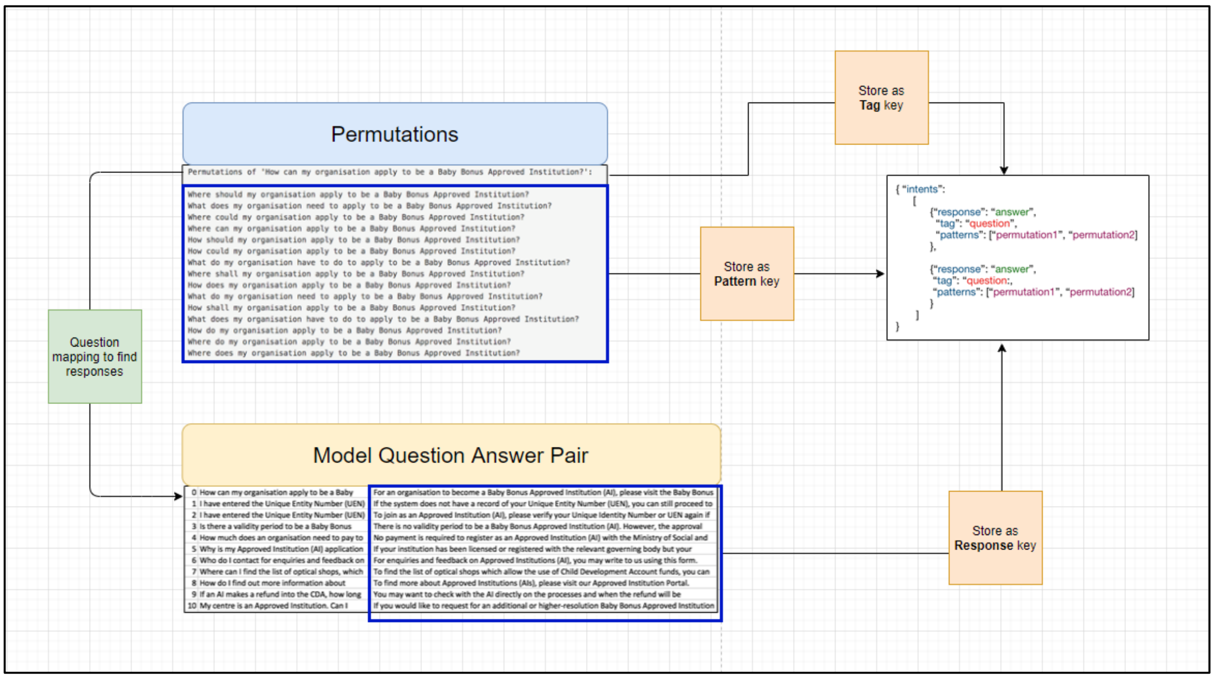
Figure 4 List of FAQ Permutations



0	How can my organisation apply to be a Baby	For an organisation to become a Baby Bonus Approved Institution (AI), please visit the Baby Bonus
1	I have entered the Unique Entity Number (UEN)	If the system does not have a record of your Unique Entity Number (UEN), you can still proceed to
2	I have entered the Unique Entity Number (UEN)	To join as an Approved Institution (AI), please verify your Unique Identity Number or UEN again if
3	Is there a validity period to be a Baby Bonus	There is no validity period to be a Baby Bonus Approved Institution (AI). However, the approval
4	How much does an organisation need to pay to	No payment is required to register as an Approved Institution (AI) with the Ministry of Social and
5	Why is my Approved Institution (AI) application	If your institution has been licensed or registered with the relevant governing body but your
6	Who do I contact for enquiries and feedback on	For enquiries and feedback on Approved Institutions (AI), you may write to us using this form.
7	Where can I find the list of optical shops, which	To find the list of optical shops which allow the use of Child Development Account funds, you can
8	How do I find out more information about	To find more about Approved Institutions (AIs), please visit our Approved Institution Portal.
9	If an AI makes a refund into the CDA, how long	You may want to check with the AI directly on the processes and when the refund will be
10	My centre is an Approved Institution. Can I	If you would like to request for an additional or higher-resolution Baby Bonus Approved Institution

Figure 5 FAQ Model Question Answer Pair

With the raw data files provided, the objective is to be able combine both information into a data object for easy parsing for any QA models. The Figure 6. shows a high-level logical data processing diagram to provide an overview of how each raw data set can be manipulated to form the desired object.



```
Permutations of 'How can my organisation apply to be a Baby Bonus Approved Institution?':

Where should my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation need to apply to be a Baby Bonus Approved Institution?
Where could my organisation apply to be a Baby Bonus Approved Institution?
Where can my organisation apply to be a Baby Bonus Approved Institution?
How should my organisation apply to be a Baby Bonus Approved Institution?
How could my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation have to do to apply to be a Baby Bonus Approved Institution?
Where shall my organisation apply to be a Baby Bonus Approved Institution?
How does my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation need to apply to be a Baby Bonus Approved Institution?
How shall my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation have to do to apply to be a Baby Bonus Approved Institution?
How do my organisation apply to be a Baby Bonus Approved Institution?
Where do my organisation apply to be a Baby Bonus Approved Institution?
Where does my organisation apply to be a Baby Bonus Approved Institution?
```

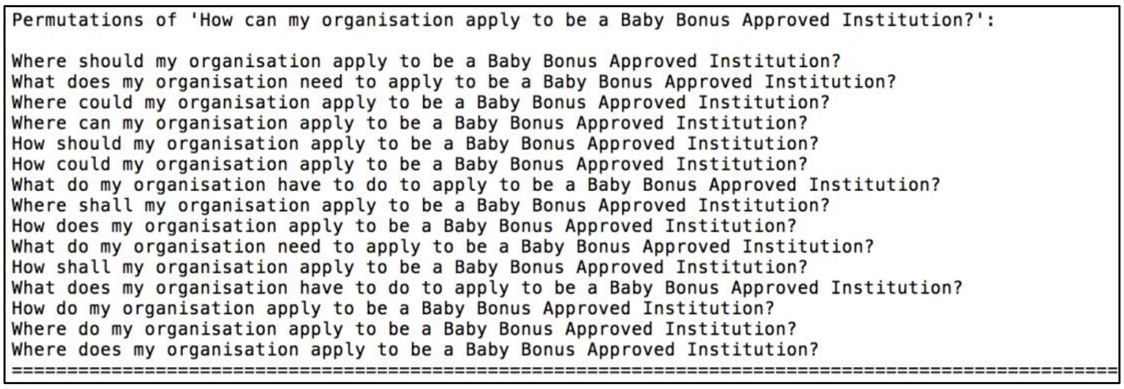
0	How can my organisation apply to be a Baby	For an organisation to become a Baby Bonus Approved Institution (AI), please visit the Baby Bonus
1	I have entered the Unique Entity Number (UEN)	If the system does not have a record of your Unique Entity Number (UEN), you can still proceed to
2	I have entered the Unique Entity Number (UEN)	To join as an Approved Institution (AI), please verify your Unique Identity Number or UEN again if
3	Is there a validity period to be a Baby Bonus	There is no validity period to be a Baby Bonus Approved Institution (AI). However, the approval
4	How much does an organisation need to pay to	No payment is required to register as an Approved Institution (AI) with the Ministry of Social and
5	Why is my Approved Institution (AI) application	If your institution has been licensed or registered with the relevant governing body but your
6	Who do I contact for enquiries and feedback on	For enquiries and feedback on Approved Institutions (AI), you may write to us using this form.
7	Where can I find the list of optical shops, which	To find the list of optical shops which allow the use of Child Development Account funds, you can
8	How do I find out more information about	To find more about Approved Institutions (AIs), please visit our Approved Institution Portal.
9	If an AI makes a refund into the CDA, how long	You may want to check with the AI directly on the processes and when the refund will be
10	My centre is an Approved Institution. Can I	If you would like to request for an additional or higher-resolution Baby Bonus Approved Institution

```
{ “intents”:
    [
        {“response”: “answer”,
         “tag”: “question”,
         “patterns”: [“permutation1”, “permutation2]
        },

        {“response”: “answer”,
        “tag”: “question:,
        “patterns”: [“permutation1”, “permutation2]
        }
    ]
}
```

Figure 6 Activity Flow of Object Construction

The main JSON object is encapsulated by the key intents, the intents key will include an array of objects and each object is denoted by three more keys namely the response, tag and patterns. Response key will map to a single string variable storing the model answer of the FAQ, the Tag key will store the model question of the FAQ and finally the Patterns key will store an array of permutation strings associated with the tag.

The constructed data object will be useful in the training implementation for Dialogflow, Deep Neural network model and future QA models and open source platforms. The next section will discuss in more technical depth of raw data pre-processing and object formation.



```
Permutations of 'How can my organisation apply to be a Baby Bonus Approved Institution?':

Where should my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation need to apply to be a Baby Bonus Approved Institution?
Where could my organisation apply to be a Baby Bonus Approved Institution?
Where can my organisation apply to be a Baby Bonus Approved Institution?
How should my organisation apply to be a Baby Bonus Approved Institution?
How could my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation have to do to apply to be a Baby Bonus Approved Institution?
Where shall my organisation apply to be a Baby Bonus Approved Institution?
How does my organisation apply to be a Baby Bonus Approved Institution?
What do my organisation need to apply to be a Baby Bonus Approved Institution?
How shall my organisation apply to be a Baby Bonus Approved Institution?
What does my organisation have to do to apply to be a Baby Bonus Approved Institution?
How do my organisation apply to be a Baby Bonus Approved Institution?
Where do my organisation apply to be a Baby Bonus Approved Institution?
Where does my organisation apply to be a Baby Bonus Approved Institution?
====================================================================================================
```

Figure 7 Permutation Formatting

Figure 7. shows the formatting of the permutations file. Every permutation contains the header “Permutations of…” and ends with line break of “===”. Since the permutation varies in sizes, manual allocation of permutations for training is tedious and inefficient.

For data cleaning on the permutation file, a python program will be used to restructure the permutations to fit the JSON structure as shown in Figure 6. The first step would be to form the Tag and Pattern keys. In the construction operation, the file is read to detect lines starting with Permutations (Header) and lines ending with “===” which denotes the end of permutation for the specific question. When the header is detected, the tag (model question) will be stored as a key in a dictionary.

If the ending line has not been reached, this would indicate that the permutations has been reached and will be iterated. Every permutation before the end line will be appended to the key tag. e.g. `{tag1: [permutation1, permutation2]}`.

```
Prepare JSON intent body: {intents:[]}

Initialize Dictionary Q
Initialize Permutation = False

Iterate files
    For line in file
        Iterate lines that are not an empty space
            If line starts with “Permutations”
                Extract text after “:”
                Set tag in dictionary: Q[text] = []
                Permutation set to False
            Else if “=” in line
                Permutation set to False
            Else
                Permutation set to True

            If Permutation is True
            //Append permutation into Dictionary
            Q[text].append(permutation)
```

The above code will traverse a file and construct a dictionary of tags and associated permutations of the tag in the following format.

`{tag: [permutation1, permutation2, …]}`

With a dictionary established, it is easy to define a random sample size of the permutation for training. And with that, the dictionary will be rebuilt into a JSON object of keys tag, patterns with a user defined sample size. E.g. {“tag”: tag1, “patterns”: [permuation1, permutation2]}. With the JSON object partially complete, the final part of the object construction will require the reading of the CSV file.

```
Open Model Question & Answer CSV file

Iterate row in CSV

    Row[0] represent id
    Row[1] represent question
    Row[2] represent response

    Iterate element in JSON Object constructed earlier
        If element[“tag”] = row[1]
            Update JSON object with(“response”: row[2])
```

The above pseudo code above would be able to construct the desired structure that will be used for training in the next few sections. Run time can be improved in the future to cater for larger data sets.

2.4.2 Training Dialogflow

The Dialogflow Platform is the chosen Chatbot Platform to be integrated into the Framework. As stated in the Chatbot Framework comparison section, Dialogflow is a closed system and the only entry point to Dialogflow is through the Client API provided to establish connection to the cloud endpoint. Since Dialogflow service utilizes Google Cloud resources, quotas and limits are applied to API calls for the purpose of throttling of request per client and effectively protecting misuse of resource usage especially for users under the free usage tier.

Before pushing intents into Dialogflow, the following quotas must be adhered to before pushing of intentions can be done.

Standard Edition	
Design Time Request: Calls to build or update an agent	60 Request allowed per minute
Maximum number of intentions	2000
Maximum number of training phrases per intention	2000
Maximum number of training phrases per agent	100,000

Table 2 Dialogflow Quota and Limits [15]

Given an approximate 300 intents of Baby Bonus with a 60 request/min quota, assuming the time for a single request completion takes 1 second. 60 Requests can be made in at least a minute. Hence to provide more leeway to accommodate faster request, the training of Dialogflow will instead perform a 40 request per minute intervals to ensure successful uploads. Therefore, Dialogflow will take place at 7 batch upload intervals.

Another scenario would be that all 40 request gets uploaded too quickly allowing the next batch to be process within the remaining 1-minute restriction as shown in Figure 8.

