Building a Machine Learning Powered Chatbot for KSU Blackboard Users

Qubayl Alqahtani, Omer Alrwais
Information Systems, King Saud University, Riyadh, Saudi Arabia

Abstract—Chatbots have attracted the interest of many entities within the public and private sectors locally within Saudi Arabia and also globally. Chatbots have many implementations in the education field and can range from enhancing the e-learning experience to answer students’ inquiries about course schedules and grades, tracking prerequisites information and elective courses. This work aim is to develop a chatbot engine that helps with frequently asked questions about the Blackboard system, which could be embedded into the Blackboard website. It contains a machine-learning model trained on Arabic datasets. The engine accepts both Arabic textual content as well as English textual content if needed; for commonly used English terminologies. Rasa framework was chosen as the main tool for developing the Blackboard chatbot. The dataset to serve the current need (i.e. Blackboard system) was requested from Blackboard support staff to build the initial dataset and get a sense of the frequently asked questions by KSU Blackboard student users. The dataset is designed to account for as many as possible of KSU Blackboard related inquiries to provide the appropriate answers and reduce the workload of Blackboard system support staff. Testing and evaluating the model was a continuous process before and after the model deployment. The model post-tuning metrics were 93.4%, 92.5%, 92.49% for test accuracy, f1-score and precision, respectively. The average reported accuracy in similar studies were near 90% on average as opposed to results reported here.

Keywords—Chatbot; RASA; conversational agents; machine learning

I. INTRODUCTION

A chatbot is an artificial intelligence (AI) software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, mobile apps or through the telephone [5]. It’s an environment that receives questions from users in natural language, relates these questions with a knowledge base, and then answer based on pre-defined answers. Chatbots are more formally referred to in the literature as conversational agents or conversational assistants. The core principle of every conversational agent is to interact with humans using text messages and act as it were able to understand the user and replay with the appropriate message. The origin of computers talking to humans back to the start of the computer science field itself. Alan Turing defined a simple test referred to now as the Turing test back in 1950 where a human judge would have to predict if the entity they are communicating with via text is a computer program or not [6]. However, this test’s scope is way greater than the case of chatbots, the main difference being that the domain knowledge of a chatbot is narrow compared to the Turing test. Turing test assumes one can talk about any topic in mind with the agent.

Conversational agent environment consists of five different main parts [7]. Starting with user messages: they are a dynamic input received by the agent to process and replay. They contain a string representation of the actual text sent by the user, and a metadata that contains additional information like a reference to the session the conversation belongs to, and possibly the date and time the message was sent to the agent, on which platform the message was sent from if the agent is linked to more than one message, etc. The agent receives the message along with the information it contains in a read-only mode only with no possible means of making changes to it. The backend is one of the significant parts of the environment that the agent has access to. It contains additional information about the agent users and the database's states to store the user messages, their metadata, and keep track of the conversation events. The agent can view and update certain aspects of the backend. The chatbot can also obtain new information from the user if necessary, by asking the user to provide it.

RASA is a modular design framework proposed by [1] that consists of two main components, Rasa Core for dialog management and Rasa NLU for natural language understanding. Those are open-sourced python libraries for building machine-learning based conversational agents. They provide dialog management and NLU capabilities in an easy manner. By nature, a modular by design architecture allows for easier integration of modules with other systems and services. For instance, Rasa NLU can be used as a service in a different system other than rasa by exposing HTTP APIs for external requests and vice versa for Rasa Core. The code can be found by visiting: https://github.com/RasaHQ.

Although chatbots have been present for a long time, 2016, before the spring, was the true start of this technology. There are two main reasons for the renewed interest in chatbots (1) massive advances in artificial intelligence (AI) and (2) a major usage shift from online social networks to mobile applications such as WhatsApp, Telegram, Slack, and many more advances in AI holds a promise that intelligent chatbots are in fact, can be within reach. The increased usage of mobile applications attracts service providers to reach users through them. However, in spite of these advances, chatbot applications entail many challenges that need to be overcome in order to reach desired goals. Chatbots not only imply changes in the interface between users and technology; they imply changing user dynamics and usage patterns. A recent study indicated that 56% of chatbot users were interested in ordering meals from...
restaurants using chatbots, while 34% had already ordered at least one meal [2]. Chatbots are considered to be beneficial for retailers in terms of customer service (about 95%), sales/marketing (about 55%), and order processing (about 48%) [3]. Generation Z and Millennials are more interested in using chatbots: 25% of a global sample aged 18 to 34 opted for a personal shopping chatbot [4], and the students using Blackboard system fall under this age range.

