B.Sc Engg. Thesis

Development of a Multilingual Conversational Agent using Deep Learning and Natural Language Processing

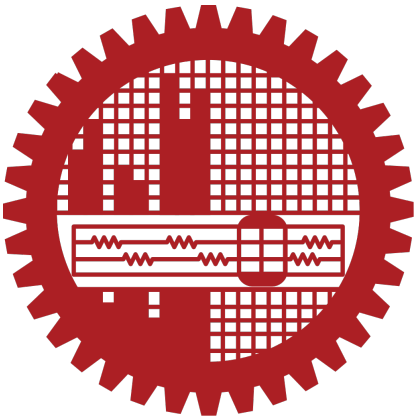
Submitted By-

A F M Mahfuzul Kabir
Student ID: 1706045

Sabbir Hossain Ujjal
Student ID: 1706146

Supervised by-

Dr. Mohammad Ariful Haque
Professor
Department of Electrical and Electronic Engineering



Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh

May, 2023

Certificate of Approval

We hereby would like to declare that this thesis titled "Development of a Multilingual Conversational Agent using Deep Learning and Natural Language Processing" has been composed and authored by A F M Mahfuzul Kabir and Sabbir Hossain Ujjal, under the supervision of Professor Dr. Mohammad Ariful Haque, Department of Electrical and Electronic Engineering. We would also like to declare the following

- This thesis has been written only to partially fulfill a requirement for the award of a graduation degree
- All contents of this document have been written by the authors, unless stated and referred.
- All external sources have been appropriately attributed

Authors

A F M Mahfuzul Kabir

Sabbir Hossain Ujjal

Signature of the Supervisor

Dr. Mohammad Ariful Haque

Professor
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology (BUET)

Acknowledgments

It fills us with profound joy to have completed our thesis work on the topic titled "Development of a Multilingual Conversational Agent using Deep Learning and Natural Language Processing". Firstly, we would like to thank the Almighty for giving us patience, integrity and most importantly, for keeping us in good health to have been able to put through effort in completing this manuscript.

We are indebted to our respected thesis supervisor, Dr. Mohammad Ariful Haque, Professor, Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, for his beneficent guidance throughout the whole research work. His proficiency in the relevant area and structured supervision effectively assisted us to finish this research work successfully. He constantly motivated us to think out of the box and brought our best possible potential whenever we faced any problem and couldn't find a way out. His encouragement and supervision have helped us in improving our skills as an individual.His invaluable insight led us to our desired destination with this research work. We are grateful to him for being a resolute guide in our endeavor. We also pray and hope that he will continue to do so in foreseeable future. We also express our gratitude towards our friends, our families and everyone who have motivated us throughout this journey.

Abstract

Conversational Agents, also known as CAs has been booming in the recent years. Almost every big tech companies such as Google, Apple, Amazon etc. have their own solution to it. Recent advancement with OpenAi's ChatGPT and Google's Bard is taking such agents to another level. As time goes forward, the use of such agents will be on a large scale. So, to keep up with the future of technology, one can never bear the cost of lagging behind. Conversational agents are artificial intelligence-driven software application that endeavours to replicate human behaviour through engaging in conversation and interaction with users via natural language. And task-oriented agents are designed to use to do activities such as answering questions or solving basic queries. Due to their innate capacity for understanding human behaviour, they are experiencing a growing trend in popularity. However, most of the state of the art solutions to such agents do not support multilingual approach. In fact, only Google's android assistant can use two languages at once, others can't. Such difficulties in using CAs result in difficulties for people with poor knowledge of English and technically unaware people. But the main goal of such agents is to make technology easier and available for everyone. Thus, we propose **a multilingual approach to conversational agents using deep learning architecture**. For multilingual purpose, we use Bengali and English. Bengali is a low resource language, meaning the available datasets for deep learning applications in Bengali are very low. Our proposed system ensures that low resource languages, such as, Bengali does not lag behind because of poor availability of datasets. Our developed conversational agent is an end-to-end voice controlled system. We utilize Natural Language Understanding (NLU) technology for understanding human command and other state of the art technologies to implement the agent . The user command is transcribed to relevant language using language identification (LID) and automatic speech recognition (ASR) models. The overall system can detect intents, keywords of user command and act or generate response per user command. The agent can be used hand-free and can perform tasks such as weather and time queries, hospital & restaurant search etc.

Keywords: Conversational Agent, Machine Learning, Deep Learning, Natural Language Understanding, Automatic Speech Recognition, Dialogue Management, RASA, Command, Actions, Response, API.

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1

Introduction

In this chapter, we provide a general overview on the functionality of a Conversational Agent, what does it refer to, the features and application of conversational agent in modern technology. Furthermore, we state the motivation and contribution behind our thesis work.

1.1 Objective

In this era of Artificial Intelligence (AI), Conversational Agents (CA) are very demanding and progressive sector of science. In this fast changing era of technology, different variants of CAs are being developed in recent years to make peoples' day-to-day life easier. The use of voice controlled assistant is not "a science fiction fantasy" anymore, it's real. Big tech companies have been developing various kinds of CAs from the early 21^{st} century. Starting from Apple's "Siri", Amazon's "Alexa" or Microsoft's "Cortana", almost every big tech companies have their own voice controlled assistant now-a-days.Recent advances in deep learning and natural language processing have made it simpler to deploy modern conversational agents.

Conversational agents can be useful in a variety of ways in our daily lives. They can assist us in automating services such as responding to frequently asked questions on websites, answering inquiries in educational services, automated call center services and so on. A voice controlled end-to-end system can allow people to experience modern technologies with hands-free access. In such case, modern technologies can be used with only voice command and the devices will response on their own. This will be useful for persons who are technically incompetent and are unable to use modern technology such as web-surfing and data retrieval from the internet. Disabled people can also benefit from such agents to make current technologies easier to use.

English is a high-resource language due to its widespread popularity. It is used practically everywhere in the world to communicate between individuals of all backgrounds. As a result, all

current technologies are increasingly focusing on English as the primary language. This creates a large gap in modern technologies for less popular languages. The utilization of multilingual technology has the ability to bridge the gap between less widely spoken languages and modern technologies. It has the ability to solve the current language barrier in modern technologies for developing nations such as, Bangladesh. The rural populations of such nations are rarely fluent in English. A multilingual conversational agent can assist people in using current technology with ease and overcoming the language barrier.

Availability of data for deep-learning base system needs huge data for training the system. Because of popularity it easier to collect huge amount of datasets for English language. As deep learning models are benefited hugely by big data, English has an unavoidable advantage over others. The simple nature of construction of English semantics (compared to Bengali) plays a vital role in this field as well. As the world moves towards the age of Artificial Intelligence, falling behind is not a viable choice. So, Conversational agents that support low-resource languages can open the door to many unimaginable possibilities. And if it's multilingual, it has the potential to be transformative in conversational agent domain.

So we propose a multilingual task-oriented conversational agent. Our proposed system is end-to-end voice controlled, meaning, a user won't have to touch the device at all to perform several tasks or inquire queries. The use of the proposed multilingual conversational agent has the potential to serve as an automated system for executing diverse tasks in various languages through voice commands, thereby providing a user-friendly experience to individuals who may lack technical expertise or proficiency in the English language.

1.2 Conversational Agent: Definition and Overview

Conversational agents (CAs), are computer programs that can understand natural language and response in natural language with human users [1, 2]. These conversations can take the form of casual chit-chat in which case the system is commonly known as a chatbot or task-oriented conversation in which case the system is known as a task-oriented agent. While some CAs exclusively employ text-based input and output, others use more sophisticated input and output modalities such as speech or by manipulating a physical or virtual body. With the help of modern machine learning facility, modern CA are enhanced with the capability of carrying out meaningful conversations with human user and learning to provide better and more relevant responses, broadening their knowledge-base and taking action that human user are desired for.

1.3 Different types of Conversational Agents

In modern era of technology, different kind of conversational agent are invented depend on their applications and usage. Most common of them are discussed here.

1.3.1 Text based conversational agents

A text-based agent is a conversational agent that interacts with users through text command, typically in the form of messages exchanged in a chat interface. Text-based agents make use of natural language processing (NLP) techniques to interpret user input and deliver suitable responses.Text-based agents have the capability to be implemented across a range of platforms and applications, such as websites, messaging applications, customer support systems, and virtual assistants.

1.3.2 Voice based Conversational Agents

This kind of conversational agents interact with user through voice. They take are designed to understand queries, commands, and requests in voice or speech and also response in natural voice. These agents are consists of automatic speech recognition (ASR) unit which converts human speech command to text and natural language understanding (NLU) techniques to interpret the meaning and intent behind and keywords in the command and text to speech (TTS) are used to response in voice. Voice-based conversational agents are becoming increasingly popular among consumers these days.

1.3.3 Task Oriented Conversational Agents

Task Oriented CA are designed for accomplished specific objectives such as searching restaurants, news query, ordering pizza or querying up-to-date data. These kind of agents are focused on some specific domain to work with and are best suited where there are well-defined tasks or actions that users want to accomplish. Some of the main domains for task oriented CAs are e-commerce, education, FAQ etc.

1.3.4 Non Task Oriented Conversational Agents

These type of conversational agents are not focused on any specific domains like in task oriented CAs rather they are focuses on engaging in open-ended conversations with users. primary purpose is to engage users in meaningful and pleasurable discussions, delivering knowledge, entertainment, or companionship.

1.4 Application of Conversational Agents

In this section we will give an overview of applications of modern CA. Task oriented CA are becoming popular in recent years and helping creating interesting applications for users.

1.4.1 Health Care

Conversational agents in healthcare have the potential to improve access to information, enhance patient engagement, and optimize healthcare delivery. They can help to address the gaps in quality healthcare. Health care conversational agents also known as "Healthbots" are being used with varying functions across a range of healthcare domains like patient-facing health bots for initial screening, mental health (i.e., depression, anxiety), nutrition etc [3]. There are several applications accessible for health care service that may act as an agent and assist in receiving optimal health care.

1.4.2 Education

Conversational agents are extremely beneficial in the educational area. They can be utilized as a teaching assistant, offering accurate information and responding to questions. In addition, they are proven to be beneficial in conducting searches for important data such as admission protocols, inquiries regarding relevant information, and retrieving current information associated to a particular topic. Almost all university websites now using a chatbot to answer basic FAQs, doing learning assessment and feedback etc. which are improving student engagement with university and teachers.

1.4.3 Business and E-commerce

The rise of e-commerce has already transformed the business landscape and continuously growing every year and many company are deploying CAs in their sales and customer service. It has estimated that CA will be able to answer up to 80% of users questions which will save human effort eventually save millions of dollars [4]. CA also has the potential to serve various additional purposes, such as providing product recommendations, tracking orders, gathering customer feedback, and conducting surveys for the companies.In certain instances, the implementation of CA is becoming an indispensable attribute for advanced technology corporations operating in the realm of e-commerce.

There exist additional applications of conversational agents beyond those previously mentioned. These systems have the potential to be utilized in various domains such as travel guidance, professional feedback systems, banking processes, restaurant services, and more.

1.5 Contribution

In this project, we create a solution to a multilingual approach in building a conversational agent. For the start, we use Bengali & English, both of which we understand and are fluent. Our proposed method can overcome several of challenges that stood in the way of a multilingual

speech recognition and natural language processing systems. Our motivation behind this work is to achieve the following:

- Create an end-to-end voice controlled conversational system for Bengali language. As mentioned before, Bengali is a low-resource language. The amount of data publicly available for this particular language is limited.
- Overcome the obstacles faced while creating a multilingual system by integrating both Bengali and English language as an end-to-end system.
- Build a working system for Bengali Speech Recognition. The AI models used for transcription speech data are known as Automatic Speech Recognition (ASR) systems. There aren't much publicly available solutions to ASR systems for Bengali speech transcription.
- Build a working language identification and transcription system that can take input of speech signals in multiple languages, detect the language and then transcribe it accordingly.
- To create a Natural Language Understanding (NLU) system to understand given command and detect intent and entities available in the command that can be in either Bengali or English.
- Design a comprehensive dialogue management system that ensures a seamless user experience.
- Integrate the dialogue system with back-end services to facilitate the appropriate execution of responses for the user.
- To make the end-to-end system fast and efficient enough for consumer use and deployment.
- Open the door to future improvements and research work in this domain for low-resource languages all over the globe.

As there aren't much work done for Bengali language in the voice controlled Conversational Agent domain, we're certain of the impact and usefulness of our work. Not all the motivations behind our work was not achieved in the due time, but surely there's scope of future works and improvement that can ensure that our language are not the reason for us to lag behind.

1.6 Possible Components of Conversational Agent

A conversational agent can have many sophisticated component for working as an end-to-end system and they can very according to the implementing process and desired application. For our proposed method we are intended to implement these following components:

- Language Identification (LID)
- Automatic Speech Recognition (ASR)
- Natural Language Understanding (NLU)

- Dialogue Management System (DM)
- Text to Speech Unit (TTS)

1.6.1 Language Identification (LID)

Language Identification is the process of identifying a language with computational processes. In deep learning, identification of language can be done by audio, text, image etc. A model that identifies language from audio clip is known as Spoken Language Identification (SLID) model [5]. Such model can have various application such as creating front ends for multi-language speech identification systems, automatic customer routing in call centers etc. The methods of SLID includes data collection, feature removal, and language classification [6]. Current state of the art SLID solutions use log-Mel spectrum method to generate spectrograms from audio and further apply convolutional neural network (CNN) for classification [5, 7].