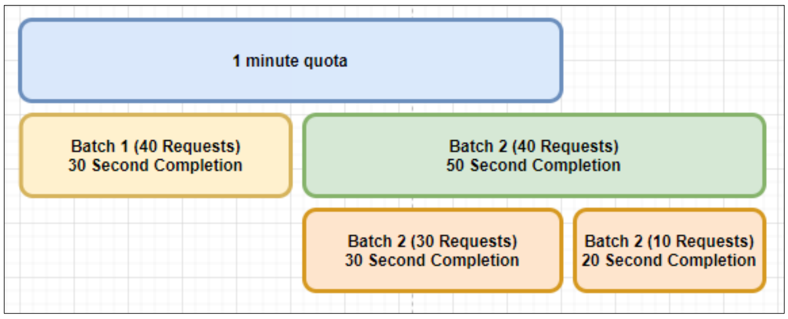

1 minute quota
Batch 1 (40 Requests) 30 Second Completion
Batch 2 (40 Requests) 50 Second Completion
Batch 2 (30 Requests) 30 Second Completion
Batch 2 (10 Requests) 20 Second Completion

Figure 8 Intent Upload Conflict

To solve the problem that can potentially cause request timeouts and partial request insertion, an alternative solution is to provide an additional delay time of 40 seconds between upload batches.

Using the constructed training data object obtained from the pre-processing steps, the following code shows the pseudo code for Dialogflow training process. From the upload conflict problem in Figure 8, the code resolves this issue by dividing the intents into batches of 40 any excess intentions would be uploaded independently. For every batch uploaded a 10 seconds delay is applied to ensure that all intents and training phrases are properly loaded.

```
//Parse data file
//Determine number of intents

If number of intents % 40 == 0
      Batch = (number of intents / 40)
Else
      Batch = (number of intents / 40) + (number of intents % 40)

Iterate Batch
      Iterate 40 request in Data Object
            Upload Intent(Data.tag, Data.pattern, Data.response)

      Sleep(40s)
```

The upload intent method will call the Dialogflow Client API request wrapper, the intent creation API takes in 3 parameters, a display name, training phrases and the intent response. Upon receiving the intent upload, Dialogflow will immediately initiate an agent training.

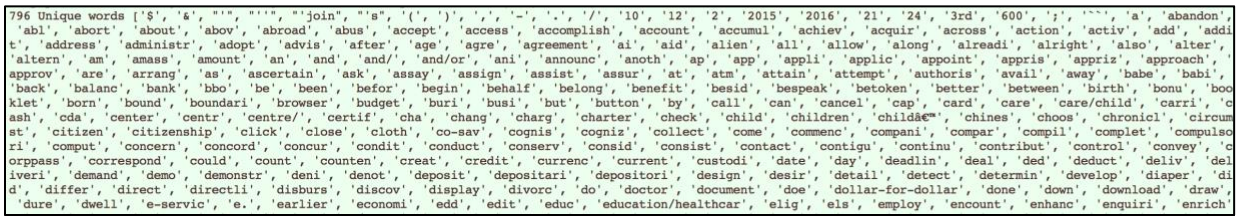
2.4.3 Training Neural Network for Text Classification

For the self-learning purpose of question answer prediction with Natural Language Processing, a text classification with deep neural network model approach [16] is researched and implemented as a proof of concept and test service for integration with the Chatbot framework.

Given a model question with a huge dataset of similar permutations of that model question, text classification was imagined to be a good technique to be applied for intent resolution. This section will go through how a Tensorflow model is built for text classification of user inputs to predict a probable intention.

The first step would involve preparation of training data, 2 arrays will be created. The first array would contain the class, classes are the target intention that the model will predicting, the second set of array will contain an array of unique stemmed words extracted from all the possible permutations of every intent that is in the data object file. The following shows a pseudo code for the data collection and an example of the array list obtained from the algorithm.

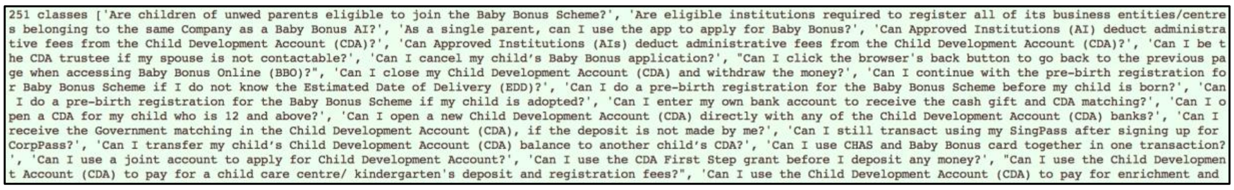
```
Load Data object file

Initialize Classes array

Initialize words array

Iterate intents
    Iterate sentence in patterns
        Word Tokenize the sentence
        Store tokenized words into words array
    Store intents into classes array

Stem words and remove duplicates in words array
```



```
796 Unique words ['$', '&', "'", "''", "'join", "'s", '(', ')', ',', '-', '.', '/', '10', '12', '2', '2015', '2016', '21', '24', '3rd', '600', ';', '``', 'a', 'abandon',
 'abl', 'abort', 'about', 'abov', 'abroad', 'abus', 'accept', 'access', 'accomplish', 'account', 'accumul', 'achiev', 'acquir', 'across', 'action', 'activ', 'add', 'addi
t', 'address', 'administr', 'adopt', 'advis', 'after', 'age', 'agre', 'agreement', 'ai', 'aid', 'alien', 'all', 'allow', 'along', 'alreadi', 'alright', 'also', 'alter',
'altern', 'am', 'amass', 'amount', 'an', 'and', 'and/', 'and/or', 'ani', 'announc', 'anoth', 'ap', 'app', 'appli', 'applic', 'appoint', 'appris', 'appriz', 'approach', '
approv', 'are', 'arrang', 'as', 'ascertain', 'ask', 'assay', 'assign', 'assist', 'assur', 'at', 'atm', 'attain', 'attempt', 'authoris', 'avail', 'away', 'babe', 'babi',
'back', 'balanc', 'bank', 'bbo', 'be', 'been', 'befor', 'begin', 'behalf', 'belong', 'benefit', 'besid', 'bespeak', 'betoken', 'better', 'between', 'birth', 'bonu', 'boo
klet', 'born', 'bound', 'boundari', 'browser', 'budget', 'buri', 'busi', 'but', 'button', 'by', 'call', 'can', 'cancel', 'cap', 'card', 'care', 'care/child', 'carri', 'c
ash', 'cda', 'center', 'centr', 'centre/', 'certif', 'cha', 'chang', 'charg', 'charter', 'check', 'child', 'children', 'childâ€™', 'chines', 'choos', 'chronicl', 'circum
st', 'citizen', 'citizenship', 'click', 'close', 'cloth', 'co-sav', 'cognis', 'cogniz', 'collect', 'come', 'commenc', 'compani', 'compar', 'compil', 'complet', 'compulso
ri', 'comput', 'concern', 'concord', 'concur', 'condit', 'conduct', 'conserv', 'consid', 'consist', 'contact', 'contigu', 'continu', 'contribut', 'control', 'convey', 'c
orppass', 'correspond', 'could', 'count', 'counten', 'creat', 'credit', 'currenc', 'current', 'custodi', 'date', 'day', 'deadlin', 'deal', 'ded', 'deduct', 'deliv', 'del
iveri', 'demand', 'demo', 'demonstr', 'deni', 'denot', 'deposit', 'depositari', 'depositori', 'design', 'desir', 'detail', 'detect', 'determin', 'develop', 'diaper', 'di
d', 'differ', 'direct', 'directli', 'disburs', 'discov', 'display', 'divorc', 'do', 'doctor', 'document', 'doe', 'dollar-for-dollar', 'done', 'down', 'download', 'draw',
 'dure', 'dwell', 'e-servic', 'e.', 'earlier', 'economi', 'edd', 'edit', 'educ', 'education/healthcar', 'elig', 'els', 'employ', 'encount', 'enhanc', 'enquiri', 'enrich'
```

Figure 9 Word bank from training phrases

Figure 9. shows the unique stemmed words after iterating every intent permutation. For this experiment stop words are being considered as some of the baby bonus intentions used in this experiment are quite similar and hence the retaining of stop words might be useful to improve classification accuracy.

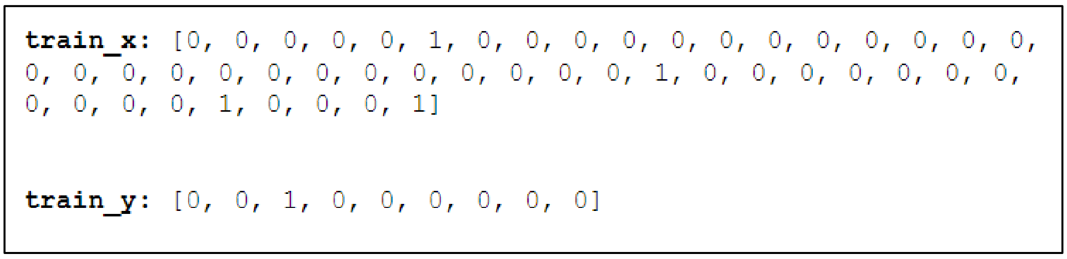


```
251 classes ['Are children of unwed parents eligible to join the Baby Bonus Scheme?', 'Are eligible institutions required to register all of its business entities/centre
s belonging to the same Company as a Baby Bonus AI?', 'As a single parent, can I use the app to apply for Baby Bonus?', 'Can Approved Institutions (AI) deduct administra
tive fees from the Child Development Account (CDA)?', 'Can Approved Institutions (AIs) deduct administrative fees from the Child Development Account (CDA)?', 'Can I be t
he CDA trustee if my spouse is not contactable?', 'Can I cancel my child's Baby Bonus application?', "Can I click the browser's back button to go back to the previous pa
ge when accessing Baby Bonus Online (BBO)?", 'Can I close my Child Development Account (CDA) and withdraw the money?', 'Can I continue with the pre-birth registration fo
r Baby Bonus Scheme if I do not know the Estimated Date of Delivery (EDD)?', 'Can I do a pre-birth registration for the Baby Bonus Scheme before my child is born?', 'Can
 I do a pre-birth registration for the Baby Bonus Scheme if my child is adopted?', 'Can I enter my own bank account to receive the cash gift and CDA matching?', 'Can I o
pen a CDA for my child who is 12 and above?', 'Can I open a new Child Development Account (CDA) directly with any of the Child Development Account (CDA) banks?', 'Can I
receive the Government matching in the Child Development Account (CDA), if the deposit is not made by me?', 'Can I still transact using my SingPass after signing up for
CorpPass?', 'Can I transfer my child's Child Development Account (CDA) balance to another child's CDA?', 'Can I use CHAS and Baby Bonus card together in one transaction?
', 'Can I use a joint account to apply for Child Development Account?', 'Can I use the CDA First Step grant before I deposit any money?', "Can I use the Child Developmen
t Account (CDA) to pay for a child care centre/ kindergarten's deposit and registration fees?", 'Can I use the Child Development Account (CDA) to pay for enrichment and
```

Figure 10 List of Model Intentions

Figure 10. shows the array of classes generated for the training experiment. This classes are unique intentions retrieved from the data object file. These classes will be the vectorised and used as the Y target data for prediction in the later section.

The next step after data preparation will be to prepare the Tensorflow input. This step will involve the transformation of training sentence inputs into patterns of binary inputs. Where 1 will represent a word from the training input matches the word position in the word array while 0 will represent a mismatch. Similarly, for the output, 1 will represent the intent element position of the class array. From the simple illustration below, the TRAIN_X input will consist of words represented by 1 in the element position of the word array, TRAIN_Y will be the matched intent with 1 being the position element of the classes array.



```
train_x: [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 1]


train_y: [0, 0, 1, 0, 0, 0, 0, 0, 0]
```

Figure 11 Input and Output Training Vector Illustration

The final step of this experiment will be to train the neural network. The neural network will be build using tflearn [17] deep learning library, a high-level API for Tensorflow and a quick way to start training the neural network. The neural network build for this experiment consist an input size equivalent to the length of the word array, every input node will consist of the 0/1 value of the TRAIN_X vector. Two hidden layers of 12 neurons, the hidden layer neuron count was derived from a research of optimality for text classification problems and given a larger input size, the neuron calculation is referenced from a rule of thumb that helps for supervised learning problems and helps in preventing over-fitting, formula is as shown in Figure 12.

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Figure 12 Hidden Neuron Rule of Thumb Obtained from Stackoverflow

Finally, an output layer of size equivalent of the TRAIN_Y, which is also the number of Baby Bonus Intentions. Regression is applied to find the best equation parameters and Deep Neural Network is defined for the model of this classification test and the classification model is constructed.

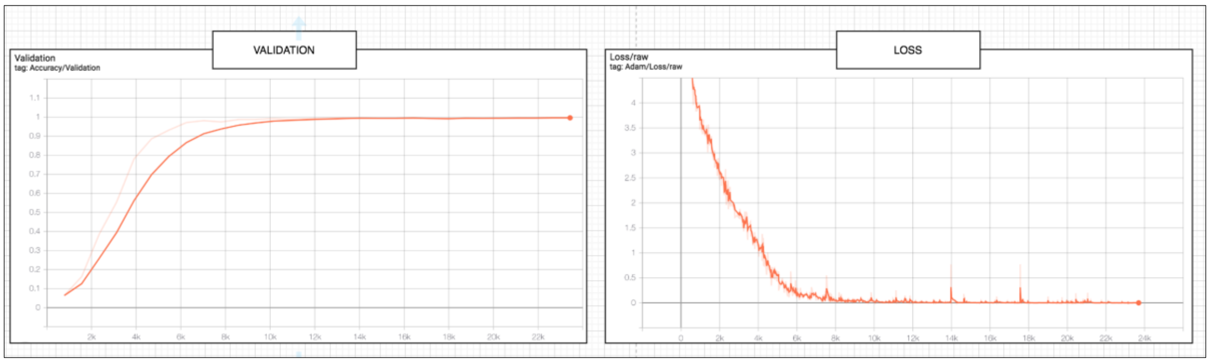


Figure 13 Deep Neural Network Training Validation and Loss

The model was fitted with an epoch of 50 and a batch size of 32 in attempts to improve the matching performance. However, accuracy already hit 80% at the 10th epoch, hence it is not wise to make any conclusions to this model as there are certainly faults in the hidden layer configuration or that the input data are too similar. Due to time constraint, this was the furthest analysis that can be explained for this experiment.