King Saud University (KSU) is the largest public university in Saudi Arabia and at any time encompass thousands of students whom some of them use and struggle with the online course and learning management system (Blackboard), as such the customer support staff (which is not more than three employees) at KSU is overwhelmed with enquiries which can cause great user dissatisfaction and affect the adaption of Blackboard system by KSU students. The goal of this work is to build a Minimum Viable Product (MVP) for an Arabic Chatbot which is intended to serve users of Blackboard system from King Saud University students by answering their frequently asked questions about the Blackboard system to reduce the load of answering repeated, one answered questions and allow customer service staff to focus on more dynamic issues that require human intervention.

II. RELATED WORK

A revision for several chatbot related papers that highlights the usage of chatbots in the education field was conducted, along with other chatbot usages in different areas such as retail and government entities. A review and summarization of those implementations are discussed in the next paragraphs.

A. Chatbots in Education

A chatbot named EASElective that was built to advise students on what to choose as an elective course was proposed in [22]. EASElective is a conversational agent that was built to supplement existing academic advising systems. It has an interactive, online interface that supports basic official course information to informal students' opinions about that course. Its major components included intent detection, conversational management routines, dialogue design, course information management, and a collection of analyzed students' peers' opinions. In this study, a survey was conducted to capture students' perceptions of the chatbot. The subjects were briefed about the chatbot's purpose and instructed on how to use it and were given up to a half-hour to interact with it and then fill the surveys. The survey results showed that many students preferred to either ask their friends for course information. Around 22% preferred to ask the program leader or use the official university website instead of the chatbot. There were a number of limitations, including the chatbot not having enough interactions to learn from before going live. And also, the chatbot patterns usages are neither recorded nor pre-defined in advance to prepare the appropriate responses.

Another chatbot implementation to enhance the LMS experience was proposed by [23]. This model classifies the main keywords that could be asked by students using R programming language, and this classification is then used in an Artificial Intelligent Markup language (AIML) script as a query. If this query was unsuccessful, it would run against SQL lite. If neither AIML nor SQL lite worked, then the student query will be transferred to a human agent to take over and answer the query. Although the implementation of AIML scripting language is easy and also free to use as a scripting language, this model is a rule-based model and is less tolerant to changes in users' input and, thus, harder to capture the user intent.

Another study for developing a chatbot for university inquiries was put forward by [24]. This study discussed the development of a deep-learning based chatbot using RASA framework. RASA has many connectors to be used in integrating it with communication platforms. One of them is for Facebook. This chatbot is integrated with FB as the majority population is using FB as their main social media channel. This chatbot uses Long Short-Term Memory (LSTM), which is a recurrent neural network architecture that is used in deep learning. This architecture is included in RASA framework. Although the chatbot performed well in terms of intent classification and provided the appropriate replies, there was a platform limitation as they had to perform platform-specific steps to run the chatbot on Facebook, which can result in some limitations to the interaction with the chatbot.

A chatbot for instantly answering students' questions to reduce teacher's workload was proposed in [25]. It supports multiple common social platforms, including Telegram, Facebook Messenger, and Line. The chatbot can reply to commands and natural language questions. Once the instructors transfer the course-related data to an internet database, the chatbot can reply to questions about the course materials and logistics (e.g., course plan). It also supports student login to provide profile-based answers such as the schedule of student registered courses.

B. Chatbots in other Fields

Chatbots also have many usages in other fields besides education. Some of those applications are in healthcare, such as self-diagnoses based on symptoms, using chatbots as a communication means in e-commerce websites, providing account data and paying bills in banking, etc. Below are some of the related works of chatbots.

A text-to-text chatbot engages patient's medical issues were proposed by [26]. It's a medical chatbot that diagnoses diseases using AI. This chatbot was built to reduce medical costs and improve patient's accessibility to medical knowledge. In this chatbot, a series of questions about the patient's symptoms are asked to give suggestions that help in clarifying the disease. The accurate disease is found based on the user reply to those series of questions, and in case of major diseases, a doctor is suggested to be consulted. The patient's past responses are recorded, and in order to reach an accurate diagnosis, the patient is asked more specific questions. There are three main components of the system, which are (1) user validation and symptoms extraction from the conversation, (2) mapping of extracted, potentially ambiguous symptoms to their corresponding database codes, and (3) personalized diagnosis and referring the patient to a specialized doctor if required. The sole focus of this system is extracting symptoms by analyzing natural language using NLG components, which in term makes it easier and less technical for the end-user.