1.6.2 Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) is the process that capables a program to convert human speech to text. It's an important technology that can establish and improve human-human and human-computer interactions [8]. It's nowadays used in several types of application, such as live transcription and captioning, virtual assistant and chatbots, voice commands and interaction etc. Previously, ASR models needed large labeled training data for a decent enough performance. This used to be a problem for low resource languages. The current state of the art ASR systems use a new approach known as Semi-Supervised learning to address to the mentioned problem. The wav2vec 2.0 shows that a pre-trained model on huge unlabeled speech data followed by fine tuning using small amount of labeled data can outperform all previous ASR systems [9, 10]. An acoustic model trained as the process mentioned above, with the help of a language model for better decoding can be used for better interaction with machines and achieve several tasks using only voice command.

1.6.3 Natural Language Understanding (NLU)

Natural Language Understanding (NLU) is a field of Natural Language Processing (NLP) that process and interpret natural language input from users and translate it into machine-readable instructions. NLU unit uses NLP techniques such as intent identification and entity extraction for understanding the content of user input. Intent recognition is the process of identifying the user's objective in input text and is the first and most important part of NLU because it establishes the meaning of the text. Entity recognition is a specific type of NLU that focuses on identifying the entities in a message, then extracting the most important information about those entities. These intent and entries are used in intent-slot filling method for further executing different task in conversational agent. Natural Language Understanding (NLU) unit is the brain of a Conversational Agent (CA).

1.6.4 Dialogue Management System (DM)

Dialogue Mangement System (DM) manages and controls the flow of conversation between user and the agent. Its principal job is to comprehend the conversation's context and make decisions about what the agent should say or do next in on context of that context. The DM system typically receives input from the Natural Language Understanding (NLU) unit, which provides information about the user's intent, entities, and other relevant information. Based on this input, the DM decides on the appropriate response to provide to the user. Normally DM system consists of two part Dialogue State tracking, which tracks the state of the dialogue flow and Dialogue policy, which decides which response should convey to user. Overall DM system is the controller of the conversation flow of a CA.

1.6.5 Text to Speech (TTS) Unit

Text-to-speech unit converts text input to voice or speech output. This unit consists of a machine learning model which can convert text input in any language and returns speech in that respective language. It makes the conversational agent fully automatic and user don't need to touch or watch the response himself. User can hear the response in voice because of TTS unit.

1.7 Challenges

As we move forward to create a solution for the aforementioned problem, there are many challenges behind it. A voice controlled multilingual conversational agent can have several components; and for each individual components, there can be numerous challenges. In this section, we discuss the challenges to overcome while building this project.

1.7.1 Combining several system as an end-to-end system

A Conversational Agent is comprised of multiple components customized to specific tasks. Constructing a functional system while ensuring optimal performance of all constituent components is a challenging undertaking.

1.7.2 Bengali as a Low-resource Language

The first challenge any AI engineer can face while working with Bengali language is the availability of good dataset. For any deep learning model, dataset works as a fuel. For the best performance, the dataset needs to be diverse, long and annotated correctly. For our system we need both speech and text dataset for training in Bengali and English. Till early 2022, the only publicly available dataset for Bengali ASR system was the **OpenSLR** dataset. However, the dataset was not diverse enough, as it had data from only 505 persons, most of which are male. The Bengali CommonVoice dataset for speech recognition was released in mid 2022, which promises to have better diversity with almost 20,000 speakers' data [11]. And for text datasets, Amazon Massive dataset [12] helps us in implementing NLU system.

1.7.3 Linguistic Challenges for Bengali Deep Learning Systems

Encoding any kind of text data is a vital part in creating a deep learning model, both for transcription and intent/entity detection. Bengali is a very complex language morphologically.

Grapheme Diversity

One of the most unique features of the Bengali writing system is the use of orthographic syllables as graphemes. In Bengali, there are 168 commonly used grapheme roots, and of them, 80 are consonant conjuncts consisting of 2, 3 or higher consonants. A high number of consonant conjuncts leads to challenges in grapheme to phoneme mapping which depreciates the performance of NLU, ASR and TTS models, as the same conjuncts often have different pronunciations based on the context [11].

Inflection

In linguistics, inflection is the process of forming word variants using a template word based on grammar (e.g., variants based on tense) or sentence semantics (e.g., variants based on the gender of nouns) [13]. Like most other Indic languages, Bengali is highly inflectional. Nouns have over 100 case markers/inflection variations and verbs are found to have more than 40 unique variations. This makes the number of whitespace-tokenized 'words' occur very frequently, on the level of tens of millions compared to 188k words in English. Bengali morphology also allows compound word creation which leads to a high number of Out of Vocabulary (OOV) words. This issue causes challenges in NLU, TTS and ASR modeling, especially for low-resource languages like Bengali.

Diphthong and Triphthong

A diphthong is a combination formed with two vowels in a single syllable. The semi-vowel of the diphthong is placed either on the onset or the coda of the syllable. Thus diphthong is the linguistic summation of vowels plus glide. imilarly, we have triphthongs. Triphthong is a combination of three vowel sounds where the first vowel glides to the second which again glides to the third. In English, there are 5 Tripthongs whereas Bengali has 17.

1.7.4 The Unicode System

Bengali unicode system is not exactly perfect. In fact, different variants of the same character exists in the Bengali unicode system. For example, the no space character, that creates a diphthong or triphthong in the Bengali language has several variants in the unicode system. Also, many characters in the Bengali language have different variants that have "nukta" character attached to another letter with no space character. "Nukta" is a dot like character that can be added to other letters to create new letter. However, it's not expected for all of the variants of those letters to exist in the same system, as a deep learning model can easily get confused by them. Because

of these NLU models have a lot of flaws that prevent them from producing accurate inference findings. The solution is to replace these characters and normalize the input dataset. But such solution for the entire language wasn't available always.

1.7.5 Availability of Training Facilities

Deep learning models are super efficient when properly trained. However, for a super efficient systems like these, super efficient facilities are also required. Training deep learning models require high-end GPUs and bigger size RAMs. In a third world country like ours, such facilities are very expensive and difficult to get.

1.7.6 Designing Dialogue Flows and Responses

The optimal dialogue flow is a crucial element for ensuring a smooth and uninterrupted user experience in the context of a conversational agent. The process of constructing a dialogue flow is a challenging undertaking in the deployment of conversational agents, and the inclusion of multilingual functionality further compounds this difficulty.

For response generation, CA leverages various back-end applications. The process of implementing these responses for CA requires a significant level of programming expertise, as the procedures for their utilization vary across different services.

1.7.7 Application Programming Interface (API) services

For retrieving data from internet conversational system will need help of API services. But all API services don't work properly with system and there is some limitations in usage for these services. Moreover these services are very hard to find in multilingual purposes especially in Bengali.

Chapter 2

Relevant Works

In this chapter, we provide short description about the previous works done on Conversational Agent and it's components.

2.1 Literature Relevant to Conversational Agent

Artificial intelligence (AI) [14] has emerged as an influential technology that is increasingly impacting our daily lives. Conversational agents in particular, also known as virtual agents [15] , have grown in popularity due to their abilities to perform a wide range of task such as customer service [16], information retrieval, e-commerce [17], education [18], and other day-to-day activities. For this large of area of use, there has been a massive amount of work done to enhance the classic conversational agent and chatbot system and makes Human-Computer Interaction (HCI) [19] more comfortable and easy to use.

The first known agent for conversation was ELIZA [20], which a rule-based agent that could response in a natural language. It was developed by *Joseph Weizenbaum* at the *MIT Artificial Intelligence Laboratory* . The idea behind this agent was simple pattern-matching [21][22] using some pre-defined rules, which give it the ability to process user input and search for pattern and response according to the pre-defined rule.

In 1995, ALICE [23] was introduced which was devoloped using simple pattern-matching algorithm and *Artificial Intelligence Markup Language (AIML)* [24] and an improved version of ELIZA. It was the first computer program which could achive most human like response.

By inspired these agents, research was conducted for building more sophisticated and advance virtual agents and chat-bot and tried different kind of algorithms and techniques. In this study [25] author studies an overview of chat-bot technology and their impact and applications of

different kind of chat-bot usage.

In this study [26], author studies language and context based human interaction to find appropriate response. Jabberwacky was designed to replicate normal human conversation in an enjoyable, amusing, and natural way. Jabberwacky was able to process responses based on a dynamic database of thousands of online human interactions made possible by the advent of the Internet. After that an updated version of Jabberwacky was introduced name "Cleverbot" [27] which can response more naturally and passed turing test [28] which a test to decide the machine's ability to exhibit intelligent behaviour equivalent to, or indistinguishable from, that of a human.

In [29], author introduced an agent called *Rea* that can show non-verbal behaviours and interact with user.

In [30][31], author developed a chat-bot that can be used by any University to answer FAQs to curious students in an interactive fashion.

In [17] author analysis the advantage of using virtual agent for e-commerce. According this study, e-commerce will be greatly benefited by using chat-bot or virtual agent for giving common question answer and giving instruction to user for any particular task. Moreover, user are getting service any time as there is no need to interact with provider all the time [16].

Most of them the study was done in English and as technology advances agent on different languages are begun to emerge. But still there a not many work on low ranking languages because of low labeled data resource for training and building an agent.

In [32] author introduced first chat-bot in Arabic Dialect. The author also present the challenges in building chat-bot in language with dialect and low resource. In [33] author build an end-to-end interface for Arabic dialect bot.

In [34], author introduced enhancement method for Vietnamese chat-bot. There are bots in other languages like chinese [35], Japanese [36], Hindi [37] and other popular languages but there are a few work on Bengali language although Bengali language is one of the most used language in the world.

In [38] this study author proposed a chat-bot for Bengali language using BNLP (Bengali Natural Language Processing) and BNLTK (Bengali Natural Language Toolkit). This bot was build for educational usage in question answer fashion. Author propsed machine learning method like search algorithm, Bayesian algorithm for chosing best answer for given question.

All these study was focused on agents that can take text input from user and process these and give proper response according to input. In 2010 Apple introduced *SIRI* as personal assistants for end users. Later in 2012 Google launched Google Assistant and many other personal assistant was introduced like Amazon's Alexa, Microsoft's Cortana etc. These CAs has not only chat-bot

functionality they also have advanced speech recognition and speech analysis capability along with task-oriented capabilities.Though these personal agents have multilingual capability they provide limited usage for low resource language and give low performance languages like Bengali. In our work we proposed an end-to-end system that can give a better performance as personal assistance not only in English but also in Bengali.

2.2 Literature Relevant to Language Identification (LID) and Automatic Speech Recognition (ASR)

The [39] paper proposes classification of speech data. The authors use features extracted from speech to classify emotions of different kinds such as, anger, happy, sadness, surprise etc.

In [40], the authors review different techniques used for speech recognition. This 2010 paper defines the most common speech categories and different approaches.

In [41], the authors focuses on the recent advances in speech recognition techniques. The study reviews different approaches such as, LSTM, transformer, conformer etc. in speech recognition systems.

In [10], the wav2vec2 based XLS-R models are proposed. The models are pre-trained with huge unlabeled speech data and the transformer based network can be fine-tuned for any language with relevant labeled dataset.

The [42] proposes another self-supervised approach to speech representation. The model is light-weight and can also be used for down stream tasks such as, speech classification, transcription etc.

In [43], the author represents and discusses about LID systems and various kind of approach to it. The comparison between the use of phonology, morphology, syntax and parsody are discussed in this paper.

In [42], a new approach for speech representations are proposed using Transformers. The author proposes a use of Self-Supervised learning with masked prediction of hidden units for several speech classification and recognition problems.

In [44], the authors propose a dynamic LID and ASR systems using RNN-T. The proposed system uses a joint LID and ASR system to dynamically identify the language and the transcription for joint detection of English and Hindi.

The literatures helped us gather knowledge about the conventional ASR and LID systems. These helped us decide on our own how to build the system for our use.

2.3 Literature Relevant to Natural Language Understanding (NLU) and Dialogue Management (DM) system

In [45] conducted a survey associated to joint intent detection and slot-filling models in the context of natural language understanding. They also compare the outcomes of this approach with those of individual detection of intent detection and slot filling tasks.

In [46] authors conducted a study on attention method for joint intent detection and slot filling for edge devices. They proposed a method called Fast Attention Network that optimize both latency and accuracy.

In [47] authors proposed a novel architecture for joint intent detection and slot filling task using a non-autoregressive model

In [48] authors conducted study that examines the impact of transfer learning on multilingual tasks aimed at training low-resource languages through the utilization of cross-lingual pre-trained embeddings.

In [49] the authors conducted a review of various approaches that are currently available for dialogue management in conversational systems. They also provided guidance for selecting an appropriate dialogue management approach that is aligned with the user's interests.

In [50] authors developed a response selection models using transformer architecture for dialogue system and showed that is possible to pre-train these architectures on a comprehensive and varied dataset, and subsequently fine-tune them for task-oriented dialogue within particular domains.

In [51] authors also used transformers to encode dialogue context and proposed two novel approach for dialogue system

In [52] authors presented a architecture called Recurrent Embedding Dialogue Policy (REDP) architecture which used recurrent neural network architecture to encode dialogue history.

In [53] authors proposed a hybrid methodology for dialogue state tracking and have demonstrated that the amalgamation of techniques such as slot tracking and candidate generation yields superior outcomes for dialogue tracking systems.

In [54] researchers conducted a study on the existing frameworks for conversational agents and analyzed their respective strengths and weaknesses.

Upon conducting a thorough review of relevant literature, we determine to utilize joint intent detection and slot-filling techniques for our NLU unit as this approach allows us to accomplish both tasks through the utilization of a single model, which is advantageous for speeding up response

times. Furthermore, the RASA framework was chosen as the most suitable option for deploying our Conversational agent due to its modular architecture, which facilitates experimentation with different components of our system and allows for the selection of the most effective approach for our system.