3 Full Stack Framework Implementation

3.1 Overview

Given the software research considerations described in Chapter 2, this chapter will discuss the main implementation of the Chatbot framework. The chapter will begin with an analysis of the React based frontend, Node JS backend, Python neural network and response comparison service and an overall visualization of the framework's system architecture which will include data and control flows to better understand the movement of information during a session.

The Chatbot framework consist multiple input methods, this chapter will be going through each input methods and its technical review of input state storage, query transmission and response handling. Chatbot services like ask Jamie, Dialogflow and MICL QA Engine integration methods will be discussed, the methods will include utilization of the API technology, Google cloud connections and web crawlers.

Unlike available Chatbot services like Ask Jamie and MICL's QA engine, Dialogflow and the Deep Neural network's QA engine requires training of datasets. A walkthrough data pre-processing and Chatbot training will be discussed in detail. The use of Dialogflow and Deep neural network is a proof of concept to introduce how effortless it is to integrate open source Chatbot software and self-build models into the Chatbot framework.

Finally, after going through the building block of the framework, a technical write up of response comparison service will be included to demonstrate how multiple Chatbot response comparison can be handled simultaneously and efficiently as a micro service. This chapter hopes to provide an in-depth explanation of the decisions made in building a modular and decoupled Chatbot framework to achieve application extensibility and scalability.

3.2 Frontend

The Chatbot framework will be fronted by the React JS framework for its component development architecture and usage of the virtual DOM for state management and DOM manipulations. To have a better understanding on the user interface structure of the application, Figure 14. illustrates the application skeleton and the respective components that will make up the body of the application.

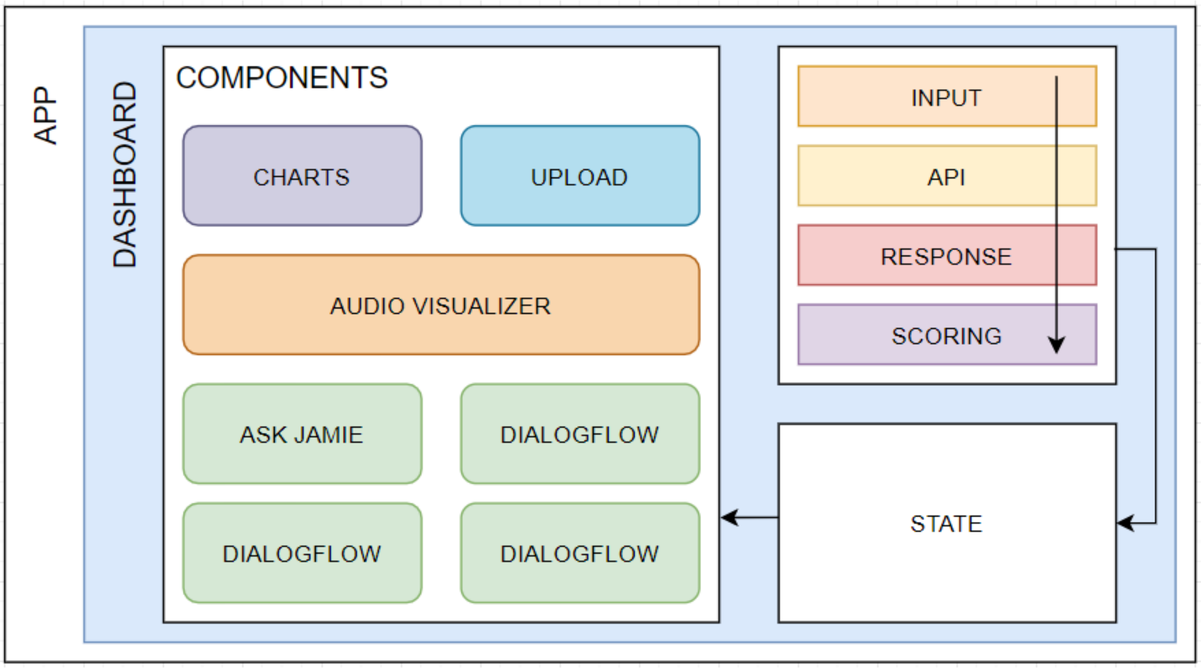


Figure 14 Frontend Component Diagram

Figure 14. shows a simple visualization of the Chatbot framework components. React components are building blocks of the user interface. Each of this component exist within the same space but executes independently from one another, each component shown in the figure have its own structure and methods. Each component will make a call to the server directly from the client-side which allows the DOM to update dynamically without refreshing the page and affecting the UI as a whole [18].

App is the overarching component of the application this is represented by a class method. As shown in the above figure, the React class method is used to create a component, each component must contain only one render method which is responsible for parsing the HTML functions in a JavaScript environment. The render method will be in charge of returning a HTML representation of the component as a DOM node.

For this application implementation, the App component will act as a container to provide extensibility for easy integration of other web pages. As shown in the figure above the Dashboard would represent the Chatbot Framework for Baby bonus, and as more Chatbot projects or frameworks are being implemented, the Chatbot framework would enable a smooth on board of applications.

The focus of this implementation segment would be a technical write up of the development design of the Dashboard page. The Dashboard page provides the main interface of the Chatbot framework and the body to various component used to features like visualization, analysis and response display. Due to the tight development time frame and various feature requirements, the Dashboard is coded relatively heavily. Although the features are implemented in its own components, they are required to share some common states for multiple display options to be possible. Some optimization has been done to decouple the code components; however, the Dashboard component still suffers from heavy usage of states and functions due to huge amounts of API requests. Suggestions for further improvements will be stated at the latter chapter.

3.2.1 Input Handling

The below figure illustrates a flowchart of the Dashboard app when performing an FAQ input with multiple Chatbot responses. The illustration will be aided by code snippets to show the logical flow of data and states management.

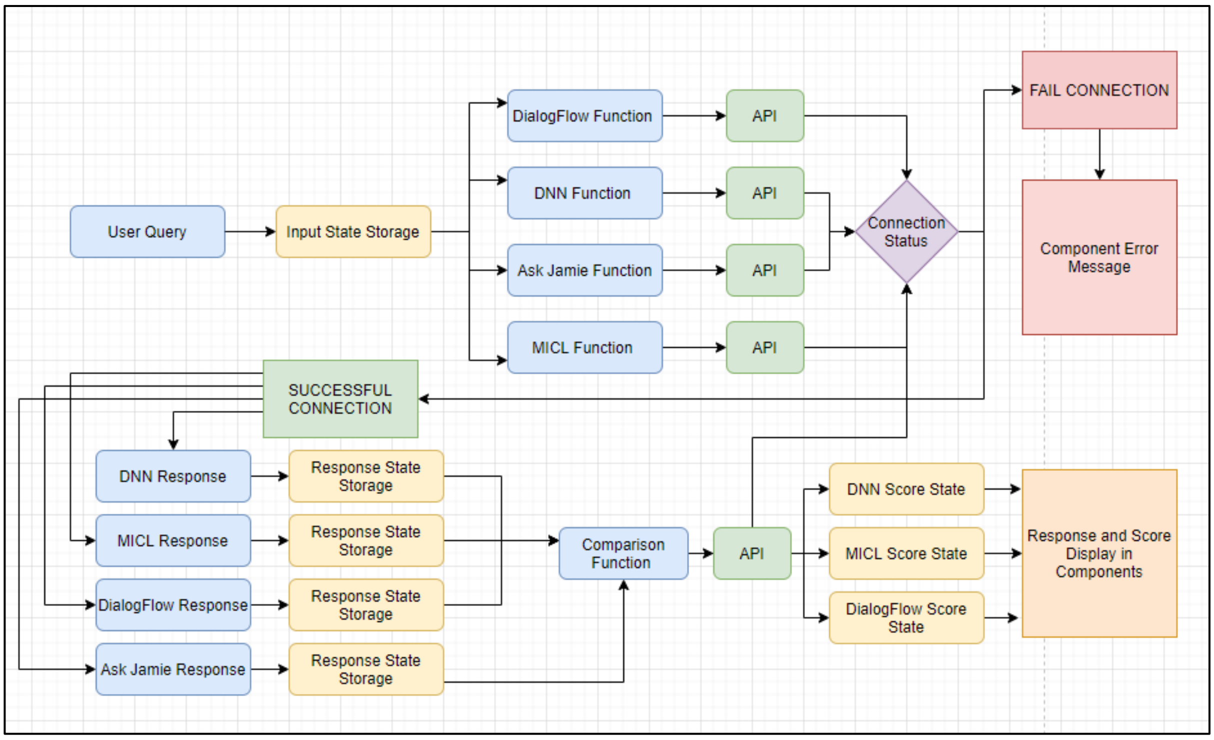


Figure 15 Chatbot Interaction Activity Diagram

The flowchart demonstrates the dataflow of a user query event, response retrieval and response comparisons. On the code level when a user inputs a query in the input field Reacts onChange handler will trigger a handle Input function setting an input state for use when querying the Chatbot services.

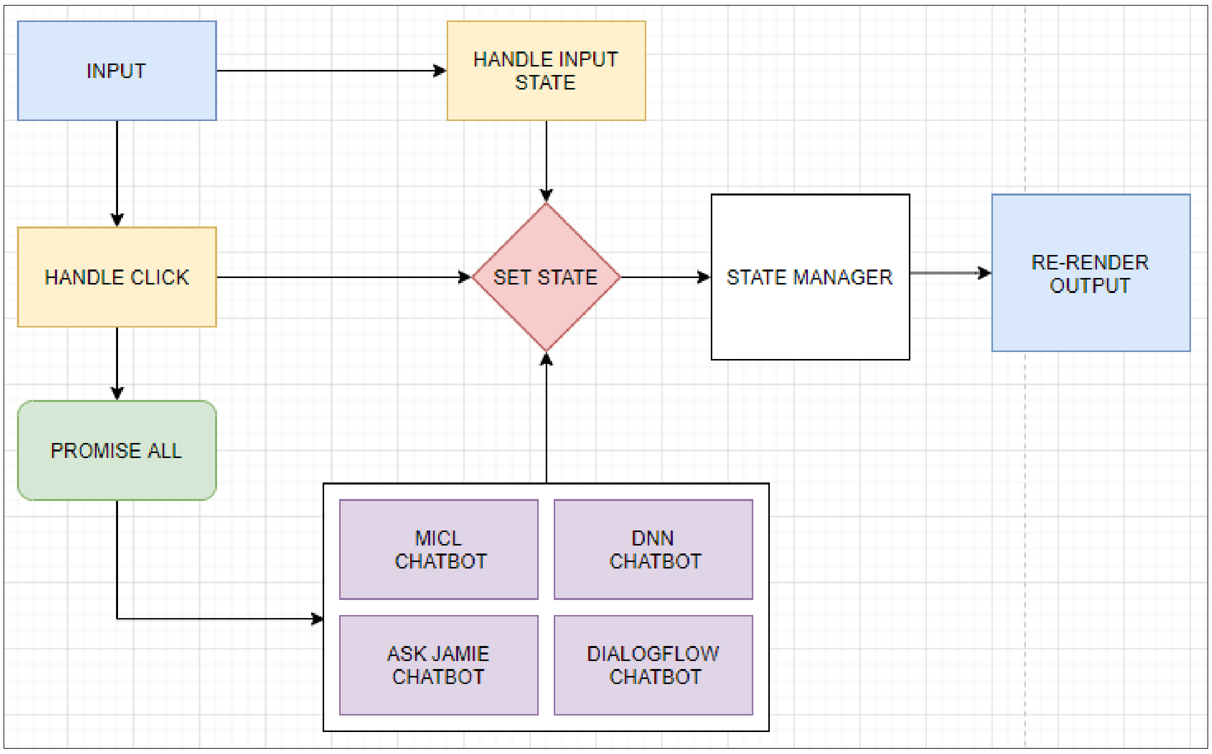


Figure 16 Multi Chatbot Interaction Activity Diagram

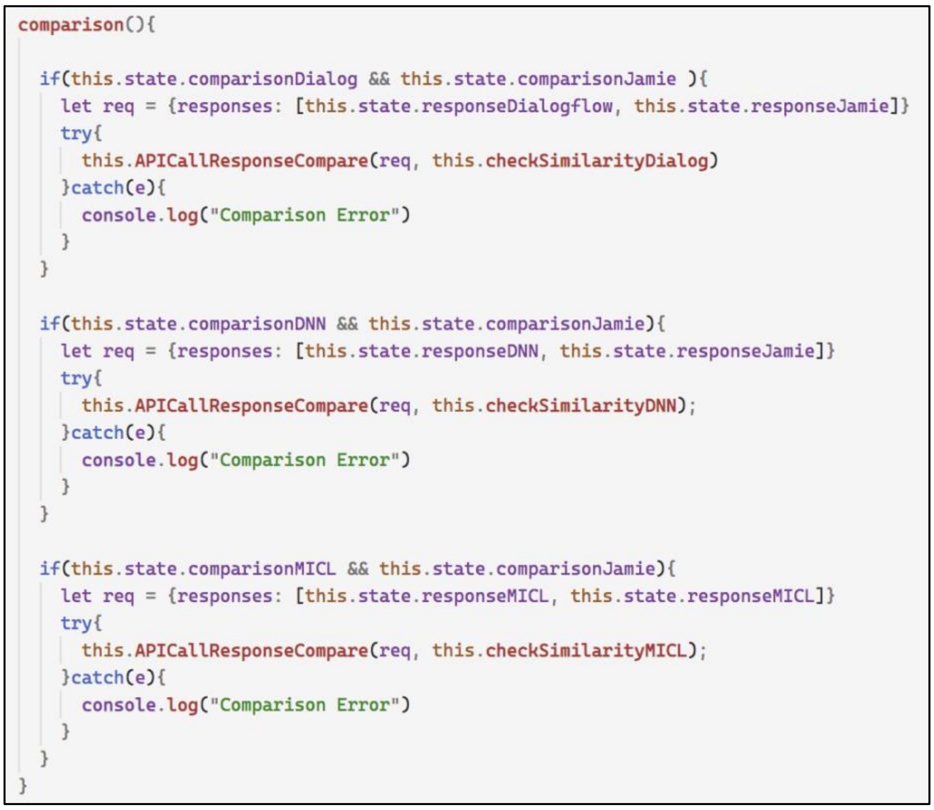
Figure 16. describes a flow of a user input request, event listeners triggered and state interaction for every function and event called. The application will begin with the input field, on every change of the input field, a set state method is called to hold a new input state. When the user request is confirmed, the submit button will trigger the on Click listener and in turn calls the handle click function to execute the query request. The handle click function will encapsulate the input in a JSON object passed into the selected Chatbot functions.