King Saud University (KSU) is the largest public university in Saudi Arabia and at any time encompass thousands of students whom some of them use and struggle with the online course and learning management system (Blackboard), as such the customer support staff (which is not more than three employees) at KSU is overwhelmed with enquiries which can cause great user dissatisfaction and affect the adaption of Blackboard system by KSU students. The goal of this work is to build a Minimum Viable Product (MVP) for an Arabic Chatbot which is intended to serve users of Blackboard system from King Saud University students by answering their frequently asked questions about the Blackboard system to reduce the load of answering repeated, one answered questions and allow customer service staff to focus on more dynamic issues that require human intervention.
Another example of chatbot usage in e-commerce to support customers in their website journey is called “SuperAgent” [27]. This chatbot scrapes public e-commerce websites’ content of products description, user questions and answers, and product reviews and feeds them to its knowledge base. It uses NLP techniques to understand users’ text and machine learning techniques to predict responses to it, including opinion mining for product reviews, fact QA for product information, and FAQ search for customer reviews and chat for greetings and goodbyes.

ChatPy is one of the chatbot implementations in the wholesale business [28]. “Mundirepuestos” is a wholesaler responsible for predicting intents, extracting entities and retrieving responses. It uses the saved model in the filesystem. The Core component is responsible for choosing the appropriate next action with regards to the conversation context and uses Tracker store to store the conversation states, messages and metadata.

To avoid issues in [22], the system needs to be internally deployed and used by diverse students’ backgrounds while recording their usage patterns and interactions which is recorded by default in Rasa framework. Rule-based chatbots like the one presented in study [23] cannot learn which is not the case for Rasa as it has interactive learning capabilities which allows it to learn and refrain from making the same mistake in the future. Unlike the cases presented in study [24] and [28], this work is going to use Rasa API to communicate with the chatbot which remove the platform specific limitations and allows for more customizations. There needs to be a fit number of participants to fairly evaluate the chatbot to overcome the limitation in study [25]. As opposed to the case presented in study [26], Rasa provides a fallback policy that can be triggered when the prediction of the action to be taken is below a specified threshold; this fallback could be used to ask the user to rephrase or show some buttons for the user to choose from. Knowing the user intent can help greatly in providing the right answer to the user question and also helps in performing actions based on the user intent; Rasa uses deep learning embeddings to detect user intention which is not the case in study [27].

III. SYSTEM DESIGN

Fig. 1 below gives an overview of Rasa open-source architecture that consists of two main components which are the Rasa NLU and Dialog management (Core). Rasa NLU is a summary of chatbot related works in different fields is shown in Table I.

<table>
<thead>
<tr>
<th>Title</th>
<th>Field</th>
<th>Method</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developing a Chatbot for College Student Programme Advisement [22]</td>
<td>Education</td>
<td>Intent detection, conversational management routines, dialogue design, course information management.</td>
<td>Minimal interactions to learn from before going live. Patterns usages are neither recorded nor pre-defined in advance to prepare the appropriate responses.</td>
</tr>
<tr>
<td>Enhancing LMS Experience through AIML Base and Retrieval Base Chatbot using R Language [23]</td>
<td>Education</td>
<td>Artificial Intelligent Markup language (AIML) script as a query, SQLite.</td>
<td>Using a rule-based model only.</td>
</tr>
<tr>
<td>Developing FB Chatbot Based on Deep Learning Using RASA Framework for University Enquiries [24]</td>
<td>Education</td>
<td>Long Short Term Memory (LSTM) a recurrent neural network architecture included in RASA framework.</td>
<td>Facebook-specific implementation to run the chatbot which limits customization.</td>
</tr>
<tr>
<td>Using a Multiplatform Chatbot as an Online Tutor in a University Course [25]</td>
<td>Education</td>
<td>DialogFlow’s ML engine and knowledge base.</td>
<td>Low number of participants for evaluating the model.</td>
</tr>
<tr>
<td>ChatPy: Conversational agent for SMEs: A case study [28]</td>
<td>Business</td>
<td>DialogFlow’s ML engine and knowledge base.</td>
<td>Facebook-specific implementation to run the chatbot which limits customization.</td>
</tr>
</tbody>
</table>

![System architecture overview](image-url)