Chapter 3

Dataset & System Architecture

3.1 Dataset

In this section, we provide a brief overview of the datasets which have been used to train and prepare the models of our architecture. Our proposed system has several deep learning based models which require good datasets for training and evaluation purpose.

3.1.1 Datasets for Automatic Speech Recognition

The Automatic Speech Recognition (ASR) model for our proposed solution has two subsections in it. One part is used to transcribe Bengali speech signal to text while the other is used to transcribe English speech signal. For the different languages, we use two different models, both of which require a unique dataset. For Bengali language, we use the Bengali Common Voice dataset [11]. For English language, we use English Common Voice dataset [55].

Bengali Common Voice Dataset

The Bengali Common Voice dataset is a dataset containing 400 hours of Bengali spoken data from all over Bangladesh and India. The dataset is collected on *Mozilla Common Voice Platform* and have been a joint effort of *Bengali.Ai*. The dataset is publicly available and is more flexible in diversity and size compared to previously available datasets. The dataset contains a CSV file with transcriptions of all speech data including the age, gender, accent etc. information and the path to the related audio data. The audio data are in mp3 and small in length (5-9 second on average), making them better suitable for ASR model training. The CSV file contained some more information, such as whether a particular transcription is correct or not. The up-vote, down-vote columns represent the accuracy of the said transcription.

English Common Voice Dataset

Compared to other languages, dataset for English is easy to find. The Common Voice Dataset for English, prepared by *Mozilla Common Voice Platform* is by far the most diverse dataset publicly available in English. The dataset contains approximately 9,000 hours of speech data from over 42,000 unique speakers. The data covers a wide range of accents, ages, and genders, making it a diverse enough for training ASR models. The English dataset, just like the Bengali one, also contains a CSV file with almost the same information as Bengali. The audio data are also in mp3 and small in size, making the pre-processing steps of the both datasets almost similar.

3.1.2 Datasets for Language Identification

Detection of language from spoken data is not as easy as it sounds. The Bengali Common Voice dataset for ASR is new and very few dataset exists for Bengali language classification. So, the language identification model, which was able to detect the languge from spoken data, is trained on a custom dataset created from the Bengali Common Voice and English Common Voice dataset used for ASR training and evaluation. The classification dataset was created simply by assigning each ASR dataset with a language and then merging them. For diversity purpose, the resultant dataset was shuffled several times. The final dataset contained a CSV file containing the path to the audio data and a language. The audio data was a mixture of both Bengali and English Common Voice Dataset. To avoid any kind of biasness, the total number of data from each language was kept almost equal by eliminating excess data from English corpus randomly.

3.1.3 Datasets for Language Model and Dialogue Management

A huge amount of data for training to build a robust conversational agent which will can understand user command and give proper response of the command. Moreover as our system has to have multilingual capability we needed a dataset not only with large number of corpus but also they are properly labeled for multiple language especially Bangla and English. For this reason we chose to use *Amazon MASSIVE* dataset [12] which is composed of one million realistic, human-created virtual assistant labeled utterances spanning 51 languages 60 intents, 55 slot types, and 18 domains. This dataset was created by tasking professional translators to localize the English-only SLURP [56] dataset into 50 typologically diverse languages from 29 genera. This dataset was released in two steps and we used data from first release which contains around 11.5k utterance for each language. We only took Bengali and English utterances for training our system.

3.2 System Architecture

We proposed to build an end-to-end conversational system. Our system consists of five major components described in . Our overall system are shown graphically in figure . Here we can see that, our system will take voice input from user, pass it to ASR unit which will converts the input to text format. Important information in the command will be extracted by NLU unit of our system and return structured information for our system. DM unit will use these information

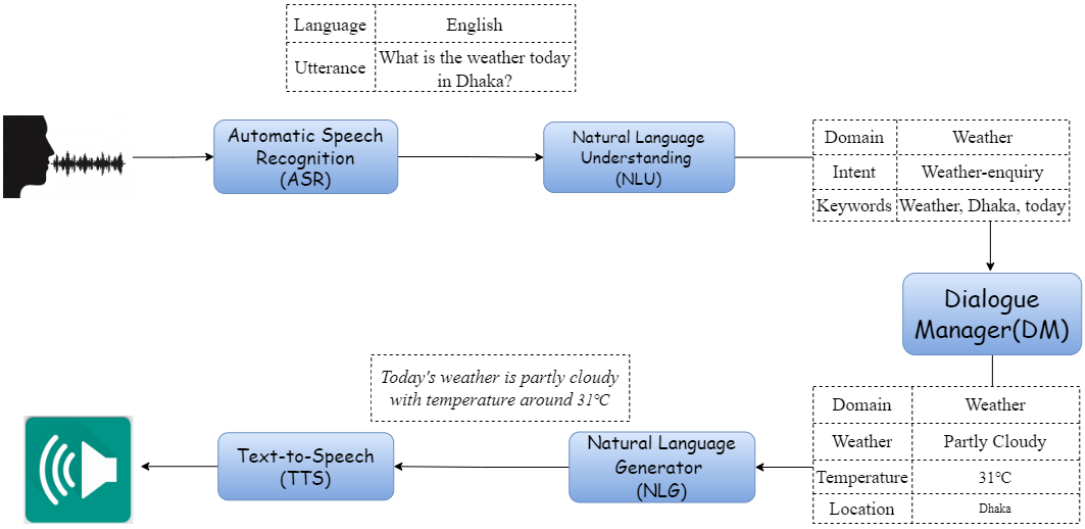


Figure 3.1: End-to-End System Architecture of our proposed system

and trigger proper action for generating response for user like calling API, retrieving information from internet etc. NLG unit will convert these data to human understandable information and pass to TTS unit which will give desired response in voice for user convenience.

3.3 Models

In this section, we discuss about the models we use for our desired solution. As our system consists of different units and most of them are deep-learning based system, we used different models for different parts of the system. Each of the models have been trained on the datasets mentioned above.

For ASR acoustic models, we use *Facebook's XLS-R* model [10], which is a transformer based self-supervised model. For Language Identification, we use *HuBERT* model [42] which is also a sel-supervised model with masked prediction of hidden units. The models are integrated in a way to create a working pipeline that can identify language and use the corresponding language vocabulary to transcribe to text. The transcribed text from the acoustic model is further used as an input to the *KenLM language model* [57] which better predicts the transcription of the given audio data.

3.3.1 Facebook's XLS-R Model

Facebook's XLS-R model is a large-scale, multilingual language model based on the wav2vec 2.0 [10, 9]. The model was trained on a massive amount of unlabeled audio data across 128 languages, including languages with low resource. The model is capable of performing well on a wide range of downstream tasks, such as classification, speech transcription, speech translation etc. The pretrained model is mainly trained on a huge amount of unlabeled audio data with upto 2 billion parameters. This huge amount of pretrained unlabeled data opens the door to fine tune the model with small amount of data, that can outperform all previous state-of-the-art models.

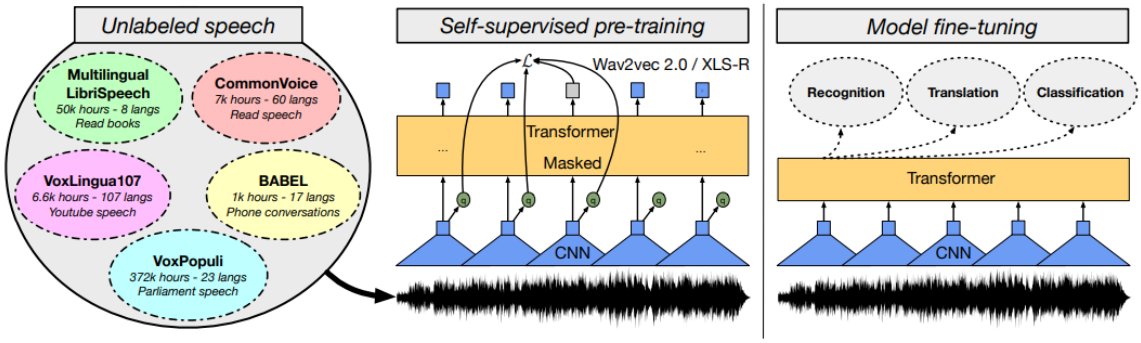


Figure 3.2: Block Diagram of Facebook's XLS-R Model.

The figure 3.1 shows a working pipeline of the XLS-R model for speech representation. The model is pre-trained on a large multilingual wav2vec 2.0 Transformer (XLS-R) on 436K hours of un-annotated speech data in 128 languages. The training data is from different public speech corpora and the model can be fine-tuned for several multilingual speech tasks. This model is used to create two separate acoustic models for both Bengali and English language. The XLS-R model provides three versions of the pretrained models, starting from models with 300 million parameters to upto 2 billion parameters.

- Wav2Vec2-XLS-R-300M
- Wav2Vec2-XLS-R-1B
- Wav2Vec2-XLS-R-2B

Due to limitations in computing facilities, we've used the 300 million parameter version of XLS-R model, which have been enough for a satisfactory result in our use case.

3.3.2 KenLM Language Model

The KenLM language model [57] is widely used for building and querying language models for various use. It uses a method known as N-gram language models for minimizing time and space cost. This method uses *Hash Tables* and *Probing* to efficiently find associated probabilities and penalties. This can distinguish between two phonemes that sounds almost the same in audio data and determine which is the correct transcription. A simple example can better illustrate the use case of this language model. Imagine a speaker says "To do better in exam, you have to read the book". In this particular sentence, an acoustic model can detect *rid* instead of *read*, as they sound almost exactly the same. This can reduce the overall performance of the ASR model and can result in wrong intent detection in several cases. The language model, which is already trained on a datset of huge language data, can use probabilities of all possible sentences and detect the most probable sentence that was intended by the speaker.

3.3.3 HuBERT Model for Speech Representation

HuBERT (Hidden-Unit BERT) is also a transformer based self-supervised model for speech representation [42]. But instead of using huge unlabeled data for speech representation, it uses a technique known as masked prediction. HuBERT model benefits from an offline clustering step to generate noisy labels for a BERT-like per-training. Concretely, a BERT model consumes masked continuous speech features to predict predetermined cluster assignments. The predictive loss is only applied over the masked regions, forcing the model to learn good high-level representations of unmasked inputs to infer the targets of masked ones correctly. Intuitively, the HuBERT model is forced to learn both acoustic and language models from continuous inputs. First, the model needs to model unmasked inputs into meaningful continuous latent representations, which maps to the classical acoustic modeling problem. Second, to reduce the prediction error, the model needs to capture the long-range temporal relations between learned representations. One crucial insight motivating this work is the importance of consistency of the targets, not just their correctness, which enables the model to focus on modeling the sequential structure of input data.

We mainly use HuBERT model for Language Identification (LID) for it's light-weight. The model is not as heavy as the previous XLS-R model and can be trained to classify speech data in two languages, Bengali and English. Since we're using only two languages for our initial design, the model can be trained for binary classification problem while the classes are nothing but the two languages we need to detect.

3.4 RASA Framework for NLU and Dialogue Mangement

RASA Framework

RASA[58] is an Open Source Language Understanding and Dialogue Management framework that provides a flexible and extensible toolset for building and experimenting conversational Ai. We used rasa as our main conversation handling platform. There are several existing framework other than RASA for implementing conversational agent such as **Dialogflow** [59], **Amazon Lex** [60], **Microsoft Bot Framework**[61], **Pandorabots**, **Chatterbot** [62] etc. We chose to use RASA framework as Rasa is an open source frame work and it supports a huge number of languages which helps us to build our multilingual conversational agent. Moreover Rasa can be deloyed easily in various platform like Telegram, Slack, Messenger, Whatsapp and other social services [63]. Rasa framework also support conversation driven development which enables rasa system to learn more from users and using those insights to improve performance.
RASA pipeline flows graph components where more than one module of the system can be processed parallelly rather sequential pipeline where one module executed at a time and advance one-by-one.

Graph Components Method

Graph components method is one of the method for completing a system work where all the components of the system are connecting graphically according their dependence to each other.

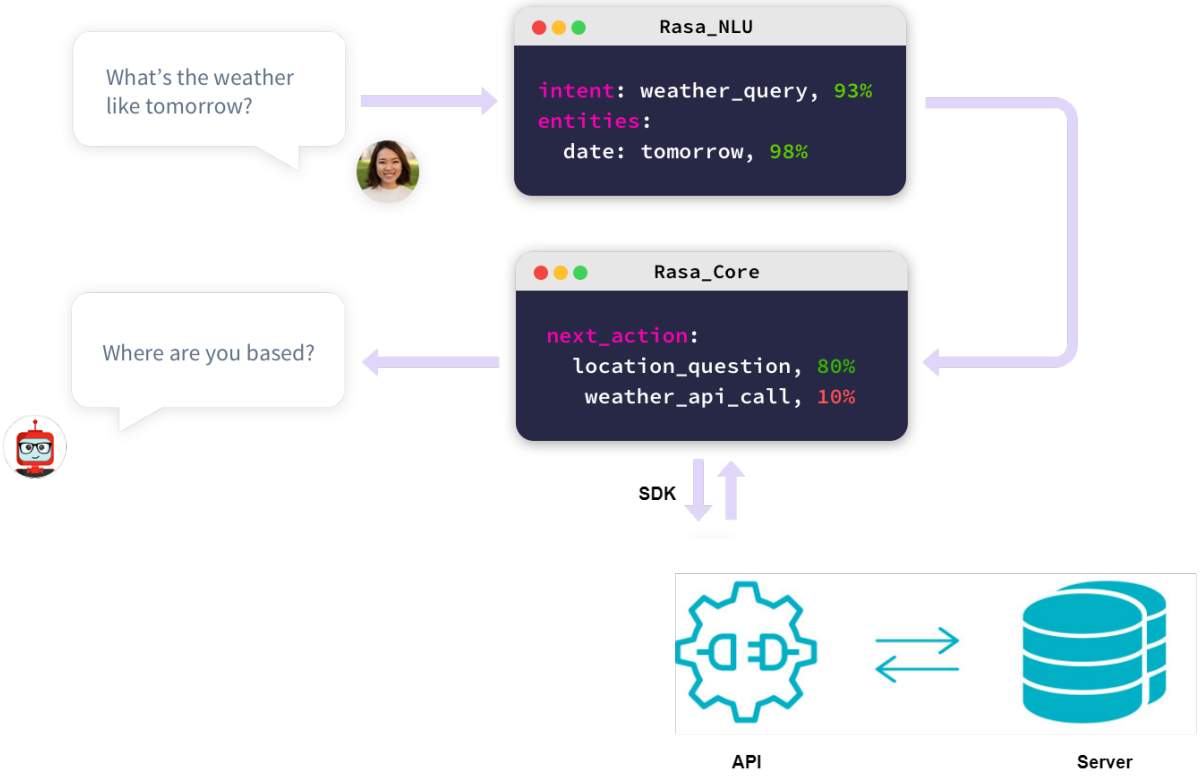


Figure 3.3: RASA Working pipeline

This method is more compatible than the sequential execution method. Graph component optimize computational graph for execution of a model. They can be helpful to represent different model architectures flexibly. Figure 3.4 shows graph component execution method of RASA method. From the figure we can see that more than one components can be executed at a time in graph method which is not possible in sequential execution method.