For efficient handling of request, JavaScript offers the use of Promises to ensure an asynchronous execution of code. Since all the request calls are of an API request nature, they will be added to a message queue and executed one after another. Only after the all promises are completed, the next line of instruction will be executed.

3.2.2 Multi-request Handling

To also prevent redundant API request on inactive Chatbot services, another additional feature of this framework would be the inclusion of selectable Chabot services for evaluation. This will therefore put ask Jamie Chatbot service as a required response component of the framework due to ask Jamie being a highly available Chatbot while other services are disabled at start. Disabled Chat services would not be executed by the Promise all function, and this can be achieved by setting Boolean value on an AND comparator as seen in the Promise all function in Figure 17. if a particular QA service is not active a False value in an AND operation would result in an overall False output.

For each successful Chatbot service requests, a response state would be kept, this is necessary as the ask Jamie Chatbot will be used as a bench mark for response accuracy testing with all other Chatbot systems, hence the state of ask Jamie responses will be retained along with other Chatbot responses.

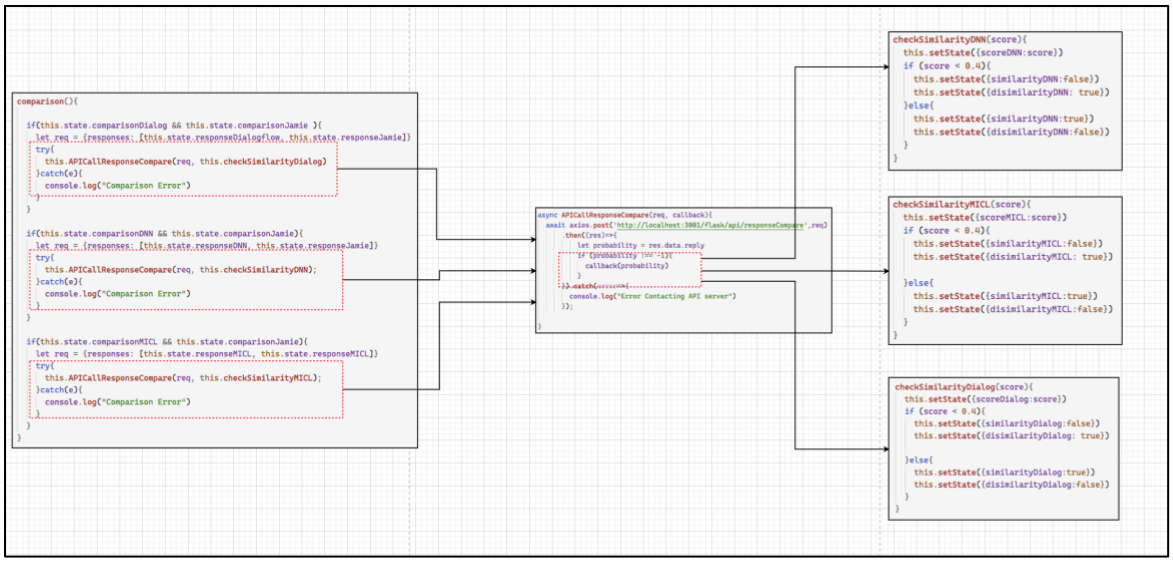


```
comparison(){

  if(this.state.comparisonDialog && this.state.comparisonJamie ){
    let req = {responses: [this.state.responseDialogflow, this.state.responseJamie]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityDialog)
    }catch(e){
      console.log("Comparison Error")
    }
  }

  if(this.state.comparisonDNN && this.state.comparisonJamie){
    let req = {responses: [this.state.responseDNN, this.state.responseJamie]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityDNN);
    }catch(e){
      console.log("Comparison Error")
    }
  }

  if(this.state.comparisonMICL && this.state.comparisonJamie){
    let req = {responses: [this.state.responseMICL, this.state.responseMICL]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityMICL);
    }catch(e){
      console.log("Comparison Error")
    }
  }
}
```

Figure 17 Comparison Conditional Statements

Only after the completion of the promises and state storage of responses of the functional Chatbot services, the comparison function will be called to prepare for response similarity comparison. Figure 17. shows a set of IF conditions that is used to ensure that response pair of Chat service and ask Jamie model are valid before executing the comparison services.

For every successful response pair, an array will be set to store the pair of responses and passed into the APICallResponseCompare function. The parameter of the function excepts the following parameters APICallResponseCompare *(JSON object of response pair, callback function to check determine similarity between response pair).*



```
comparison(){

  if(this.state.comparisonDialog && this.state.comparisonJamie ){
    let req = {responses: [this.state.responseDialogflow, this.state.responseJamie]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityDialog)
    }catch(e){
      console.log("Comparison Error")
    }
  }

  if(this.state.comparisonDNN && this.state.comparisonJamie){
    let req = {responses: [this.state.responseDNN, this.state.responseJamie]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityDNN);
    }catch(e){
      console.log("Comparison Error")
    }
  }

  if(this.state.comparisonMICL && this.state.comparisonJamie){
    let req = {responses: [this.state.responseMICL, this.state.responseMICL]}
    try{
      this.APICallResponseCompare(req, this.checkSimilarityMICL);
    }catch(e){
      console.log("Comparison Error")
    }
  }
}
```

```
async APICallResponseCompare(req, callback){
  await axios.post('http://localhost:3001/flask/api/responseCompare',req)
      .then((res)=>{
          let probability = res.data.reply
          if (probability !== -1){
            callback(probability)
          }
      }).catch(error=>{
        console.log("Error Contacting API server")
      });

}
```

```
checkSimilarityDNN(score){
  this.setState({scoreDNN:score})
  if (score < 0.4){
    this.setState({similarityDNN:false})
    this.setState({disimilarityDNN: true})
  }else{
    this.setState({similarityDNN:true})
    this.setState({disimilarityDNN:false})
  }
}
```

```
checkSimilarityMICL(score){
  this.setState({scoreMICL:score})
  if (score < 0.4){
    this.setState({similarityMICL:false})
    this.setState({disimilarityMICL: true})

  }else{
    this.setState({similarityMICL:true})
    this.setState({disimilarityMICL:false})
  }
}
```

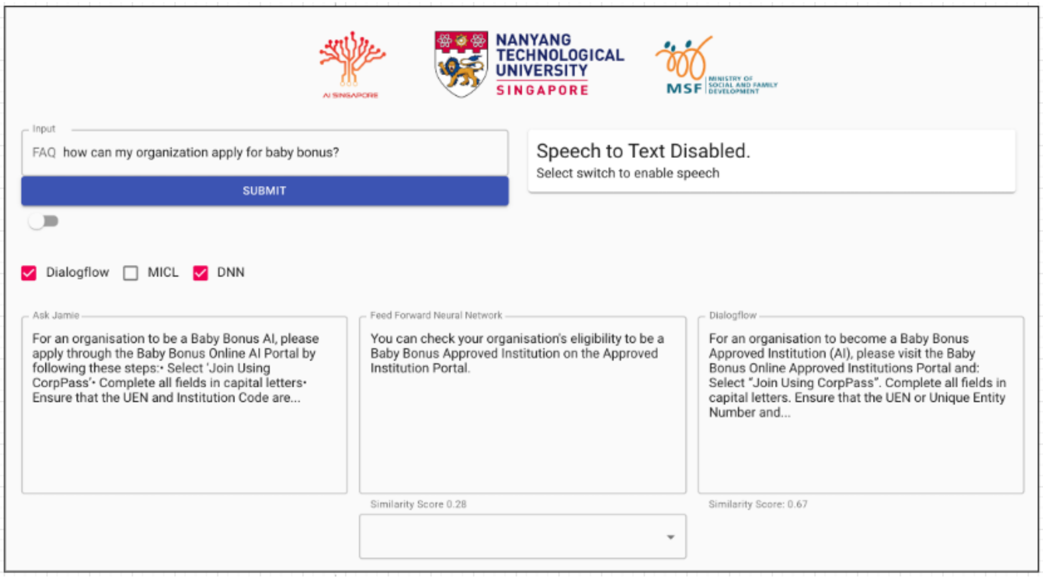
```
checkSimilarityDialog(score){
  this.setState({scoreDialog:score})
  if (score < 0.4){
    this.setState({similarityDialog:false})
    this.setState({disimilarityDialog: true})

  }else{
    this.setState({similarityDialog:true})
    this.setState({disimilarityDialog:false})
  }
}
```

Figure 18 response Comparison Flow Diagram

To reduce code complexity and redundancy, the APICallResponseCompare function takes in a callback function so that there is no need for repeating the same API request for each Chatbot response. In each check similarity function, the objective of this feature is for the user to set a similarity threshold, for this example a threshold of 0.4 is provided since some responses are of similar intent just written differently.

States are split into similarity state and dissimilarity Boolean states and are changed actively to respond to constant user input and new response comparisons. The similarity prefix with a True state will mean that the responses are similar while False would mean that responses are different, vice versa for the Dissimilarity prefix. The diagram above provides a more high-level description of the comparison procedure and score determination.

3.2.3 Chatbot Framework User Interface



FAQ how can my organization apply for baby bonus?

Speech to Text Disabled.
Select switch to enable speech

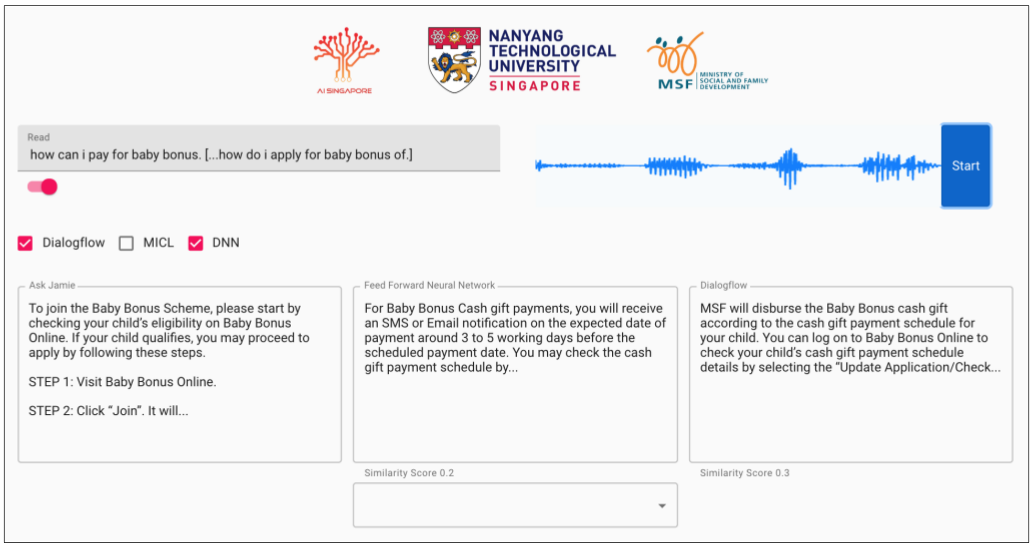
[✓ Dialogflow] MICL [✓ DNN]

Ask Jamie
For an organisation to be a Baby Bonus AI, please apply through the Baby Bonus Online AI Portal by following these steps:• Select 'Join Using CorpPass'• Complete all fields in capital letters• Ensure that the UEN and Institution Code are...

Feed Forward Neural Network
You can check your organisation's eligibility to be a Baby Bonus Approved Institution on the Approved Institution Portal.
Similarity Score 0.28

Dialogflow
For an organisation to become a Baby Bonus Approved Institution (AI), please visit the Baby Bonus Online Approved Institutions Portal and: Select "Join Using CorpPass". Complete all fields in capital letters. Ensure that the UEN or Unique Entity Number and...
Similarity Score: 0.67

Figure 19 Chatbot Framework Application Text



how can i pay for baby bonus. [...how do i apply for baby bonus of.]

[✓ Dialogflow] MICL [✓ DNN]

Ask Jamie
To join the Baby Bonus Scheme, please start by checking your child's eligibility on Baby Bonus Online. If your child qualifies, you may proceed to apply by following these steps.