To avoid issues in [22], the system needs to be internally deployed and used by diverse students’ backgrounds while recording their usage patterns and interactions which is recorded by default in Rasa framework. Rule-based chatbots like the one presented in study [23] cannot learn which is not the case for Rasa as it has interactive learning capabilities which allows it to learn and refrain from making the same mistake in the future. Unlike the cases presented in study [24] and [28], this work is going to use Rasa API to communicate with the chatbot which remove the platform specific limitations and allows for more customizations. There needs to be a fit number of participants to fairly evaluate the chatbot to overcome the limitation in study [25]. As opposed to the case presented in study [26], Rasa provides a fallback policy that can be triggered when the prediction of the action to be taken is below a specified threshold; this fallback could be used to ask the user to rephrase or show some buttons for the user to choose from. Knowing the user intent can help greatly in providing the right answer to the user question and also helps in performing actions based on the user intent; Rasa uses deep learning embeddings to detect user intention which is not the case in study [27].

III. SYSTEM DESIGN

Fig. 1 below gives an overview of Rasa open-source architecture that consists of two main components which are the Rasa NLU and Dialog management (Core). Rasa NLU is responsible for predicting intents, extracting entities and retrieving responses. It uses the saved model in the filesystem. The Core component is responsible for choosing the appropriate next action with regards to the conversation context and uses Tracker store to store the conversation states, messages and metadata.
Fig. 2. Rasa architecture.

Rasa ensures that messages are being processed in the right order using the lock store. Actions are running on the Action Server and executed when called by the Core component. Fig. 2 shows the message flow and how Rasa architecture works.

The user first types in a message; this message is then passed to the interpreter in which NLU is used to extract user intention, intent for short, and any entities contained in the text. The conversation state is then saved in a Tracker object, and an event is created, i.e., new message arrival. The state is then received by the policy, and the next action is chosen by the policy to be taken. The action is also logged into the Tracker and then implemented, which could be a response that is based on an external API call or a simple text response that is sent back to the user.

A. Data Collection

To gather the required data for training the chatbot to answer most frequent questions asked by students, the LMS Blackboard team admin was contacted to provide this data. This data is in the form of a Word document format and will be used to manually generate the training data and build chatbot stories to train the chatbot. It contains 17 of the frequently asked questions by the Blackboard users. Examples are shown in Appendix 1.

There are two datasets to be built, one for the NLU model which contains examples for each user intent along with labeled entities. The data for this project are in MSA, Modern Standard Arabic and some common English words. The second dataset is for the Core model or the Dialog Management model, it has all the possible flows for the conversation (intents with their corresponding actions). The latter might not be needed when using mapping policy which maps each intent to an action or a template. The dataset to serve the current need (i.e., Blackboard system) was requested from Blackboard support staff to build the initial dataset, and get a sense of the frequently asked questions by KSU Blackboard student users. This data will be then increased by synthesizing text that could be asked by the chatbot users to increase the chances of understanding students’ questions about the Blackboard system. For chatbots systems, the datasets should be continuously updated after deployment for continuous enhancement. The data formats for both NLU and Core are written in a user-friendly format to make it easier to build, revise and edit. For the NLU model, examples for each intent along with labeled entities are created. There are two available formats for building the dataset, either as a json or a markdown format. Markdown formats are the most used as it can be rendered by most text editors.

B. Building NLU Corpus

The main goal of building this corpus is to make the chatbot see many examples of what the user might say regarding a specific intention of the user. There are two available formats for building the dataset, either as a json or a markdown format. Markdown formats are the most used one as it can be rendered by most text editors. Below is an example of markdown NLU dataset record.

```markdown
1. ## intent: FAQ_login_issue
2. `-system: "أمتيض المدخل إلى نظام إدارة التعلم"
3. `-intent: "FAQ_login_issue"
4. `-entities: []
5. `-start": 19,
6. `-end": 33,
7. `-value": "نظام إدارة التعلم",
8. `-entity": "system"
9. `}
```

The other format is the json format; it’s not sensitive for whitespaces and better in exchanging data among applications. The actual NLU corpus can be found in Appendix 2.

C. Building Stories

Stories are a type of data that is used in order to teach the chatbot the possible messaging flow with user. Markdowns are used to specify the conversation paths i.e., stories. Below is an example for Dialog management model training data.

```markdown
1. ## story 84854865
2. * greet
3. - utter_ask_howcanhelp
4. - utter_ask_issue
5. * inform("system","نظام إدارة التعلم")
6. - action_replyissue
```

The naming convention for stories is to start with two hashes, followed by the story name. Actions are events that start with a dash. The actual Core model corpus can be found in Appendix 3.