RASA system architecture consists of three main parts:

- Rasa NLU (Natural Language Understanding)
- Rasa Core (Dialogue Management)
- Rasa SDK (software development Kit)

3.4.1 Rasa NLU

Rasa NLU is the natural language understanding module. This component handles the workflow for understanding the user's input and extracting relevant structured information from user messages. Rasa NLU uses different machine learning algorithms to classify intents and extract entities from user messages. Rasa NLU produces a structured data representation of the user's command, which the CA may utilize to build a proper response.

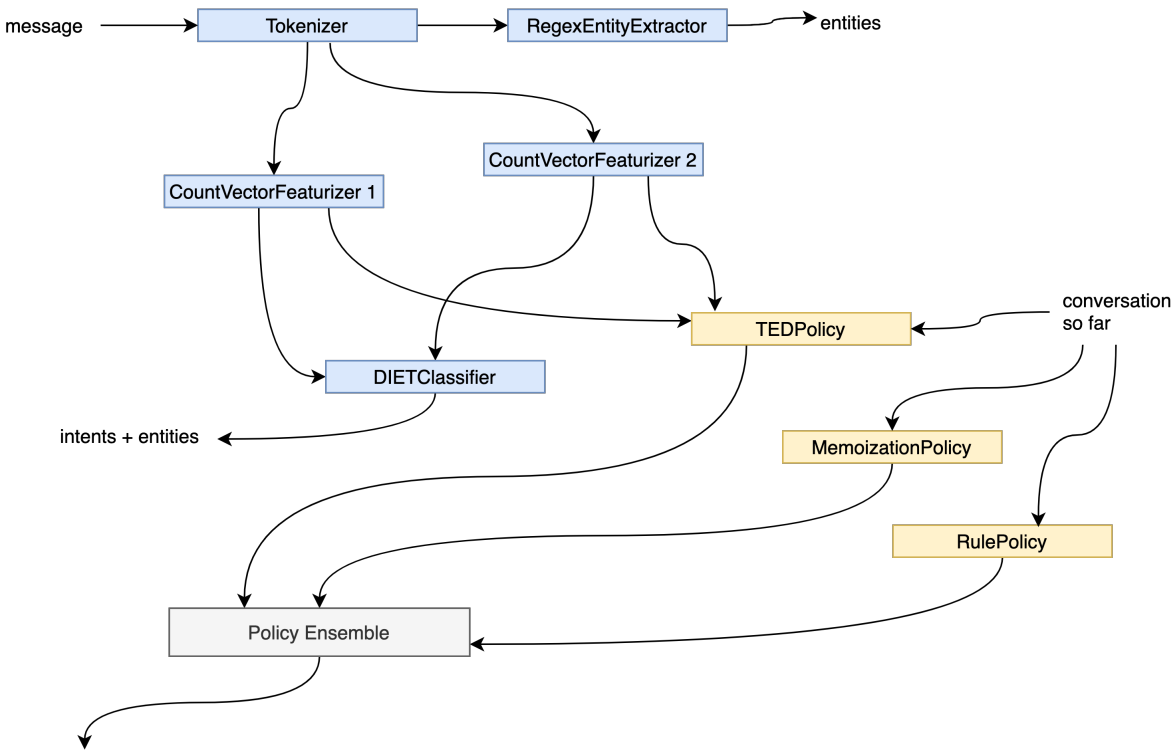


Figure 3.4: Graph Architecture method of RASA Pipeline

The NLU component consists of two parts: intent classification and entity extraction. Intents are the classes of given command and entities are keywords of the command. For example, if given command is "What is the weather today in Dhaka?", The intent of this command is weather query and keywords are today, and dhaka which will be extracted as entities. Intent classification identifies the user's intent behind a particular input. The system can be trained to recognize an arbitrary number of intents. Entity extraction, on the other hand, identifies specific pieces of information within the input, such as dates, locations, or names. For our system, we implemented a joint intent-slot approach, which uses a single language model to detect intent and classify the input command. A slot filling mechanism is utilised to convert the extracted data to a structured format which will subsequently be used in the RASA core for dialogue propagation.

DIET for Language Understanding

Dual Intent and Entity Transformer (DIET) [64] is a new multi-task architecture for intent classification and entity recognition. This architecture is modular which enables it to incorporate different pre-trained word embedding from language models and combine these with sparse word and character level features in a plug-and-play fashion.

DIET architecture consists of several components. Key parts among them are-

- Featurizer
- Identifier

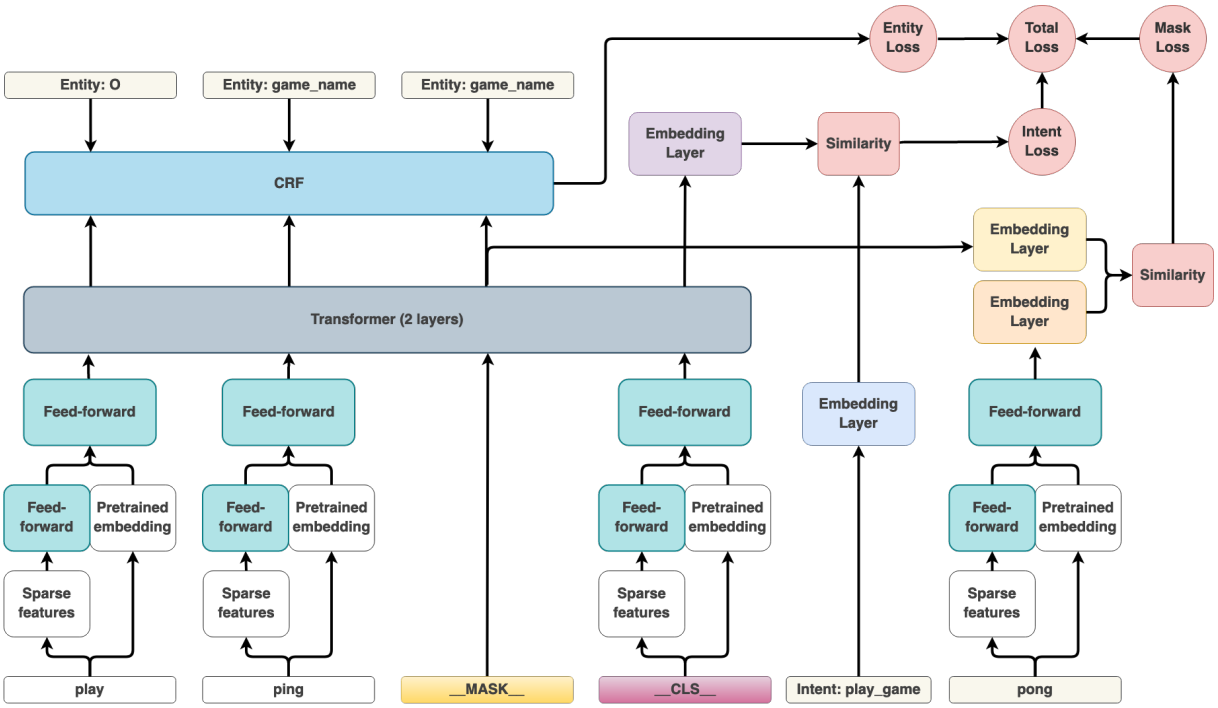


Figure 3.5: DIET Architecture

Featurizer module of DIET architecture takes text input and converts it to a sequence of tokens and then each token is featurized with sparse features and dense features.Sparse features are token level one-hot encoding and multi-hot encodings of character n-grams. And for dense features, any pre-trained word embedding such as ConveRT [65], GloVe [66], BERT [67], GPT-2 [68], mBERT [69] etc. These embedding can be used individually or multiple embedding can be used for more accurate analysis. Generated sparse features and dense features combined together and then pass to a transformer [70] head which process the features using attention mechanism and returns representation for intent identification and entity recognition. Intent and entity labels are predicted through a Conditional Random Field (CRF) [71] tagging layer on top of the transformer output sequence.

For our multilingual capable system we used Language-agnostic BERT Sentence Embedding (LaBSE) [72] as our main embedding unit. This a Bidirectional Encoder Representations from Transformers (BERT) [67] based model which has cross-lingual capability and because of its agnostic behaviour it can handle multiple language input with lower number of parameter in comparison of other multilingual models. BERT sentence embedding (LaBSE) model supporting 109 languages. Because of that LaBSE can provide large language coverage for research and also can give a satisfactory result for low resource languages in NLP research.
We also tried some other language model such as multilingual BERT (mBERT), Generative Pretrained Transformer 2 (GPT-2), DistilBERT etc as featurizer to evaluate our system. Results are explained in Results section.
We used DIET model as our building block for NLU system with different featurizer and

transformer head which give us an excellent result for identifying intent and entity recognition which is discussed in later chapter.

3.4.2 RASA core for Dialogue Management

Rasa Core is a fundamental component of the Rasa platform that manages the conversation flow and determines the appropriate action for the CA to take in response to user input. Rasa Core uses machine learning techniques to predict the next optimal action based on the current conversation state and the user's input to do predict the next response. DM system maintains a state tracker, which tracks the current state of the conversation and is updated with each user message. The state tracker helps the system make informed decisions about what action to take next and which pieces of information to request from the user. The system trains a policy network, which is a type of neural network, using data sets of previous conversations to predict the best action to take.

Rasa core takes structured information as input which was formed by Rasa NLU module and use these information like intent, entity etc and previous conversation state from tracker to predict the next action. For these rasa uses various policies such as:

- Transformer Embedding Dialogue (TED) Policy
- Memorization Policy
- Rule Policy
- Fallback Policy
- UnexpecTED Intent Policy etc

Transformer Embedding Dialogue (TED) Policy

Transformer Embedding Dialogue (TED)[73] policy uses transformer architecture for modeling multi-turn conversation that capture the underlying structure of the dialogue, rather than just memorizing specific patterns in the training data. TED policy architecture consists of several transformer encoders which are shared for both tasks. For the next action prediction, the dialogue transformer encoder output and the system action labels are embedded into a single semantic vector space which is the used to compared to all possible system actions and the one with the highest similarity is selected for next action. TED has the ability to dynamically adjust the importance of different parts of the dialogue history during the prediction of the next action because of self-attention mechanism that allows the model to selectively attend to relevant parts of the dialogue history while ignoring irrelevant information. This enables the model to make more accurate predictions, particularly in situations where the context of the conversation is complex and the user's intent is unclear. We used TED policy as our main action selection policy for long and unseen conversation in our system.