STEP 1: Visit Baby Bonus Online.

STEP 2: Click "Join". It will...

Feed Forward Neural Network
For Baby Bonus Cash gift payments, you will receive an SMS or Email notification on the expected date of payment around 3 to 5 working days before the scheduled payment date. You may check the cash gift payment schedule by...
Similarity Score 0.2

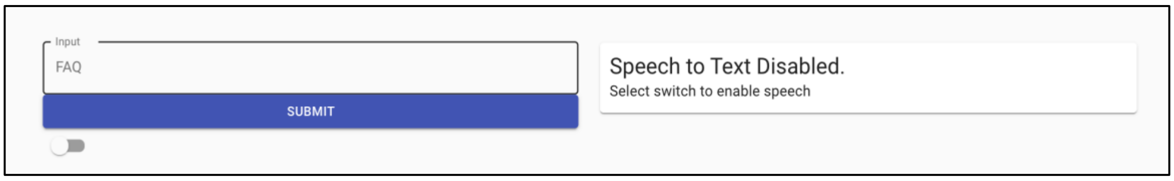
Dialogflow
MSF will disburse the Baby Bonus cash gift according to the cash gift payment schedule for your child. You can log on to Baby Bonus Online to check your child's cash gift payment schedule details by selecting the "Update Application/Check...
Similarity Score 0.3

Figure 20 Chatbot Framework Application Speech

Figure 19. and 20. shows the complete full stack implementation with the two different input modes text and speech respectively. The web application interface is built on the Dashboard JSX file which consist of various interconnected components.

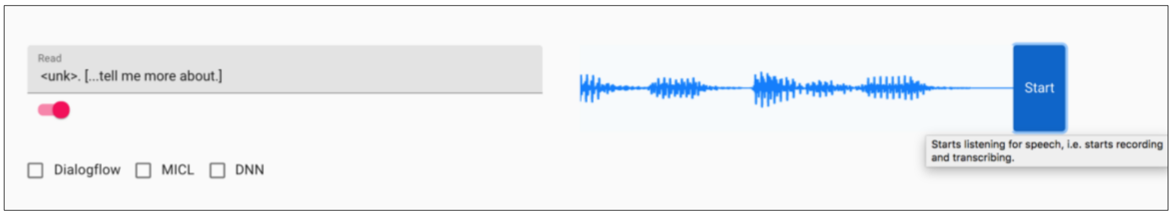
The diagrams showcase a one to one dialog between the many available Chatbot system. This frontend implementation discusses the general user input, a selectable Chatbot service and response comparison data flow and will represent the basic product for deployment. In Chapter 4, the second feature design part will cover more advanced features that value adds to the overall implementation.

3.2.4 Speech to Text Input



Speech to Text Disabled.
Select switch to enable speech

Figure 21 Text Input Method



Starts listening for speech, i.e. starts recording and transcribing.

Figure 22 Speech Input Method

Another feature added for the Chatbot framework is the use of live speech streaming input, this means the user will be able to communicate to the chatbots through voice. As shown in Figure 22. a user has a choice between two input options, text and speech input respectively. The choice of including a speech input is to add convenience and a hands-free option when making a query to virtual assistants.

For the implementation of the speech input streaming, a full duplex communication channel between a client and server must be established. This two-way interaction is enabled by the Web Socket protocol, A Web Socket responsibility is to provide persistent real-time communications between the client and server over a single TCP socket connection. The Web Socket protocol has two primary agenda, the first is to establish a handshake then enable data transfer, once the frontend and backend both have their handshakes in, they will be able to transfer data to one another at will with less overhead [19].

As soon as the TCP socket is established, the server side will make connection with a speech to text transcribing service hosted by AISG Singapore that begins to listen for input data from the client. Since data required by the transcription is in the form of audio bytes, an audio recorder module is required to initiate a browser recording session whenever data stream is enabled.

Audio data are large input object, and will be transferred to the backend through the persistent socket channel in the form of BLOB (Binary Large Objects), a blob object cannot be changed directly but it can be sliced into parts and put together again on the other end [20].

Hence, this makes streaming audio through sockets more efficient and would allow batch feeding of transcribing service during long pauses between input requests instead of waiting for full audio recordings. The AISG transcription service will hold two states a determination state and a final state. In the determination state, a short time frame is set to guess the probable audio input while the final state will deliver the most confident transcription. The react client in this case will actively read the states as its being returned from the web socket and upon the determination of the final transcription, the transcribed state will be sent to the Chatbot services. The following dialog map shows the speech to text input process and client server interaction for a better visualization of the above explanation.

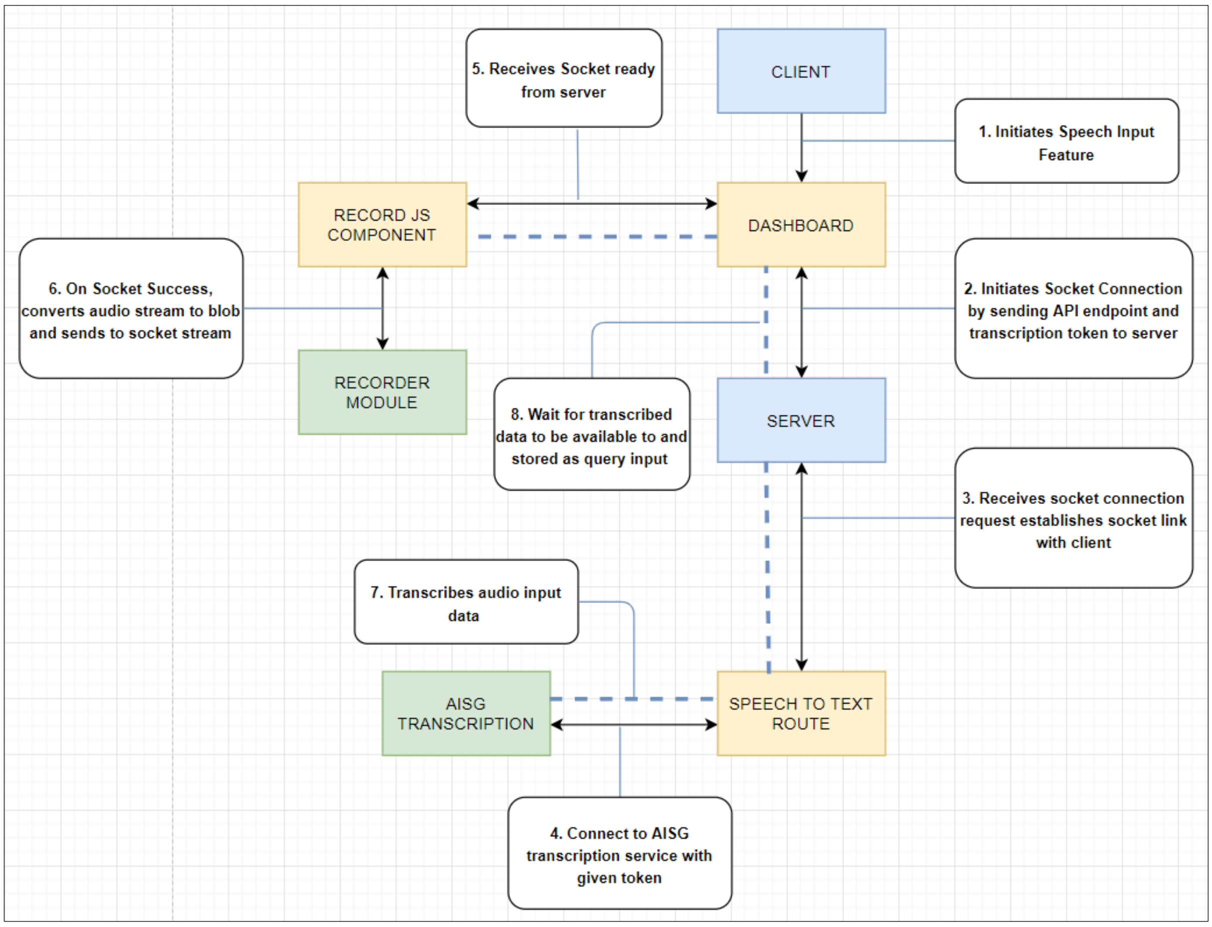

5. Receives Socket ready from server
CLIENT
1. Initiates Speech Input Feature
RECORD JS COMPONENT
DASHBOARD
6. On Socket Success, converts audio stream to blob and sends to socket stream
2. Initiates Socket Connection by sending API endpoint and transcription token to server
RECORDER MODULE
8. Wait for transcribed data to be available to and stored as query input
SERVER
3. Receives socket connection request establishes socket link with client
7. Transcribes audio input data
AISG TRANSCRIPTION
SPEECH TO TEXT ROUTE
4. Connect to AISG transcription service with given token

Figure 23 Dialog Map of Speech to Text Initialization

Web Sockets are excellent and convenient ways to achieve real-time solutions in an application. It provides flexibility to leverage on full duplex communication to provide a fast and responsive Chat service experience.

3.2.5 Audio File Input

The audio file input is an independently development feature, used mainly for Audio transcription tests. Hence, it is currently not integrated to the main application stack. This feature includes two API endpoints developed on the Node Server `/record/api/transcribe` and `/record/api/upload`.

The objective of this feature is to be able to upload and transcribe audio files to test transcription accuracy on the AISG hosted Speech to text service. Speech transcription poses risk such as packet loss or noise which can potentially affect the quality of transcribed text. Hence, the use of audio file upload is designed to analyse the performance of speech transcription service and understand what is required to further improve transcription.

The implemented audio file upload process is still in the development phase hence, the process of upload is split up into two components and has to be selected separately.

The first step will require a user to first upload an audio file which will trigger an upload listener to encapsulate the file object into a form data for transfer to the Node server backend via `/record/api/upload` API. The upload API will leverage a Node module called Formidable to parse the received form data and create an instance of audio file in the Backend directory.

On successful file upload, the `/record/api/transcribe` will be triggered to send the uploaded audio file to the transcription server. The transcribed text will be returned to the frontend for processing and analysis. The figure below shows a dialog map describing the process of the file input feature.

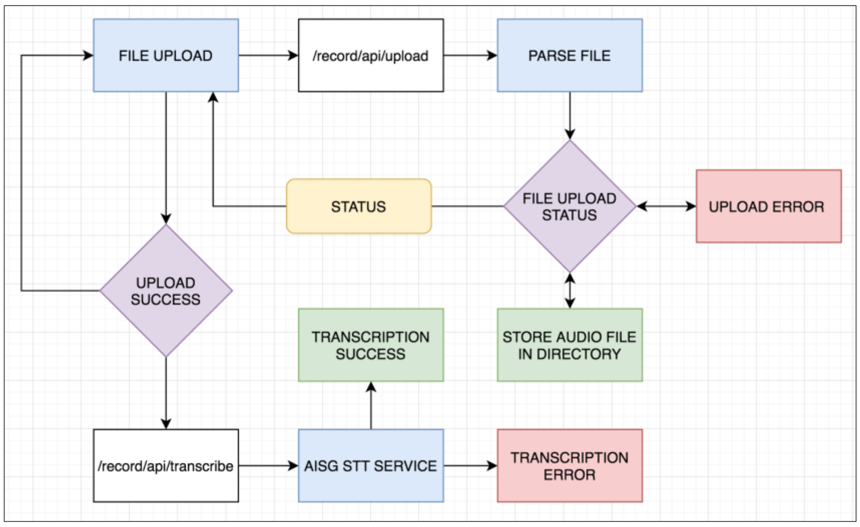


Figure 24 Dialog Map of File Upload

3.3 Node JS Backend and API services

The backend of this project will be used middleware between the Frontend software, Chatbot service and the response comparison service. As explained in Chapter 2, the Backend makes use of Express, a minimal and flexible Node JS web application framework. Express will serve as a API endpoint that provides routing table which is used to perform different actions based on HTTP methods and URL to the respective Chatbot services. This segment will discuss the architecture of backend services utilizing Node JS and Express JS.

The architectural style used for the designing of this network heavy application would be the REST architecture. REST is an abbreviation for Representational State Transfer, RESTful applications makes use of HTTP methods like GET, POST, PUT, DELETE) to perform creation, reading, updating and deletion operations. For this backend implementation, Node JS is opted as an optimal framework for building the RESTFUL API is able to support intensive I/O operations like speech streaming and handle multiple concurrent requests as mentioned in Chapter 2.

3.3.1 Node JS Architecture

Figure 25. shows a typical HTTP request/response call flows with Node and Express. For this implementation, Node will be used as a runtime server-side execution and used as a app server for React application. With Node receiving client requests, express will sit in the routing layer where the requests are processed by middleware functions before being sent out to the handlers.