D. Implementation

Rasa environment requires a list of hard and software requirements for running Rasa on Docker. Although there are minimum hardware requirements on Rasa official website, the hardware requirements depend on the size of the model and training data as the training time and the size of the NLU data are positively correlated. Those requirements need to be met to
develop the chatbot and train it in a productive manner. Markup language will be used for building the dataset, defining stories and domain, for training and testing the model we will use the command line interface. Python 3.6 or higher will be used for developing the chatbot actions and replies. And finally, docker will be used to host the chatbot system. The domain is the context the chatbot operates on. It is the place where user intentions or intent, entities, actions, responses and slots can be defined in and the chatbot should know about. The domain.yml file is the file where the domain is specified on and can be found in Appendix 4. For the initial model configuration, the suggested configurations by Rasa official website will be used and the data will be trained on that configuration. In the testing and evaluation phase, the model will be fine-tuned and evaluated to select the best parameters for the model configuration.

IV. TESTING AND EVALUATION

As opposed to traditional software testing techniques such as unit tests and functional tests, Rasa has specific types of tests which are the data validation test, the NLU model test, and dialog management model test. The purpose of data validation is to make sure that there are no typos or major inconsistencies in the data or the domain. Fig. 3 indicates that there are no errors or inconsistencies in the chatbot data. If there were errors in the training data, they must be fixed and the model needs to retrain as errors will cause the model to stop working or produce unwanted behavior.

By synthesizing test stories, we can simulate users’ interactions and test the chatbot on a data the chatbot did not see before. This will allow us to see if the model will behave in an expected manner when provided with certain data. The test stories are similar to the training stories with a single difference which is the user message. To test the chatbot, three to four test stories were written on each intent in a total of 61 test stories and these test stories are placed in ‘tests/test_stories.yml’. Those test stories can be found on Appendix 5. These test stories are written by the chatbot developer in a way that simulates actual interaction with the chatbot. The purpose of these tests is to see if the dialog model predicts the next action in a conversation correctly. For example, when the user sent "goodby" and one time as “affirm”. Also we can see that the intent ‘FAQ_submit_button_is_not_working’ was two times falsely predicted as ‘FAQ_in_lms_sound_issue’ and so on. This graph is particularly helpful in optimizing the NLU model by adding more examples and removing examples that might mislead the model into falsely predicting intents. The intent prediction confidence distribution histogram in Fig. 5 is used to show how many samples were correctly and wrongly predicted along with the confidence of the prediction. For our model to perform well, we need to try to minimize the number of samples that were wrongly classified which will automatically increase the correctly classified sample.

![Fig. 3. No Conflicts in dialog model data (stories).](image)

**TABLE II. NLU CROSS VALIDATION RESULTS**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Accuracy</td>
<td>1</td>
</tr>
<tr>
<td>Train F1-score</td>
<td>1</td>
</tr>
<tr>
<td>Train Precision</td>
<td>1</td>
</tr>
<tr>
<td>Test Accuracy</td>
<td>0.924</td>
</tr>
<tr>
<td>Test F1-score</td>
<td>0.911</td>
</tr>
<tr>
<td>Test Precision</td>
<td>0.922</td>
</tr>
</tbody>
</table>

![Fig. 4. NLU intent confusion matrix.](image)
From Table III, we can see that all actions were predicted correctly with a value near to 1 for F1-score, precision, and accuracy. The reason for such high results is that the dialog management model is classifying actions based on the results of the intent classifier. If there are no errors in predicting the intention of the user, the prediction of the next action becomes easier and hence, result in a high hit rate.

**TABLE III. DIALOG MANAGEMENT RESULTS**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>335/338</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.991</td>
</tr>
<tr>
<td>Precision</td>
<td>0.992</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.991</td>
</tr>
</tbody>
</table>

As mentioned earlier, we will try to optimize the NLU model by adding more examples and removing examples that might mislead the model into falsely predicting intents. We will also change some of the NLU model configurations to see if those changes yield better results (Table IV).

**TABLE IV. PRE-TUNING VS. POST-TUNING METRICS**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Pre-Tuning Score</th>
<th>Post-Tuning Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Accuracy</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Train F1-score</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Test Accuracy</td>
<td>0.924</td>
<td>0.934</td>
</tr>
<tr>
<td>Test F1-score</td>
<td>0.911</td>
<td>0.925</td>
</tr>
<tr>
<td>Test Precision</td>
<td>0.922</td>
<td>0.925</td>
</tr>
</tbody>
</table>

Although those are minor changes, they do have an effect and it means that it’s possible to further optimize the model by adding more data and tuning the model parameters to find the ones that best fit the data. The average reported accuracy in similar case studies mentioned in [22], [24], [27] is near 90% as opposed to our results which is slightly higher.