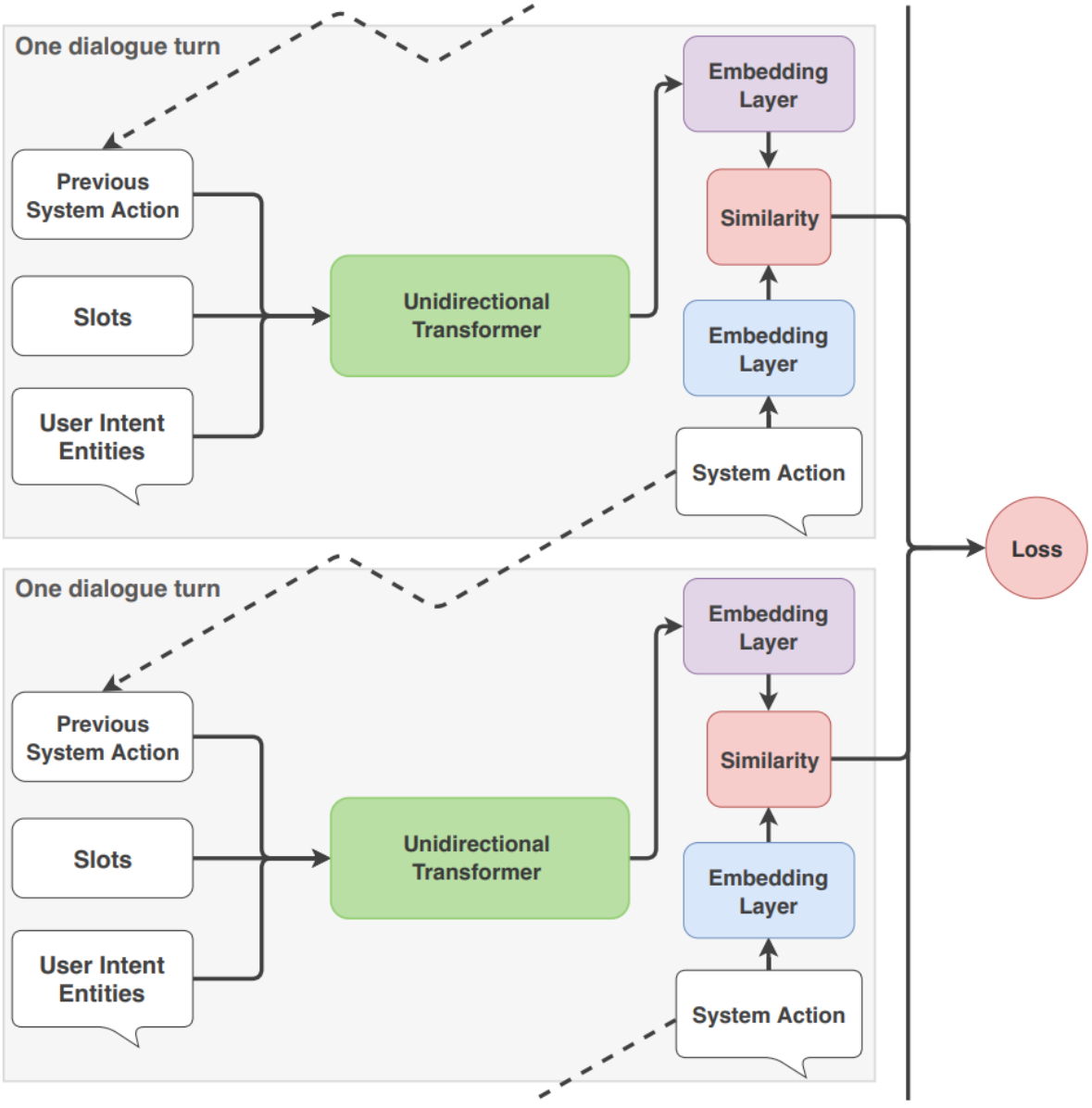


Figure 3.6: A schematic representation of two time steps of the transformer embedding dialogue (TED) policy

Memorization Policy

The Memorization policy is designed to memorize the dialogue history and predict the next action based on previous user inputs and the corresponding actions taken by the CA. This policy remembers common story pattern of dialogue flow of training dialogue and checks the matching story for current conversation and predict next action from the matching story.The Memorization policy works by constructing a lookup table that contains all prior user inputs as well as the accompanying agent actions. During inference, the policy simply searches the lookup table for the most recent user input and returns the matching action. If no exact match is found, the policy fallback to another policy set in the Rasa Core configuration.
The Memorization policy has the advantage of providing very fast and precise responses to previously witnessed user inputs, which is especially useful when the dialogue history is short and the user's intent is clear. However, it is unsuitable for managing complex conversation or situations with a long or confusing conversational history.

Rule Policy

The Rule policy is a policy that handles conversation parts that follow a fixed behavior or rules. It makes predictions based on any rules that was defined during training. These rules are essentially a set of conditions that must be met in order for a particular action to be taken.
The Rule policy is beneficial when the conversation flow is predictable and short. That's why we configured our rule policy for short and always fixed such as trigger a greet action if user greets etc.

Fallback Policy

The Fallback policy is used to handle situations where the chatbot is unable to understand the user's input or is unable to determine the appropriate response based on the current conversation state. This policy is triggered when Rasa core returns a low confidence score on a NLU prediction or by a low confidence score on an action prediction by another policy.In such situations, the Fallback policy trigger a default action that should be taken by the agent, such as providing a fallback message or asking the user to rephrase their input. This policy helps to improve user experience by providing a default action for unexpected command that the Rasa NLU coludn't process properly.

UnexpecTED Intent Policy

UnexpecTED Intent Policy helps to review conversations and also allows agent to react to unlikely user turns.

UnexpecTED Intent Policy has the same model architecture as TED Policy. The difference is at a task level. Instead of learning the best action to be triggered next, UnexpecTED Intent Policy learns the set of intents that are most likely to be expressed by the user given the conversation context from training stories. It uses the learned information at inference time by checking if the predicted intent by NLU is the most likely intent. If the intent predicted by NLU is indeed likely to occur given the conversation context, UnexpecTED Intent Policy does not trigger any action. We used UnexpecTED Intent Policy on top of TED [73] policy which helps TED policy to surface these unique conversation paths from past conversations.

3.4.3 Rasa SDK (software development Kit)

Rasa SDK is a Python software development tool which provides capability for running custom of a conversational agent. While RASA NLU and Core decides which response to give for a command, Rasa SDK produces actual response and sends to user. Rasa SDK consists of a set of pre-built components that are be used to extend the agents capabilities, such as custom actions, connectors to external APIs, and integration with third-party services. Using these component conversation flow and customise response are designed and agent response according to these defined actions. We used this module to defined our custom actions. All are actions are custom action as we have to response accordingly the command's language and call third party APIs if needed. We used a few third party APIs for fetching data from internet like open weather API, foursquare API etc.

Chapter 4

Processing & Experimentation

The conversational agent we offer uses several deep learning models, APIs and other mathematical models. In this chapter, we describe the training process of our models and related data pre-processing. The training process can be divided into several of subsections, i) Data Pre-processing, ii) Feature extraction & Training.

4.1 Data Pre-processing

Each Deep Learning network requires different kind of data processing for training and evaluation. The required input of a particular deep learning network depends on the model used. Regarding this, all of the data has be processed and prepared for training, so that the neural network can understand the data.

4.1.1 Data Pre-processing for ASR

The ASR dataset, as discussed before, consists of several useful information. The CSV file contains the path to the audio file, the transcription of it and other information such as, accent, age, gender etc. for diversity. For our particular task, we only need two of these information, the audio clip and the transcription of it. We also used the *up votes* and *down votes* information of the dataset to filter data with probable incorrect transcriptions. Thus filtering can ensure the model is not fed with wrong information.
The pre-processing part mainly includes two kind of processing. i) Audio Processing & ii) Text Processing. While the audio processing is exactly the same for both Bengali & English language data, the text processing is totally different due to different vocabularies. The pre-processing techniques are described in the next subsections.

Audio Processing

All of the datasets with audio are usually available in **.mp3** format due it's compression and size. However, our deep learning model uses raw speech data for learning method and can not use mp3 file for training. Thus, the files were converted from **.mp3** to **.wav** file. The process was done by using the **ffmpeg** python library. For faster conversion of the whole data, parallel processing algorithms were used. Due to limitation in computational device, we were able to only convert 40,000 speech data at a time. However, since English is a high resource language, the 40,000 speech data proved to be enough to be used for training purpose. The Bengali dataset was converted to **wav** for all of the data and after filtering with up votes and down votes, almost 140,000 speech data were available for training.

Text Processing for English Language

The text processing of English transcriptions are comparatively easier as English, as a language, is linguistically simple. The unicode transcription of English language is perfect and there's never been an English character with similar unicode representation. Thus, the text does not need normalization. It was further confirmed by extracting vocabulary from the English transcription and examining all of the unique characters present in the dataset.
The transcriptions always contain punctuation of different kinds. The punctuation in English language has no acoustic phoneme while speaking. So, for an acoustic model that scans utterance of human speaker, punctuation are of no use at all. So, the dataset was cleaned by removing all the punctuation from it.
The space character in English language was replaced by '|' character. As a final touch, two characters 's' and '/s' were added. The final vocabulary of characters for the English dataset stands as below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
|	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
p	q	r	s	t	u	v	w	x	y	z	<s>	</s>	[PAD]	[UNK]	

Table 4.1: Vocabulary for English ASR model

The vocabulary contains 31 characters, 27 of them are from the alphabet and the rest are special characters. The characters are all in lower hand so that the model is not confused by 'a' and 'A', which have the same utterance.

Text Processing for Bengali Language

The Bengali language is a lot more complex than English linguistically. The use of grapheme diversity, inflection, aspiration and germination results in a complex language pattern that can prove to be challenging while transcribing speech to text. Also, the unicode representation of the Bengali language is not perfect. Several characters in the Bengali language contain multiple unicode representations. In [74], the authors discuss about the possible problems faced in Bengali

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
|	</s>	<s>	ঁ	ং	ঃ	অ	আ	ই	ঈ	উ	ঊ	ঋ	এ	ঐ	ও	ঔ	ক
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
ন	প	ফ	ব	ভ	ম	য	র	ল	শ	ষ	স	হ	া	ি	ী	ু	ূ
54	55	56	57	58	59	60	61	62	63	64	65	66	67				
ৃ	ে	ৈ	ো	ৌ	্	ৎ	ড়	ঢ়	য়	\u200d	[UNK]	[PAD]					

Figure 4.1: Vocabulary for Bengali Speech Recognition Model

text. The common problems faced regarding this case are broken diacritics, nukta problems, invalid hosonto, unwanted doubles etc. A normalization tool for processing the text data to machine readable, simple form was developed by the authors of [74]. The normalization tool known as *bnUnicodeNormalizer* can normalize all of these aforementioned problems, all at once. This normalization tool also offers to selectively normalize each particular problem or leave some of the problems that are not relevant. As an example, the nukta problem can be treated by either deleting all the letters previously combined by nukta and defining them by any letter + nukta; or it can delete the nukta sign from the whole vocabulary and define the letters with dotted nukta as an entirely new and unique character. This gives the user a flexibility for using a process best for the model. The Bengali dataset containing transcription was thus normalized by using this normalization tool.

The latter part of text processing of Bengali language remains the same as English. All of the punctuations were removed from the vocabulary, space character was replaced by '|'. As there's no upper or lower case character in the Bengali language, there was no possibility of the model getting confused due to that. The resulting vocabulary of the Bengali language is shown.

The vocabulary, as can be seen, contains a total of 67 unique characters. The ASR model for Bengali language was trained on this vocabulary to perfectly predict the transcription of given audio input.

4.1.2 Data Pre-processing for Language Identification (LID)

The data pre-processing for Language Identification is comparatively simpler than the ASR models. As mentioned, the dataset was custom created using Bengali and English common voice dataset by classifying each of the audio file with the language. As the audio was already converted to .wav file for ASR models, the audio pre-processing was not needed. However, classification problems in deep learning requires the classifying column to be encoded to each class. As our LID system is a basically a binary classification model, we assigned each language to a class labelel. The label was 0 for Bengali language and 1 for English. The class encoding was done by using python function *Class_ Encode_ Column*. No further pre-processing was required.

4.1.3 Data Pre-processing for KenLM Language Model

For training the n-gram language model, we used the Amazon MASSIVE dataset. As our NLU and Dialogue Management uses the same dataset, and the target use of the overall system is to work as an virtual assistant, it was important that the user command is recognized without any error. The Amazon MASSIVE dataset was annotated to detect intents and entities. However, for the language model to predict the correct transcription, such intent and entitites are not required. So, we extracted only the text file of the actual command from the whole dataset, deleted everything else in order to train the language model. The result was an array of user commands, transcribed in English and Bengali. Afterwards, both of the dataset needed to be normalized. The unicode normalizer function mentioned previously was used to normalize all problematic representations of each of the characters available in the dataset.
One of the problem faced for an ASR system is to correctly transcribe name of places it has never heard before. As the English dataset was not focused on use in Bangladesh, the acoustic model was exposed to errors in transcribing districts or places in the country. This would be very much problematic since detecting the name of the place, in other words the entity, is a vital requirement for the NLU system. So, we used a python loop to simply replace any name of a city in the dataset with all 64 districts of Bangladesh. Such as, 64 copies of the sentence "Book me a flight to London" was made, each containing a city of Bangladesh. The resulted sentences would be "Book me a flight to Dhaka", "Book me a flight to Sylhet" etc. The same procedure was done with Bengali dataset as well.
The final arrays of the both datasets were converted to two text files, each containing all the sentences of each language. KenLM uses the text file to further train and build n-gram Language Models.

4.1.4 Data Pre-processing For NLU and Dialogue Management system

For training our CA we used *Amazon MASSIVE* dataset as discussed in **Chapter-2**. We used only English and Bengali command from the dataset as our CA can communicate with user both English and Bengali. This dataset is composed of realistic, human-created virtual assistant utterance spanning 60 intents and 44 slot types. We build our system with 18 intents that's why we filtered those data that align our task from MASSIVE dataset discarded the remaining. Though this dataset was created from human command for virtual agent, the dataset was created mostly for US context like- places with US address was frequently present in the dataset, news search commands was also related to US context and affirs. As we building the CA for especially in Bangladesh and other Bengali Countries, we have to change these context and add relevant context and circumstances with Bagladesh and her surroundings. Thats why we added more augmented data from this dataset, adding relevant context so that our system can response for both Bangladesh context and US and other English Language using context.