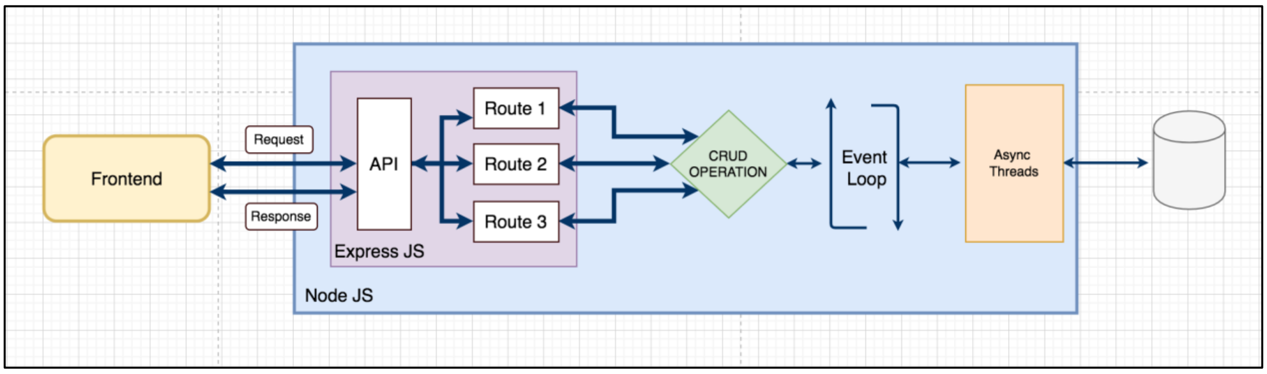


Figure 25 Backend Operation [21]

Following the above architecture, the Node JS will start by initializing the module App.js. This module first imports the Express application object. However instead of setting a Router object to initialize routes directly on the app module, a routes directory is created to segregate routing to the specific Chatbot services. This design is to function as a middleware to ensure proper route organization, additionally middleware can be useful when certain conditions needs to be applied to a particular route in the system. For example, in future implementations when more Chatbot services are integrated into the framework, there will be a need to secure certain Chatbot services with features like authorization or a log in mechanism, this would make middleware implementation very useful as it can be used to prevent unauthorized users from accessing certain routes.

3.3.2 API Routing

```
const dialogflowRouter = require('./routes/dialogflow');
const miclRouter = require('./routes/micl');
const flaskRouter = require('./routes/flask');
const askJamieRouter = require('./routes/askJamie');

const STT = require('./controllers/MainController');
const upload = require('./upload');

app.use('/dialog', dialogflowRouter);
app.use('/micl', miclRouter);
app.use('/flask', flaskRouter);
app.use('/jamie', askJamieRouter);

app.post('/stream/record', STT.streamByRecording)
app.post('/stream/import', upload.single('file'), STT.streamByImport)
```

The code above defines the content of the app.js module which consist of four Chatbot service calls and two audio streaming services. Figure 26 below demonstrates a clearer overview of the API services that the backend will provide.

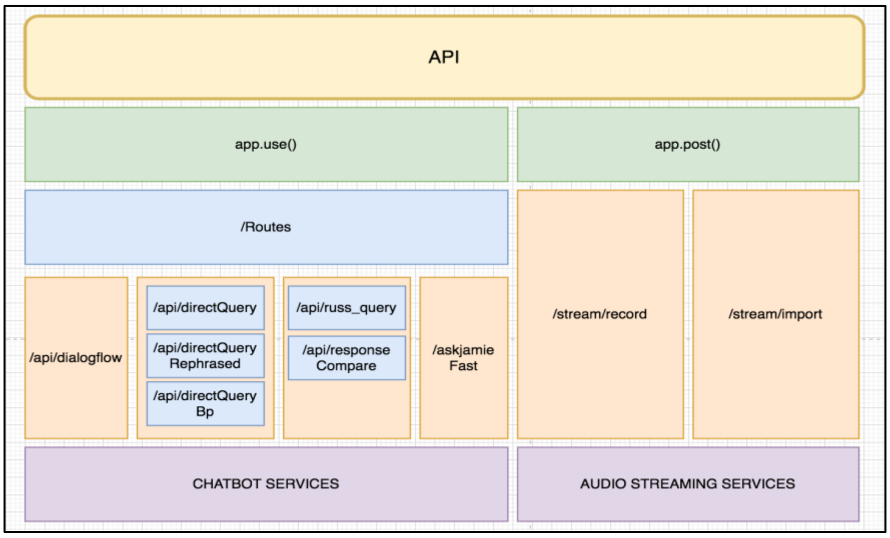


Figure 26 Backend System Architecture

3.4 Chatbot FAQ services

For the current implementation, four Chatbot services are researched and used for the testing of the framework. Each service has its own unique API request methods and are used to demonstrate the flexibility of the framework to be able to handle any types of Chatbot connection protocols. The following section will be discussing each Chatbot APIs and how each of these services handle a request and response from their respective third-party interface.

3.4.1 Ask Jamie

Ask Jamie is a production level Chatbot service developed by Government Technology, Singapore (Govtech Singapore) and will be used as a benchmark model to test for accuracy among other Chatbot services. Ask Jamie does not currently have an official API service provided for the public users, an alternative to integrate Ask Jamie services onto the Chatbot framework is to make use of a web crawler to scrap the Baby Bonus website for responses. An initial implementation to reach Ask Jamie was through the Selenium web testing framework.

The Selenium framework uses a WebDriver to make direct calls to the browser using a desired browser's native support for web automation, this allows Selenium to be able to interact with a browser and simulate user interaction events. Hence for this implementation, Selenium was used to capture the HTML code of the Baby bonus website, search for the Ask Jamie element and simulate a user query request and retrieve the reply.

```
let driver = await new
Builder().forBrowser('chrome').setChromeOptions(options).build();
try {
	await driver.get('https://www.babybonus.msf.gov.sg');

	await driver.findElement(By.name('chat_input')).sendKeys('baby bonus',
	Key.RETURN);

	await driver.wait(until.elementLocated(By.className('speech\-
	bubble1')),10000)
} finally {
	var replyPromise = driver.findElement(By.className('last_li'))
	                  .findElement(By.className('speech\-bubble1'))
	                  .findElement(By.tagName('div')).getText();

	replyPromise.then((text)=>{
	              AskJamieReply = text
	              driver.quit()
```

The above solution was successful in sending an FAQ input to ask Jamie and receiving a reply. However, the idea of making use of selenium lies in the time taken to get a response. For an input to be made, three asynchronous request has to be made to the web driver as shown below, first request is made to navigate to the Baby Bonus website. Next the driver will invoke an element search method, this element returns a Web Element for the HTML in the form of a DOM tree where the finder traverses the DOM tree to locate the desired element and a sendKey method is called to perform the input. Finally, the last driver request is set to wait for response element to be created, a 1 second delay is set to cater for more heavy response from ask Jamie.

```
await driver.get('https://www.babybonus.msf.gov.sg');
await driver.findElement(By.name('chat_input')).sendKeys('baby bonus', Key.RETURN);
await driver.wait(until.elementLocated(By.className('speech\-bubble1')),10000)
```

Although the above solution was effective for the purpose of integrating Ask Jamie with the Chatbot framework, the response fetching can be very unpredictable at times due the fixed delay waiting time. In circumstances where ask Jamie encounters load time issues that exceeds the delay limit, an empty response will be fetched instead. Furthermore, increasing the wait time would also be worsening the response time on the framework.

An alternative solution was provided by another Final Year Student which was able to use a proprietary API called:

```
https://va.ecitizen.gov.sg/flexAnsWS/ifaqservice.asmx/AskWSLanguage
```

This API solution was able to receive a user input and respond with an XML string. The script then makes use of a web scraper to extract the responses off the xml text. This solution was observed to perform much better than the Selenium solution as the use of API exceeds the run time performance of a live web scrap solutions with manual await settings.

Since the web scraper library is used in the python environment, the ask Jamie API solution was written in python. And because the chosen API middleware is on the node JS platform, a "child_process" module is used to spawn a shell processes that is able to perform command line execution. The following code was able to trigger a python file execution on a shell process, the STDOUT received on the command line process will be read into data variable and returned as a API response from Node JS.

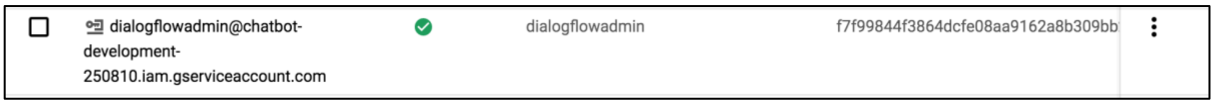
```
const { spawn } = require('child_process');
const pyProg = spawn('python3', ['ask_jamie.py','--test_questions', query]);

pyProg.stdout.on('data', function(data) {
      res.json({reply: data.toString()});
});
```

3.4.2 Dialogflow

Dialogflow is an open source natural language processing platform developed by Google. It is used for building up conversational application easily. After the training of the Dialogflow model as discussed in Chapter 2. This section will discuss its integration to the Chatbot framework.

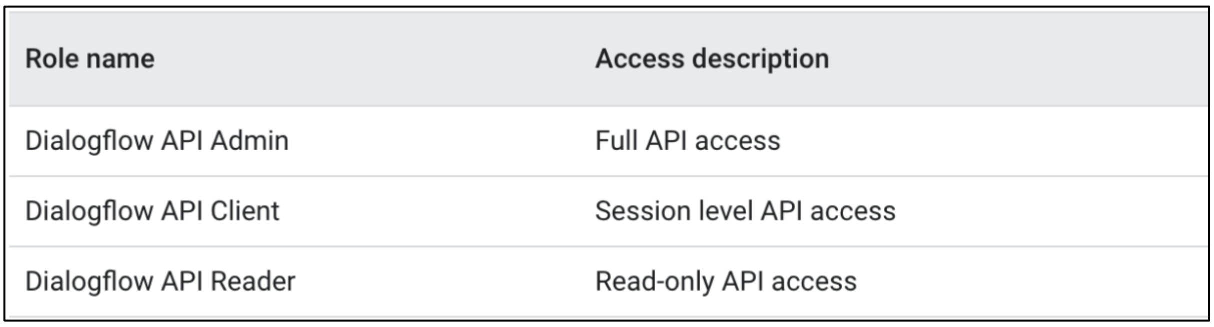
Fortunately for Node JS, Dialogflow provides a Client API option for many programming languages and frameworks, Node included. The setup of the Dialogflow API however requires a lot of pre-requisites, firstly because Dialogflow is a Google Service that runs on the Google Cloud Platform, API calls will have to go through Google Cloud before redirecting to Dialogflow services. First, the Dialogflow API must be enabled on the Google Cloud Console.

Next a Dialogflow admin service account will be created, this service account will require three essential roles that is required to perform read and write operations on Dialogflow. The Role access permission is shown in the table below.



☐	dialogflowadmin@chatbot-development-250810.iam.gserviceaccount.com	✓	dialogflowadmin	f7f99844f3864dcfe08aa9162a8b309bb

Figure 27 Dialogflow Service Account



Role name	Access description
Dialogflow API Admin	Full API access
Dialogflow API Client	Session level API access
Dialogflow API Reader	Read-only API access

Figure 28 Dialogflow Service Account Permissions [22]

After the account setup, a secret key must be generated to authenticate with the service account. After the pre-requisites are satisfied, a node JS connection code can be written to make an interaction request with Dialogflow. The following codes describes the Dialogflow API connection process.

```
const sessionId = uuid.v4();
const projectId = 'chatbot-development-250810'
const sessionClient = new dialogflow.SessionsClient();
const sessionPath = sessionClient.sessionPath(projectId, sessionId);
```

A request is prepared with attribute session key to the client session and a query input object method for Dialogflow parsing.

```
const request = {
      session: sessionPath,
      queryInput: {
        text: {
          text: query,
          languageCode: 'en-US',
        },
      },
};

   await sessionClient.detectIntent(request).then(responses=>{
      const result =
responses[0].queryResult.fulfillmentMessages[0].text.text[0];
      res.json({reply: result})
     }).catch(err=>{
      res.json({reply: "Unable to reach Dialogflow"})
     })
```

For query request and response, the session client will be using the detect Intent method for question and answer matching. The detect intent method will take in the request embedded in the JSON object for intent matching upon intent detection will return an appropriate response object.

3.4.3 Deep Neural Network

The deep neural network API is a QA model written to test the Flask API framework for QA model evaluations and also an additional self-implemented feature to try out a different QA matching technique for FAQ as elaborated in the QA Experiments section of Chapter 2. Since the DNN model is hosted by the Flask framework that offers RESTUL APIs, Node JS can simply make use of the request module to submit a HTTP request form to the FLASK API to get a response. The Flask API consist of two API functions, one will be used for as the QA model while the other will be used for response comparisons. An in-depth architecture design will be discussed in the later segments of the chapter.

3.4.4 MICL QA Matching

Similar to the Deep neural network Flask API, MICL's QA matching model is hosted on the lab's cloud services, hence only an API endpoint is provided to make the query request. The lab's QA model offers three API endpoints. Each of which presents the different approaches to intent classification and its responses.

3.5 Flask Framework

The Python Flask Framework is built to provide RESTful API services for two of features of the Chatbot framework. Flask is a simple and yet a powerful python web framework, moreover, building web RESTful services with Flask is relatively simple and fast to setup this will be used for deployment consideration in the future. The Flask application consists of three parts, the main app file that will function as an API gateway or the entry point of the application and two python class files which represents the question answering service and comparison service.

Starting with the entry point of the application, this file contains the Flask framework's most critical feature, the routing service. The API gateway for Flask API service is responsible for mapping URLs to actions, and routes are used to define the way users will access the ever-changing data. Every web framework begins with the concept of content serving, the use of routing will provide URL patterns of the feature. Each route will redirect the user request to the respective class files for computation and reconstruct the raw data received from each service files for response back to the NodeJS middleware. The following diagram provides an architecture diagram of how the Flask framework is implemented and interaction of the entry point with the service objects.