**V. CONCLUSION**

This work intended to develop a chatbot engine that helps with frequently asked questions about Blackboard system, which could be embedded into Blackboard website. It contains a machine-learning model trained on Arabic datasets. The engine accepts both Arabic textual content as well as English textual content if needed; for commonly used English terminologies. The interactions with the chatbot, as well as the users’ evaluations, are stored and used for optimizing the chatbot model to improve future interactions. Chatbot systems development entails many challenges in terms of preparing the training dataset in a way that incorporate as much as possible of users’ inquiries without confusion, preprocessing it before feeding it to the NLU model to try to normalize the data and remove unnecessary words and symbols that could confuse the model, and deploy and maintain the model to be used. Rasa framework was chosen to as the main tool for developing the Blackboard chatbot.

The actual chatbot implementation started by preparing the datasets required for Rasa NLU and Core models. The dataset is designed to account for as many as possible of KSU Blackboard- related inquiries to provide the appropriate answers and reduce the workload of Blackboard system support staff. When the data was ready, the model training and tuning began along with a number of experimentations to find the best model pipelines that fits the data. The chatbot is built using a combination of tools such as Python for programming, YAML as the markup language.

For future work, the chatbot should be deployed using Docker and Docker-compose for running the chatbot service. The chatbot can also be deployed in a distributed cluster either on cloud or on- premise to handle the workload and make the chatbot system scalable.

**REFERENCES**


www.ijacsa.thesai.org
Appendices

Appendix 1: Raw Data: Below is a snippet of the raw data used in building the chatbot. For the full file please download it through this link: https://drive.google.com/drive/folders/1zKxMGrqJYUCyAcHgUM1Mo-gA-m5g8Ag?usp=sharing.
Appendix 2: NLU Corpus: Below is a snippet of the NLU data used in building the chatbot intent classifier. For the full file please download it through this link: https://drive.google.com/drive/folders/1TN3ylufYDXm7FNwdNn-AbfhC70kBWyg?usp=sharing.

```
- intent: greet
  examples:
  - صلاً الله يحجك
  - صلاً ورحمة الله ورحمة الملك
  - صلاً ورحمة الملك
  - الله يحجك
  - علاء
  - حياء
  - السلام
  - السلام
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء
  - الله يحجك
  - علاء
  - صباح الخير
  - سلام
  - مرحباً الله بالخير
  - مرحباً بالخير
  - حياء

Appendix 3: Core Model Corpus: Below is a snippet of the Core (Dialog) data used in building the chatbot Dialog model. For the full file please download it through this link: https://drive.google.com/drive/folders/1uzF3INWfO6sAv5VziAkrUD6h1v0c-Y8?usp=sharing.

```
- story: FAQ_in_lms_sound_issue_path
  steps:
  - intent: greet
  - action: utter_greet
  - action: action_listen
  - intent: FAQ_in_lms_sound_issue
  - action: utter_FAQ_in_lms_sound_issue
  - action: action_listen
  - action: utter_goodbye
  - story: FAQ_how_to_change_password_path
  steps:
  - intent: greet
  - action: utter_greet
  - action: action_listen
  - intent: FAQ_how_to_change_password
  - action: utter_FAQ_how_to_change_password
  - action: action_listen
  - action: utter_goodbye
  - story: FAQ_how_to_see_alerts_in_lms_path
  steps:
  - intent: greet
  - action: utter_greet
  - action: action_listen
  - intent: FAQ_how_to_see_alerts_in_lms
  - action: utter_FAQ_how_to_see_alerts_in_lms
  - action: action_listen
  - action: utter_goodbye
Appendix 4: Domain File: Below is a snippet of the Domain file used in building the chatbot. For the full file please download it through this link: https://drive.google.com/drive/folders/1GPqHIRBFg9sIwSddSGrUZbQpxxIEryzv?usp=sharing.

Appendix 5: Test Stories: Below is a snippet of the Test Stories used for testing the chatbot. For the full file please download it through this link: https://drive.google.com/drive/folders/1WtU_-plb30Kkjf2ycC2tpVRxl843Y0puh?usp=sharing.