We implemented our system wiht Rasa platform which demand intent and slots annotation a defined format [75] which is different from what we get in Massive dataset. That's why we have

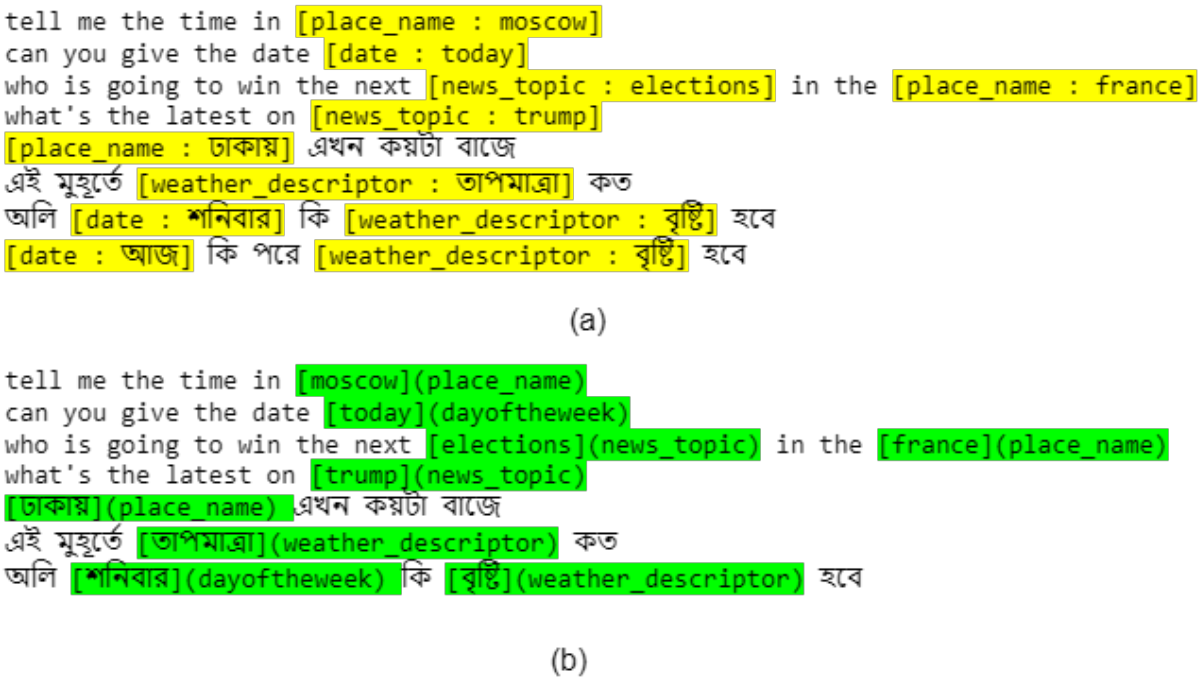


Figure 4.2: Annotation Conversion for RASA NLU. (a) Annotation in Amazon Massive Dataset (b) Annotation Converted to RASA standard

to change the annotation style for preparing our data for training.

Though Massive dataset was created by expert translators there were some mislabeled data in the dataset like some words were not annotated though they were in some slots category. First we sorted all data into category using intent label and save them into separate files. Then we check for if they belong this category and if they are not we discarded them. Then we added some data created from the reaming data using some augmentation. That way we get our dataset ready for train our NLU module with a moderate ammount of data which can give us a staifactory results for our system.

We also done some pre-processing for Bengali language as Bengali language has a large number of letters in alphabet normal *Unicode* encoding gives a runtime error in python so we used *utf-8* encoding to solve this problem. We also normalized our Bengali text before training which is described in previously.

```
- story: ask about time
  steps:
  - intent: start
  - action: utter_greet
  - intent: inquire_time
  - action: action_tell_time
  - intent: inquire_time
    entities:
    - place_name: "dhaka"
    - dayoftheweek: "today"
  - action: action_tell_time
  - intent: inquire_news
  - action: action_tell_news
  - intent: inquire_date
  - action: action_tell_date
  - intent: inquire_news
  - action: action_tell_news
  - intent: goodbye
  - action: action_utter_goodbye
```

Figure 4.3: Example Story for Dialogue Management system

4.1.5 Data processing for Dialogue Management

Our dialogue management is designed on Neural network model which predict what will be the next action for a continuing conversation. That's why we need to train our dialogue management system. For dialogue management system we have to configure 2 types of data-

- Stories
- Rules

Stories

For our dialogue management system we need dialogues for training our assistant. In Rasa stories [76] are the method for defining dialogue data for training. A story is a representation of a conversation between a user and agent. Stories are formatted in a specific format where user inputs are expressed as intents with necessary entities, and the agent responses and actions are expressed as action names. We used wide range of story for every task for our system, so that it can handle wide range of conversation with.

```yaml
- rule: tell time
  steps:
  - intent: inquire_time
  - action: action_tell_time

- rule: Say goodbye if user says goodbye
  steps:
  - intent: goodbye
  - action: action_utter_goodbye
```

Figure 4.4: Example Rules for Dialogue Management system

Rules

Rules [77] are another kind of data for training our system which includes some short pieces of conversations that should always follow the same path. Our conversational agent will follow these rules all the time if the condition for this rule is met. Rules are great for handling short and predictable conversation.Rules describe. Unlike stories rules don't generalize to unseen conversation paths. Defining rules in rasa also follow some pattern like stories. We used some rules for our system which is confirmed to happened during conversation like response a greet word when user greets etc.

4.2 Feature Extraction & Training

In this subsection, the feature extraction and training process of our system is described. As mentioned earlier our system consists of several mechine learning model for working properly as a conversational agent.

4.2.1 Feature Extraction for ASR and Models

The ASR system was built on wav2vec 2.0 architecture. The following diagram shows the working pipeline of an ASR system. The feature extraction uses CNN to extract features from raw speech data.

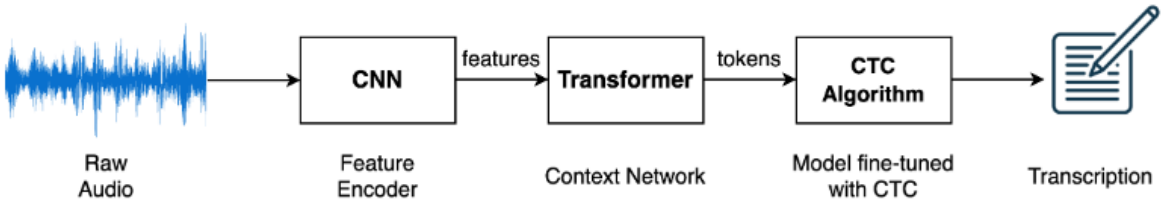


Figure: Working pipeline for wav2vec 2.0 architecture

For building the feature extractor for, the **Wav2Vec2FeatureExtractor** function was used. This function was built in the HuggingFace transformer class. The parameters for this feature extractors were pulled from the pre-trained model, wav2vec2-XLS-R-300M, which was described in previous chapter. For training the model, a tokenizer was also needed. The tokenizer assigned token for each characters in the transcription using the vocabularies for each language. As for tokenizer, the **Wav2Vec2Tokenizer** was used from the HuggingFace transformer class. Combining the tokenizer and feature extractor, the Wav2Vec2 Processor was created, which further processed the dataset to make it ready for training the models. The LID system had no particular processor or tokenizer, as there was no transcribed text. The feature extractor was enough to encode the speech signal into features. The class encoding of the language information was enough for the LID training purpose.

4.2.2 Training for the ASR & LID systems

As shown in the previous figure, after the CNN feature encoder, comes the transformer network. The transformer network is the pre-trained model used for fine tuning. The model is trained on a huge dataset of unlabeled audio. Typically, the transformer network can be any kind of network. For this particular task, the wav2vec2-XLSR-300M model is a self supervised model.
The fine tuning of the model uses a labeled dataset uning CTC to improve accuracy and build a end-to-end acoustic model for speech recognition. The transformer network described before was fine tuned using the processed and labeled dataset of English and Bengali separately.
The language identification (LID) system on the other hand, was created by using HuBERT model, as explained before. The pre-trained **hubert-base-ls960** was used for this purpose. Obviously, the XLS-R model used for the ASR network could also be used here. But this was a design choice as HuBERT model is light weight and faster in performance than the previous one.

Training Device

The fine tuning was done on a desktop with following configuration.

- Processor: AMD Ryzen 7 5800X 8-Core Processor
- RAM: 16 GB
- Graphics Processing Unit (GPU): Nvidia RTX 3060 12GB

The training was done on the GPU for parrallel computing with fp16 feature.

Evaluation Metric

The evaluation metric used for training ASR systems is known as **Word Error Rate (WER)**. WER measures the difference between the recognized words generated by the ASR system and the actual words spoken in an audio signal. The lower the WER, the better the ASR system is at recognizing speech accurately.
The LID system, on the other hand, simply used "accuracy" as the evaluation metric. This metric simply compared the class *(0 for Bengali, 1 for English)* of the test dataset with calculated accuracy of the system.

Training Parameters

The training parameters were tuned and tweaked for the optimum performance by trial and error method. The following parameters were used for training purpose.

Parameters	English ASR	Bengali ASR	LID System
per_device_train_batch_size	16	16	16
gradient_accumulation_steps	2	2	2
num_train_epochs	40	45	5
fp16	True	True	True
weight_decay	None	None	0.01
learning_rate	2e-4	1e-8	None
lr_scheduler_type	None	"cosine"	None
evaluation_matric	WER	WER	Accuracy

Table 4.2: Training Parameters for ASR & LID models

The learning rate scheduler was used to dynamically change the learning rate as the model evaluation metric reached optimum value. For English ASR model, no scheduler was needed to get a decent enough result. For Bengali ASR, a "cosine" based scheduler proved to be good enough. There was no learning rate used in the LID training system, rather a weight decay was used.

4.2.3 Training and Building of the n-gram Language Model

As mentioned previously, the language model works on top of the acoustic model to predict the transcription better using the probability of a sentence being correct. For building our language models, we used KenLM system. The dataset was prepared as discussed before as a text file containing numerous sentences for each of the languages. The n-gram models were created using simple commands.

An n-gram is a sequence of n words, where n is an integer greater than zero. The simplest form of an n-gram model is a unigram model, where each word in a sequence is considered independently. The bigram model, which is a type of 2-gram, considers the probability of a word given the preceding word, and the trigram model, a type of 3-gram, considers the probability of a word given the two preceding words.

To calculate the probability of a given sequence of words, an n-gram model assigns a probability to each n-gram in the sequence and multiplies them together. For example, to calculate the probability of the sentence "The cat sat on the mat," a bigram model would consider the probability of the first word "The" on its own, the probability of the bigram "cat The," the probability of the bigram "sat cat," and so on. The probabilities are usually estimated from a large corpus of text using maximum likelihood estimation or smoothing techniques.

The language models used in our ASR systems were 6-gram language models, meaning each sequence of words were followed by 5 other words. KenLM was used to create an *.arpa* file containing all of the probabilities of sequences. After building the *.arpa* file, decoders had to be created for the end-to-end ASR systems using *pyctcdecoder*, which is a decoder algorithm that

uses the output from both the acoustic model and the language model for a better prediction of the transcription. The language model improved the overall ASR system performance by a decent margin.

4.2.4 Experiment for NLU feature extraction

Our system use DIET architecture for NLU feature extraction and intent, entity classification.DIET architecture has two main section as mentioned in 3 3.5. We experimented with different multilingual feature extractors for extracting features from text. For feature extraction we first trained our architecture with low weight featurizer like **LexicalSyntacticFeaturizer** and **CountVectorsFeaturizer**. These model was trained with our processed training data 3. As these are low weight models which is already pre-trained, we only fine tuned for our working purpose. These featurizer works fine with English language but having error in extracting Bengali entities like place name and news topics etc.

After that we tried out some language model which support multilingual training. We first trained **mBert** model on our dataset. mBert model is already trained on huge dataset from Wikipedia text with over 100 language. For our task we fine tuned the model using our dataset.
We used rasa build in tokenizer for tokenizing our text input and then use these token sequence to train our system.. We trained our configuration 300 epoch with 64 batch size and learning rate $1 * 10^{-3}$. **ReduceLROnPlatea** is used for scheduling our learning rate. With mBert Model we got satisfactory results which is better than previous configuration.

For experimenting with parameters we next tried **m-distilBert** model which is also a Bert based model but having lower number of parameter which may help us to trade off time constraint. We follow the same configuration for training parameter as used in mBert training.
Next we experimented with BERT with **LaBSE** weights which is a language agnostic model which gives it multilingual capability for featurization.

As all these language model are pretrained with large dataset we only fine tuned these model for our working purpose. We followed **cross-validation** method for deciding our models stability over wide range of conversation text. We run 3 experiment 5 fold cross-validation with each model configuration. Results of these experiments are given in results section **??**.

Our main experiment was done for transformer head which classified the intents and entities of our system. We used different number of transformer head as classification head with different size for classification. Our base experiment was 2 transformer head with transformer size = [128 256]. We incremented number of transformer by 2 and transformer size is used either of the two 128 or 256. Our experiment shows that with higher transformer head we have slightly better classification results but it takes much higher time during prediction. So, for trade off with time constriant we decide to chose 2 transformer head with transformer size = 128.