3.5.1 Flask Architecture

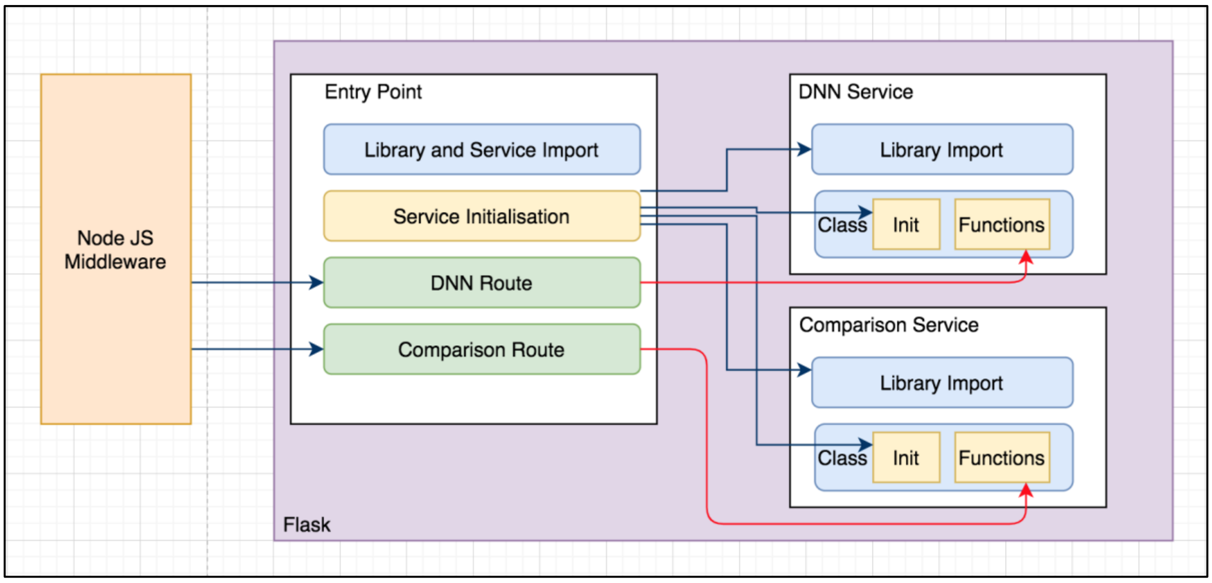


Figure 29 Flask Architecture

Flask framework is a micro service that will be started up along with the React Frontend and the Node Backend. Upon start-up of the Flask service, the main flask application will import the necessary libraries and perform a one-time initialization of DNN and comparison services, since both services loads heavy data files and Natural language processing libraries, constant initialization will degrade the run-time performance of the services. Both service class files contain a init method, the init method is a constructor object in python and this will allow the class to create an instance object of the service, in this case on the run of flask application, the QA service will initialize the training model while the comparison service will load NLP tools and vocabulary models, this instantiation will be persistent throughout the run time of the flask service.

When the Node JS receives an API request to use the Flask hosted service, the Flask routes will receive the HTTP request and trigger the service functions for computation. With the object-oriented design of the Flask service, variable components and computational functions are run independently from each other, eliminating redundant assignments and significantly improving execution time of the services. With an overview of the high-level architecture design of the Flask service, the following sections will dive deeper into the operations of each components.

3.5.2 Intent Prediction on Flask Framework

For the simplicity of this section write-up, the term DNN will be an abbreviation for the Deep Neural Network text classification model. After the training of intentions discussed in Chapter 2, the trained tflearn model is retained and initialized for used in the DNN service. This service is responsible for receiving a FAQ question string, vectorising the input string for model fitting and finally respond with the predicted intention for QA matching.

The training inputs used for this model are permutation for a particular target model question. A model question and answer set is stored for the purpose of question answer matching. Hence this model is only required to identify the closest possible question in the QA set that a user might be asking.

1. When the input intention sentence is passed in from the main flask function, the sentence will be tokenized and stemmed to its root and split into an array of words. Given the tokenized array of the sentence "How can I apply for baby bonuses".

   `[‘How’,‘can’,’I’,‘apply’,’for’,’baby’,bonus’]`

   A word array was first retrieved from a training data set discussed in the neural network training segment in chapter 2. This word array is essentially a list of stemmed words extracted from a set of sentence permutations. This will be represented by a wider array of words, or more specifically 796 unique words collected from the permutation sets.

   The next step would to be create a bag of words to determine the occurrence of the query input in the word list. The following pseudo code would describe the generation of the bag of words.

   ```
   Initialize word list into array B of Len (word list) zeros

   Iterate word S in the query input

   Iterate word W in word list along with an index counter
         If W == S
               B[index] = 1
   ```

   The execution of the above code will create an array of word occurrence defined by zeros and ones. And this will be the input vector that will be used for fitting the model.

2. After vectorising the string input into a binary vector set, the MODEL.PREDICT method will be used to predict the possible true intentions. Upon prediction, the predict

method will output all possible intentions with a confidence value that satisfies a specific threshold. These values will be used as suggestions offered to the user.

```
Vector = Generate Bag_of_words(input)

Predictions = Predict vector [Predictions returns an array of
probability]

Prediction array elements corresponds to model intention where classes[i]
= Predictions[i]

Iterate Prediction and store indexes where probability is more than the
threshold
```

From the returned array, the most probable intention will be extracted and checked against the predefined QA list to determine the response for the query. Finally, the reply, score and next probable intention will be reconstructed into a JSON object and returned to the Node Server.

3.5.3 Response Comparison on Flask Framework

Response Comparator service is responsible for response comparison between the Ask Jamie Chatbot and Other QA models. The API takes in an array parameter which contains two responses. The job of this API is to determine the percentage of match between the responses. Each sentence will be first tokenized into an array of word tokens, stop words are then removed from the tokenized sentence pair to remove redundant storage and computation of additional vocabulary.

Since accuracy probability is numerical data, construction of a bag of words will be needed to build a sentence vector. The first step to building a bag of words is to combine the word tokens of both responses as shown below.

```
Given two string responses:

   1. Baby bonus is a government payment to parents with baby
   2. Parents with baby is entitled to baby bonus

Tokenize and removal of stop words

   1. [‘Baby’, ‘bonus’, ‘government’, ‘payment’, ‘parents’, ‘baby’]
   2. [‘Parents’, ‘baby’, ‘entitled’, ‘baby’, ‘bonus’]

Union of tokens

[‘Baby’, ‘bonus’, ‘government’, ‘payment’, ‘parents’, ‘baby’,
‘entitled’,]

Create bag of words vector for each sentence

Vector 1 = [1,1,1,1,1,1,0]
Vector 2 = [1,1,0,0,1,1,1]
```

Cosine Similarity of response vectors

$$\frac{\boldsymbol{v_1} \cdot \boldsymbol{v_2}}{\|\boldsymbol{v_1}\| \cdot \|\boldsymbol{v_2}\|}$$

$\boldsymbol{v_1} \cdot \boldsymbol{v_2}$

```
For index i in range of bag of words
      Product += V1[i] × v2[i]
```

$\|\boldsymbol{v_1}\| \cdot \|\boldsymbol{v_2}\|$

$\sqrt{\mathrm{Sum(V1)}}$ × $\sqrt{\mathrm{Sum(V2)}}$ (Square omitted because vector values are 1)

From the above equation, Cosine similarity [23] can be easily obtained by dividing the two computed values. The closer the values are to 1 the more closely related the responses are to each other. The cosine similarity algorithm has been a very popular way of quantifying the similarity of sequences by treating them as vectors and calculating their vectors and hence applying the same concept strings can be treated as vectors and the just by comparing the vector sequences word token similarities can be obtained easily.

3.6 Software System Architecture

Each segment of this chapter has gone in depth to describe the various components that are used to make up each micro service. The entire Chatbot framework is feature heavy and these features are distributed across the best performing software system to provide the most optimal and reliable experience to the client. The figure below will showcase the combined system architecture of the Chatbot framework.

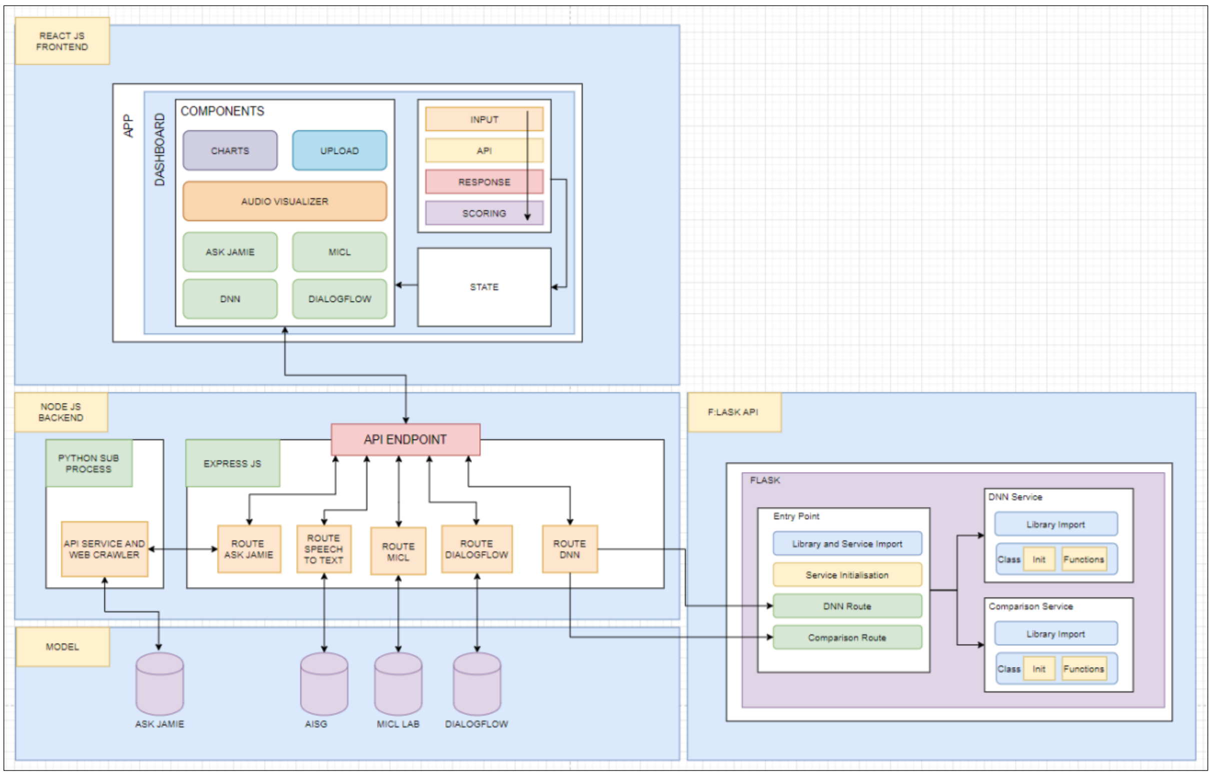


Figure 30 Full Stack System Architecture

4 Batch query input and response analysis

4.1 Overview

This Chapter will discuss an analysis feature integrated to the base framework described in Chapter 3. The overall Chatbot framework is successful in comparing responses of various Chatbot services, however it is not able to provide an overview of either Chatbot performance, therefore the motivation of this feature is to provide an interface for performing batch level testing of inputs to the observe and analyse the accuracy boundary across many test intentions.

This feature was implemented within a short duration due to time constraints. Hence, UI bugs may persist in the run of the application and overall runtime may not be the most optimal. Future upgrades can be applied to improve the overall value of the feature.

4.2 Handling batch of query text

The first problem when performing batch uploads would be to consider the handling of large amounts of requests simultaneously. The process of mass comparison will apply the same concept of a standard one to one comparison where 2 responses are generated and compared with the flask service but at a larger scale. Since React JS operates on a single thread, the responses cannot be compared concurrently, rather an asynchronous method will be used to handle the all the requests.

4.3 Bulk promise handling

To be able to handle a bulk input processing, promises are introduced. Promises are used to handle asynchronous operations in JavaScript and are very easy to manage when dealing with multiple asynchronous events rather than using nested Callback operations which often leads to Callback hell and unmanageable codes.

For this implementation only three methods are required for the entire batch processing requests. The first two method would be the Chatbot comparison benchmark and Chatbot test service, Ask Jamie and any other Chatbot service or platform respectively. The last method would be the response comparison method. Each of this method would contain dynamic parameters and return a promise.

1. The first step of this solution would be to create individual arrays that stores functions of the respective individual Chatbot services that will return a response. After which the 2 arrays are combined into a single array.

```
ArrayAskJamie = [functionAskjamie(input1),
functionAskjamie(input2), functionAskjamie(input3), …]

ArrayDialogflow = [functionDialogflow(input1),
functionDialogflow(input2), functionDialogflow(input3)]

CombinedArray = [ArrayAskJamie, ArrayDialogflow]
```

2. The next step is to call a Promise all on both array of functions promises, this would add the Chatbot functions into a message queue and executed in order. The Chatbot function will resolve a response which will be stored in an array once the Promises are complete.

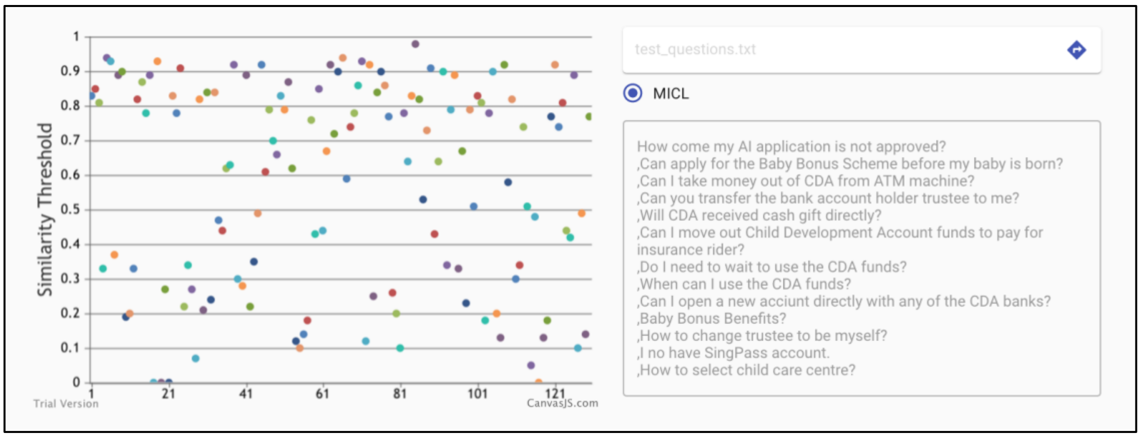
```
Promise.all(CombinedArray.map(Promise.all.bind(Promise)))
.then((response)=>{})

Response = [[AskJamieInput1Response,
AskJamieInput2Response], [DialogflowInput1Response,
DialogflowInput2Response]]
```

3. The final step is to reorganize the responses to load for loading of array of response comparison functions

```
responseComparisonArray =[ responseComparisonFunctoin([response[0][i],
[response[1][i]])]

Promise.all(responseComparisonArray)
```

The result of the promise would be an array of scores between each compared response and this will be set as a React state and passed into the Graph Plotting component enabled by the Canvas JS library. The Canvas library provides various types of graph plots to select from and can be interchanged easily. The graph type decided for batch response analysis would be the scatter plot as there are no relations between each response and a scatter graph would best represent the boundary where majority of response similarity will cluster.

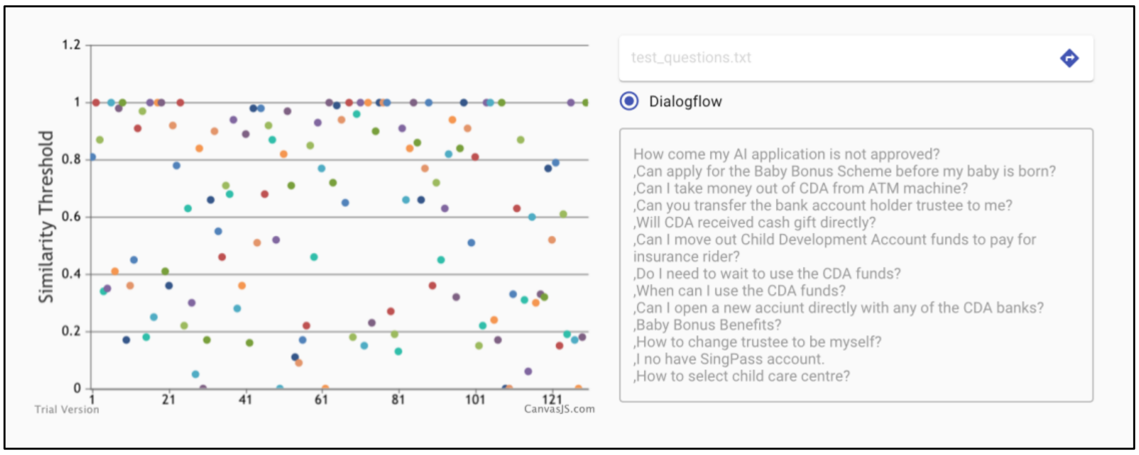
4.4 Graphical visualization of response similarity



MICL

How come my AI application is not approved?
,Can apply for the Baby Bonus Scheme before my baby is born?
,Can I take money out of CDA from ATM machine?
,Can you transfer the bank account holder trustee to me?
,Will CDA received cash gift directly?
,Can I move out Child Development Account funds to pay for insurance rider?
,Do I need to wait to use the CDA funds?
,When can I use the CDA funds?
,Can I open a new acciunt directly with any of the CDA banks?
,Baby Bonus Benefits?
,How to change trustee to be myself?
,I no have SingPass account.
,How to select child care centre?

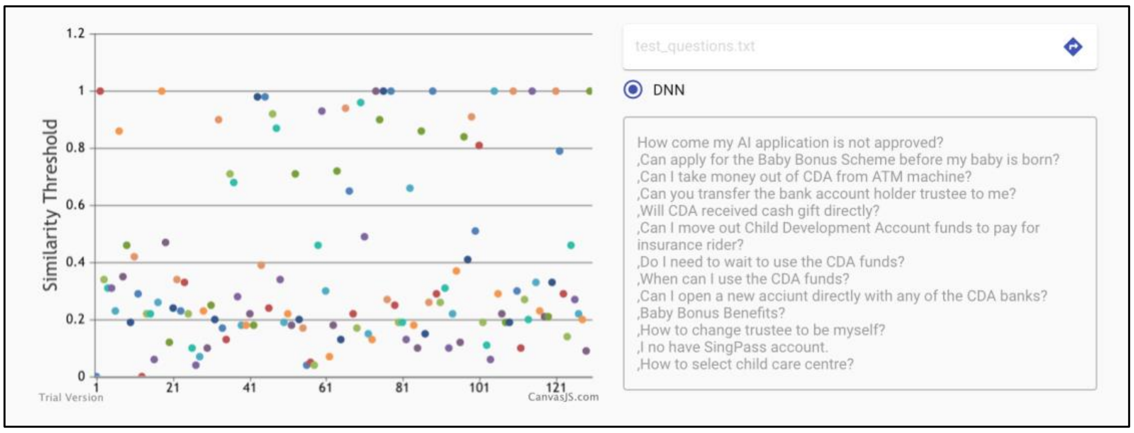
Figure 31 Accuracy Performance MICL



Dialogflow

How come my AI application is not approved?
,Can apply for the Baby Bonus Scheme before my baby is born?
,Can I take money out of CDA from ATM machine?
,Can you transfer the bank account holder trustee to me?
,Will CDA received cash gift directly?
,Can I move out Child Development Account funds to pay for insurance rider?
,Do I need to wait to use the CDA funds?
,When can I use the CDA funds?
,Can I open a new acciunt directly with any of the CDA banks?
,Baby Bonus Benefits?
,How to change trustee to be myself?
,I no have SingPass account.
,How to select child care centre?

Figure 32 Accuracy Performance Dialogflow



DNN

How come my AI application is not approved?
,Can apply for the Baby Bonus Scheme before my baby is born?
,Can I take money out of CDA from ATM machine?
,Can you transfer the bank account holder trustee to me?
,Will CDA received cash gift directly?
,Can I move out Child Development Account funds to pay for insurance rider?
,Do I need to wait to use the CDA funds?
,When can I use the CDA funds?
,Can I open a new acciunt directly with any of the CDA banks?
,Baby Bonus Benefits?
,How to change trustee to be myself?
,I no have SingPass account.
,How to select child care centre?

Figure 33 Accuracy Performance Text Classification

The figures above are response similarity results based off 130 test input sets. Similarity thresholds are between values 0 to 1 with 0 being completely unrelated and 1 being a complete match. Observations from the graph's scatter plot can effectively distinguish the better Chatbot services.

Overall, from a visual analysis of the three Chatbot test services, MICL Lab's QA model provides a better FAQ match, Dialogflow performed moderate given that it was only provided generic training phrases and the self-written text classification model did not perform as well amongst the three due unresolved training faults as mentioned in QA experiment section in Chapter 2. Furthermore, a score of 1 can be observed in Dialogflow and Text Classification models as both were trained with the model responses obtained from Ask Jamie hence this might cause some biasness in the results.

Some downsides to this comparison feature are that specific models have its processing time and some third party Chatbot services may not be too friendly with mass request from users. Hence promise optimization is necessary for the feature to cater for load times and possibly a progression system to cater for user experience.

5 Conclusion and Future Work

5.1 Conclusion

The aim of this project is to build a full stack Chatbot application framework that is fully capable of supporting multi Chatbot interactions, response comparisons between various Chatbot services and offering text based and speech-based input methods. This Chatbot framework hopes to support upcoming Chatbot and speech transcription projects by providing a platform to upload their work for performance visualizations.

This Final report showcases the technical process and engineering practices taken into consideration taken to build a full stack Chatbot framework application. This project adheres to software engineering and system architecture best practices to create an extensible and scalable platform to support future developments

5.2 Contributions

The following section will provide a brief summary of key feature contributions that has been researched and implemented for this project.

5.2.1 Web Chatbot Framework

This project involved the ground up development of a full stack client server application that serves the purpose of performing Chatbot comparison and analysis. The Stack is built with close reference to the MERN (Mongo, Express, React, Node) stack, less the use of a database. This framework uses React JS as the frontend, Node JS as the JavaScript backend and Express JS as the routing layer for RESTful APIs.

5.2.2 Text, Audio file and Speech Transcription Input

Three Chatbot query input methods are applied for Chatbot Service interactions are. A simple text input using form inputs, a file upload method that takes in audio files for transcription as query input and more sophisticated audio live streaming method that offer real-time speech transcription and interaction with the Chatbot.

5.2.3 Response Comparison Feature

A Flask framework was explored and implemented as a response comparison platform, the comparison features leveraged python's Natural Language Processing libraries for tokenization and Numpy library for string vectorising. Vector matching algorithm were researched to find the fastest and reliable comparison.

5.2.4 Batch Response Accuracy Visualization

Advanced feature and value-added services were explored with the decision to build a response accuracy graph to provide an outline of the overall accuracy of Chatbot services when tested against the benchmark model. The implementation of batch response processing made use of promise driven solutions to ensure no losses of queries and an accurate graph presentation.

5.3 Future Implementation

The current build of the Chatbot platform is a successful proof of concept in realizing a Multi Chatbot framework solution for response comparison and live speech transcription integrations. Given the limited time frame to research and develop the application, the functionalities and value add to this framework is limitless. Below are some proposed future implementation to consider to help improve the overall product.

5.3.1 Modular integration of QA engines for comparison testing

Currently, the Chatbot services implemented and separated by components, the state and configuration of each components work independently from each other. However, the current implementation would require the component reference to be present on the main dashboard. To realize a fully modular Chatbot service solution, a Chatbot storage model should be implemented to allow users to add and configure new Chatbot services without having to modify the source codes.

5.3.2 Database Consideration for QA caching and training

The Chatbot framework is a state managed application thanks to the React JS framework. Hence all inputs, response and scores can be stored and reused in other components or other features. This information is only being leveraged by the batch comparison feature, hence an implementation possibility is to make use of this response comparison data to help improve Chatbot services, this can be done by providing a database to record question and answer statuses for each Chatbot service. The framework could also offer Chatbot creators auto training features based on the collected historical data.

6 References

[1] L. J. Beilby, J. Zakos, and G. A. McLaughlin, "Chatbots," ed: Google Patents, 2014.

[2] Anonymous, ‘Ask Jamie Virtual Assistant’. [Online]. “tech.gov.sg” [Accessed: Mar. 2020]

[3] S. Aggarwal, "Modern Web-Development using ReactJS," *International Journal of Recent Research Aspects,* vol. 5, pp. 133-137, 2018.

[4] M.Hamedani*, React Virtual DOM Explained in Simple English. [Online].* “programmingwithmosh.com” [Accessed: Mar. 2020]

[5] P. Teixeira, *Professional Node. js: Building Javascript based scalable software*. John Wiley & Sons, 2012.

[6] Anonymous, *What’s Special about Node.js*. [Online]. “ilovecoding.org” [Accessed: Mar. 2020]

[7] P. Teixeira, *Professional Node. js: Building Javascript based scalable software*. John Wiley & Sons, 2012.

[8] A. Mardan, *Express. js Guide: The Comprehensive Book on Express. js*. Azat Mardan, 2014.

[9] E. Hahn, *Express in Action: Writing, building, and testing Node. js applications*. Manning Publications, 2016.

[10] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[11] J. Forcier, P. Bissex, and W. J. Chun, *Python web development with Django*. Addison-Wesley Professional, 2008.

[12] S. Janarthanam, *Hands-on chatbots and conversational UI development: build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills*. Packt Publishing Ltd, 2017.

[13] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," arXiv preprint arXiv:1712.05181, 2017.

[14] D. Mamgain, *Dialogflow vs Rasa – Which One to Choose? [Online].* “chatboslife.com” [Accessed: Mar. 2020].

[15] Dialogflow, *Quota and limts. [Online].* Available “cloud.google.com” [Accessed: Mar. 2020]

[16] J. Brownlee*, Best Practices for Text Classification with Deep Learning. [Online].* “machinelearningmastery” [Accessed: Mar. 2020].

[17] F. Wittmann, *First Steps with TFLearn*. [Online]. Available “medium.com” [Accessed: Mar. 2020].

[18] E. Atto, *Understanding the fundamentals of State in React*, “medium.com”, 2018

[19] I. Fette and A. Melnikov, "The websocket protocol," ed: RFC 6455, December, 2011

[20] R. Sears, C. Van Ingen, and J. Gray, "To blob or not to blob: Large object storage in a database or a filesystem?" arXiv preprint cs/0701168, 2007.

[21] H.-K. Ra, H. J. Yoon, A. Salekin, J.-H. Lee, J. A. Stankovic, and S. H. Son, "Software architecture for efficiently designing cloud applications using node. js," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*, 2016, p. 72.

[22] Dialogflow, *Setting up authentication. [Online].* Available “dialogflow.com” [Accessed: Mar. 2020]

[23] D. Radecic*, Calculating String Similarity in Python. [Online].* Available “towardsdatascience.com” [Accessed: Mar. 2020]