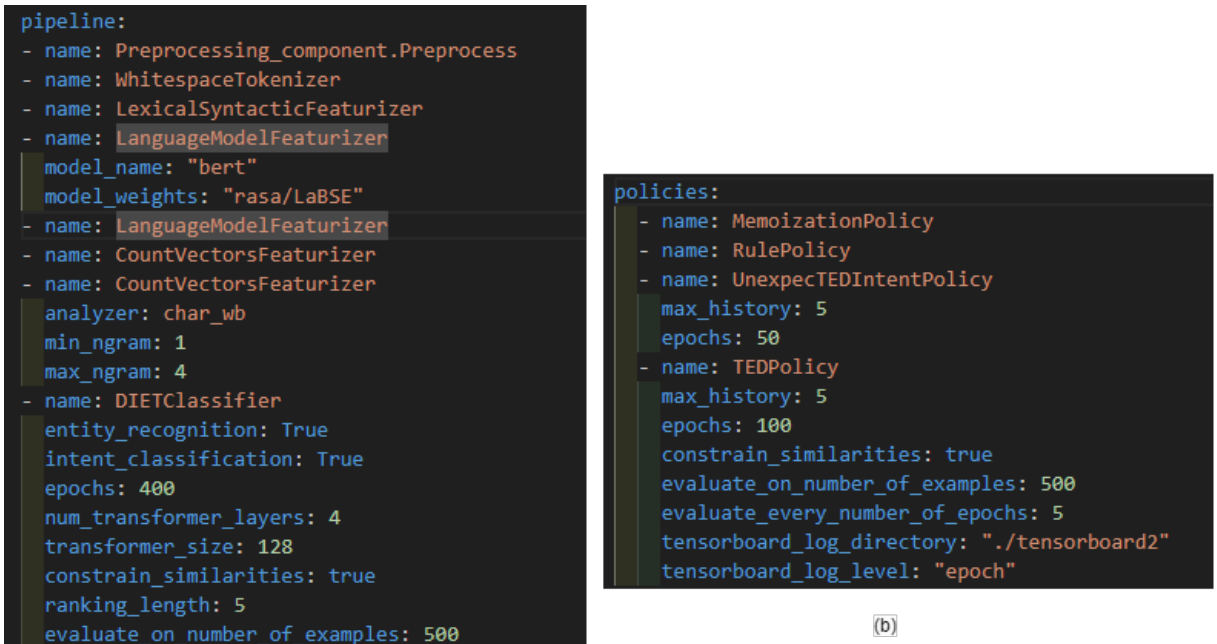


Figure 4.5: Training Configuration for RASA framework (a) Pipeline configuration for NLU. (b) Configuration for Dialogue Management

For Dialogue management training we trained TED policy and UnexpecTED Intent Policy. For understanding underline conversational flow and training TED policy we use augmentation on our story data. We converted our story data into many small conversation block using augmentation and use different combination for increasing our training data.We used augmentation== 150 which is our each story was sub-divided into 150 sub-stories. As a result our TED policy can generalize a wide range of conversational paths. For training we used same configuration as mentioned earlier.

Training NLU unit and DM unit was an important part for our Conversational agent. Proper training is the key for better working of our system. Thats why we tried different architecture and analyse their advantages and errors which is explained in results section

Chapter 5

Results

In this section we presented our experiment results for different units. First we discussed about different evaluations metrics that we used for evaluating our system and then we presented our results and findings.

5.1 Evaluation Metrics

5.1.1 Accuracy

Accuracy calculate how many instances are accurately predicted among all the predicted instances. Accuracy is the most common metric uses for model evaluation. It is the ratio of the number of correct predictions to the total number of predictions made by the model.The accuracy score measures how effectively the model predicts the proper outcome or class. Higher accuracy means better model performance.

5.1.2 Word Error Rate (WER)

Word error rate is a common metric in deep learning to measures the percentage of words in the recognized output that differ from the reference or ground truth transcription. A lower WER is generally considered indicative of higher accuracy for an ASR system.
The calculation of WER involves counting the number of substitutions (S), insertions (I), and deletions (D) required to transform the recognized output into the reference transcription. These operations represent the errors made by the ASR system. The WER can be calculated using the formula:

$$WER = \frac{S+I+D}{N}$$

5.1.3 F1 score

F1 score is an important metrics for evaluating classification task. F1 score is a measure of the model's accuracy and balance between precision and recall. Precision is the model's ability to properly identify positive classes among those it predicts as positive. In contrast, recall assesses the model's ability to properly identify all positive events among all actual positive instances [78]. The F1 score combines precision and recall into a single metric by calculating their harmonic mean [79]. It provides a balanced measure that considers both precision and recall, giving equal importance to both. A high F1 score indicates a good balance between precision and recall, meaning that the model performs well in both identifying positive instances correctly and avoiding false positives.

$$\text{F1 score} = 2 \times \frac{precision \times recall}{precision + recall}$$

5.1.4 Confusion Matrix

confusion matrix visualizes and summarizes the performance of a classification model. Confusion matrix also known as error matrix. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class [80]. By evaluating confusion matrix, model performance can be visualized and in which class the model is struggling to predicted correctly can also be understand by confusion matrix. So, confusion matrix is a common practice in evaluating and error analysis in model performance.

5.2 Performance Evaluation of different units of our system

In this section we will present our systems performance and performance of each units of our system.

5.2.1 Performance Evaluation for LID & ASR Units

The ASR and LID systems were evaluated using the metrics **WER** and **accuracy.** The ASR model, fine tuned on the XLS-R model of the Facebook, achieved a score good enough for our particular application. The results are shown in the table 5.1.

Model Name	**During Training**	**Test Dataset**		**Validation Dataset**	
		Without LM	With LM	Without LM	With LM
English ASR	5.46%	7.1%	5.3%	7.9%	5.9%
Bengali ASR	13.6%	14%	15%	13%	12.8%

Table 5.1: WER for the multilingual ASR models

Here in table 5.1, the WER achieved during training as well as the WER evaluated on the test and validation datasets are shown. During training, only the acoustic model is used. However, for the test and validation datasets, we integrated the language model with the acoustic model as well. The evaluation before and after integrating the language model is shown here as well.
As we can see, the English ASR model has a WER of about 5-7%, which is good for any ASR system. The language model significantly improves the overall performance of the English ASR system. However, as the Common Voice dataset does rarely contain the names of places in Bangladesh, it's a vital job for the language model to work it's way while detecting it. Also, the overall Bangladeshi accent does not work as good. But the overall system output was good enough.

AS for the Bengali ASR system, the WER is much higher than English. This is because the pre-trained model has not been exposed to the Bengali dataset as much as the English one. However, the result achieved here is almost 13-15%, which, in practice, was convenient. The Bengali accent is almost always transcribed perfectly while training and no issues were faced.
The LID system used the HuBERT model for fine tuning into a binary classification model. The 'accuracy' metric used here is convenient for classification models. The generated scores shown in the table 5.2

Model Name	**During Training**	**Test Dataset**	**Validation Dataset**
LID	99.9%	99.93%	99.87%

Table 5.2: Accuracy for Language Identification model

As can be seen from the table 5.2, the accuracy is never lower than 99%, which is outstanding, because a wrong detection of the language can be catastrophic while transcribing and generating response from the NLU unit.

5.2.2 Performance Evaluation for NLU Unit

We evaluated our system on various metrics. For our NLU components we calculated *f1 score* and *model accuracy* for each of the models. We also analyse their output performance as an end-to-end system using confusion matrix.

For our NLU models we used Accuracy and f1_score as main evaluation scores. Our experiment evaluation scores are given in Table 5.3.

In Table 5.3 we can see the results of our experimentation on NLU featurizers. As mentioned in 4 we used different language models as featurizer. From our experimentation we find out that BERT with LaBSE intialization gives better result than mBERT. On the other hand distill-mBERT

gives also very good results for our system. Moreover as it has much less parameters than other two it is a good trade off between accuracy and time required for results. comparison between these models are given in Figure 5.1.

Table 5.3: Result comparison of different NLU models for Intent Classification and Entity Extraction

Model Name	**Intent**		**Entity**	
	Accuracy	f1_score	Accuracy	f1_score
BERT(LaBSE)	96.0	95.9	97.	90.7
mBERT	90.8	90.3	94.7	81.1
distill-mBERT	94.7	94.6	96.4	88.1

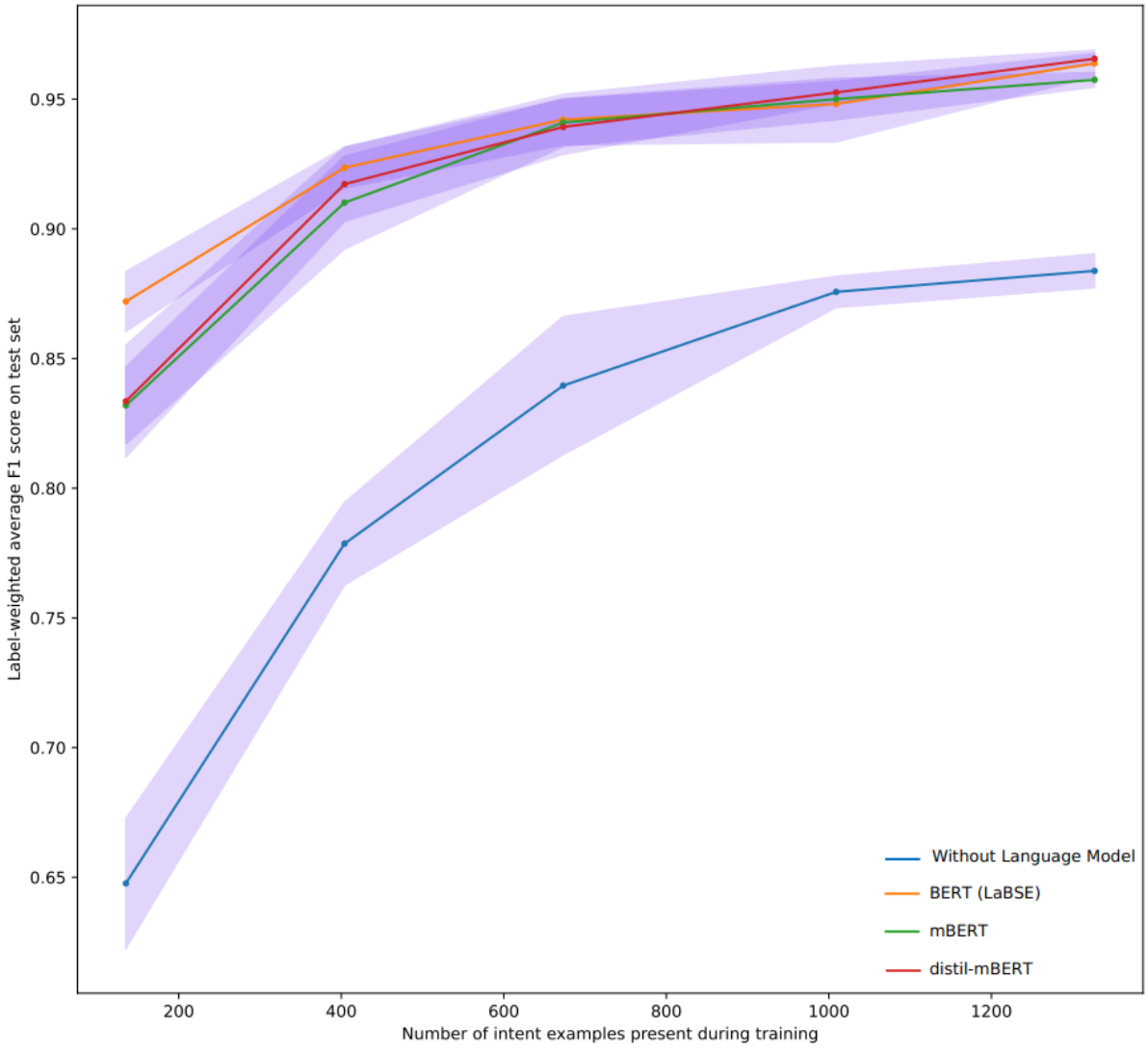


Figure 5.1: Comparison of different NLU pipelines in RASA

We also did some error analysis on our results and using confusion matrix of our trained matrix. Some of the confusion matrix of our analysis are given below. From the confusion matrix we find that main reason of low results our entity extraction scores of our model are the name entities and some places in Bengali with compound letters. They make some confusion during prediction. We trained with more classification head for better extraction of these entities.

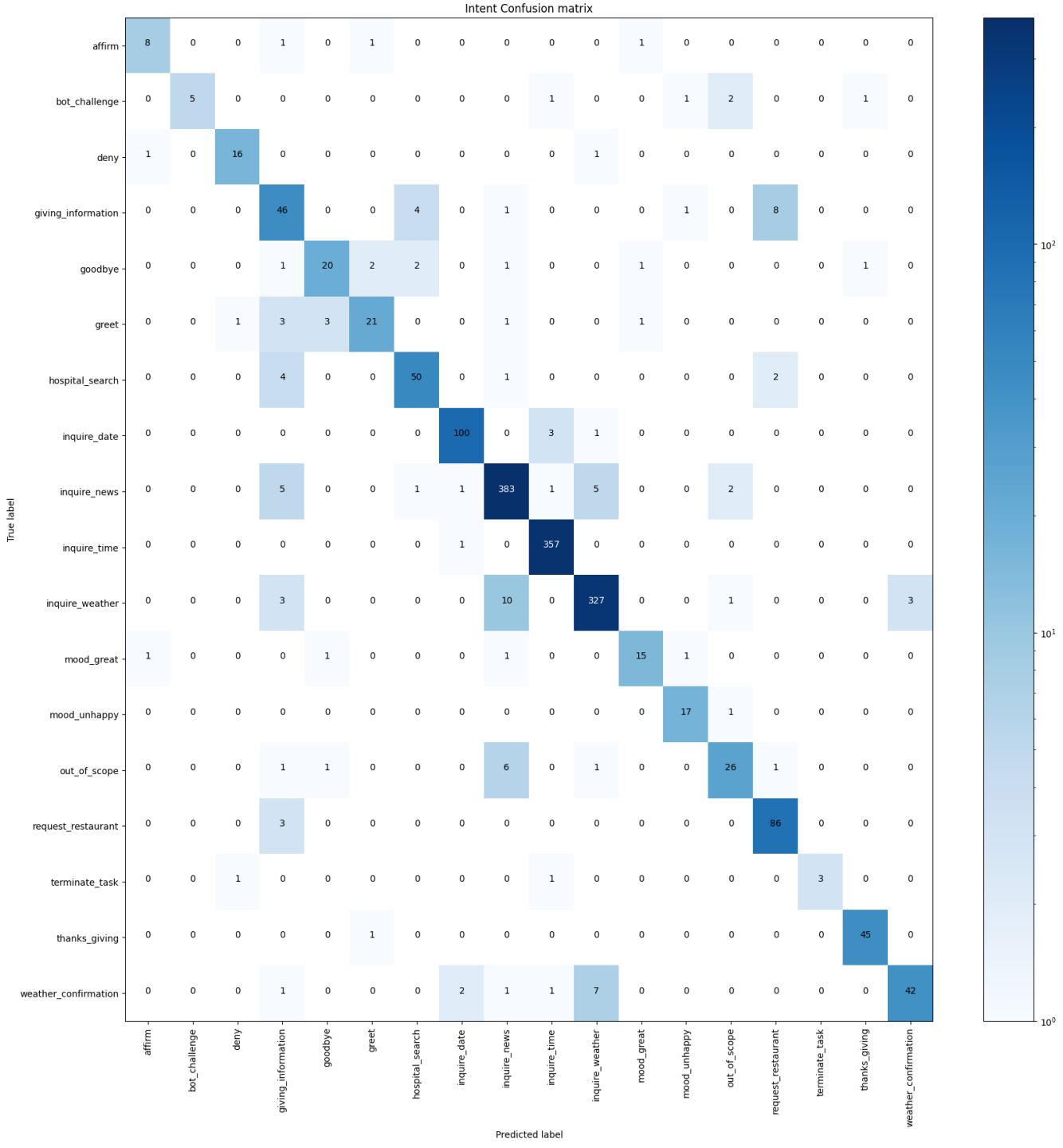


Figure 5.2: Intent classification confusion matrix of BERT(LaBSE) Model

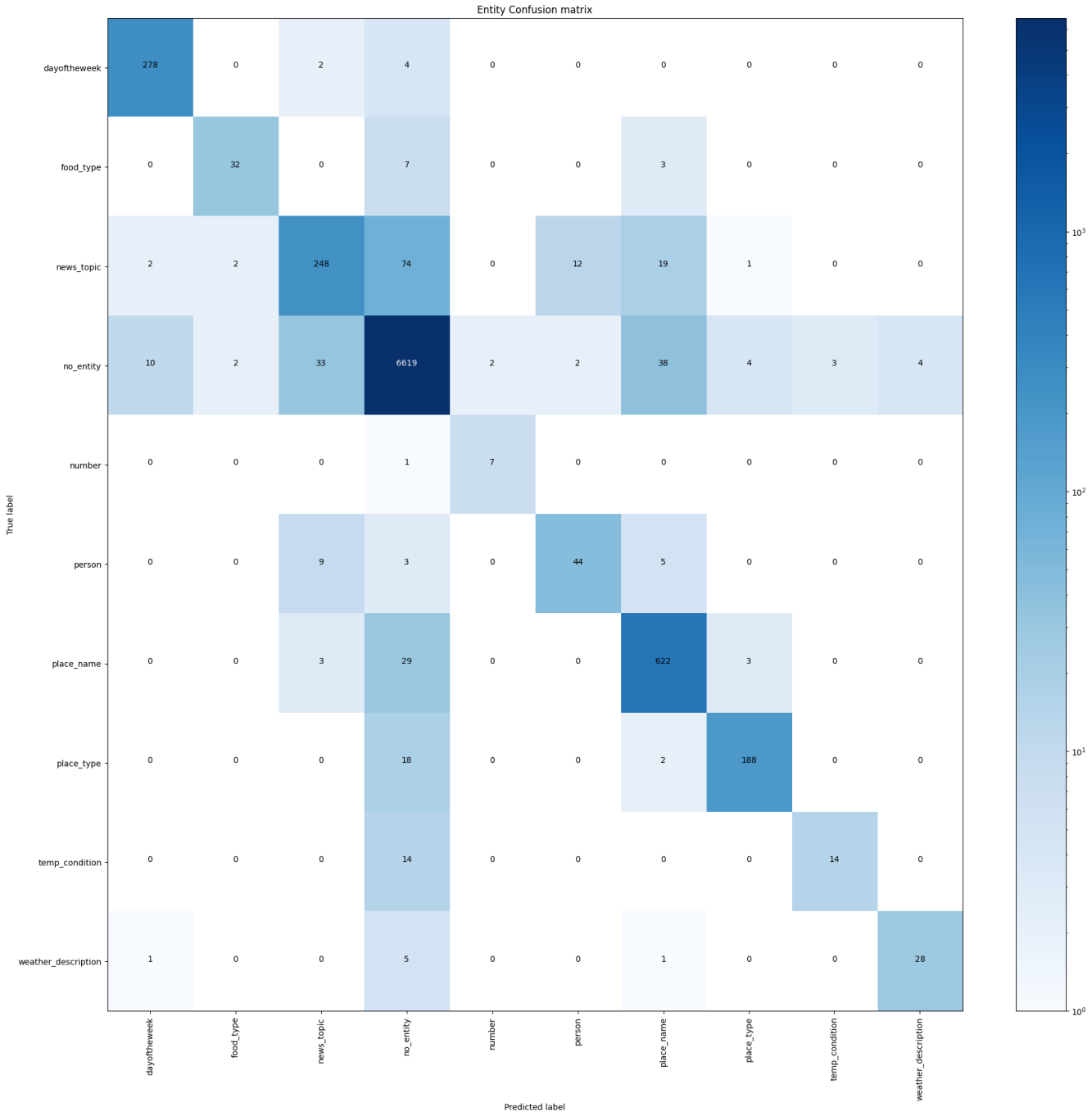


Figure 5.3: Intent classification confusion matrix of BERT(LaBSE) Model

5.2.3 Performance Evaluation for DM unit

For DM unit we train a deep learning model called TED as mention in 4. As we used huge augmented stories our system became a robust mode for our system and can almost 100% accurately predict next actions for our conversational agent. Our DM system performance analysis are shown in 5.4.

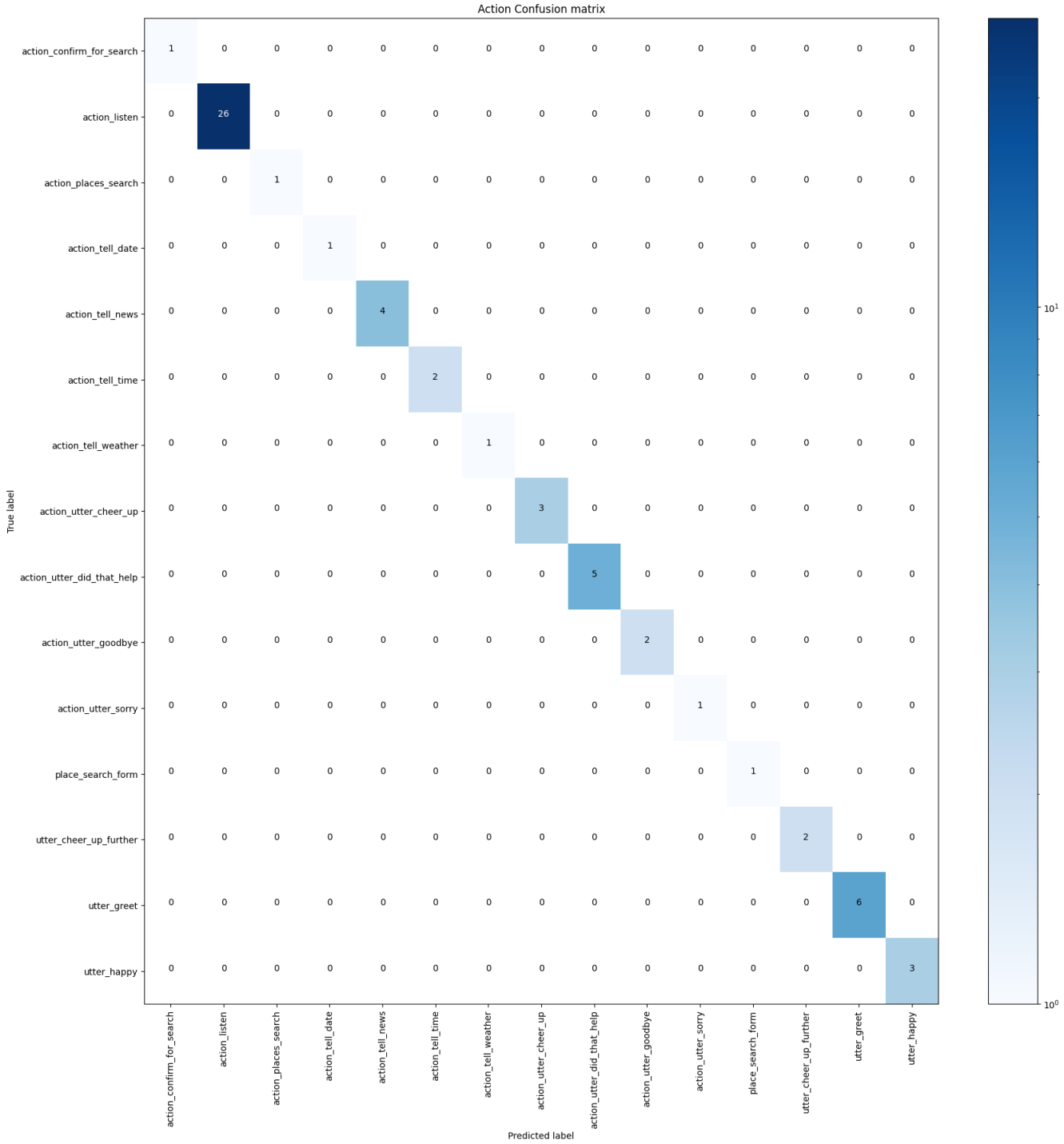


Figure 5.4: Action prediction Confusion Matrix for TED policy evaluation

5.3 Features of our implemented system

Our implemented system is a task oriented conversational agent which can support different day-to-day task automatically by voice command. Here are some of the features our system can support

- Date & Time Query
- Weather Forecast
- News Query
- Restaurant Search
- Hospital & Blood Band & Drug House search etc.

Our system can carry out these type of commands both in English and Bengali. It can handle voice input in both language and returns output in the desired language.

5.4 Results as an end-to-end Conversational Agent

We combine all of our units and build a totally functioning end-to-end system. We also build and Graphical User Interface (GUI) for better representation of our system. We tested our system with real time data and get satisfactory result from user. Here We attached some of the results of our system.

In figure 5.5 we can see the results of our system for a query of restaurant search and in figure 5.6 and figure 5.7 we can see the results for hospital search, weather query, news search respectively. From these results we can say that our system performs very good as a conversational agent.

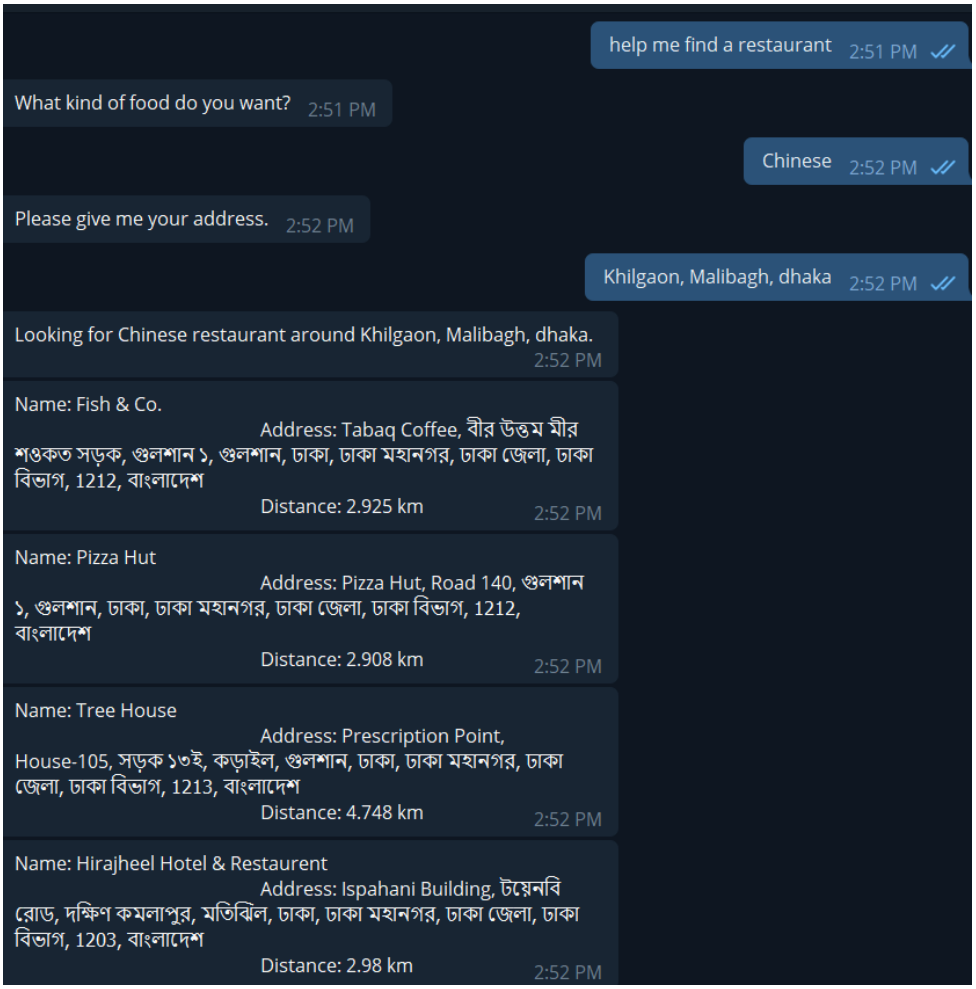


Figure 5.5: Our system result (Showing results for Restaurant search)

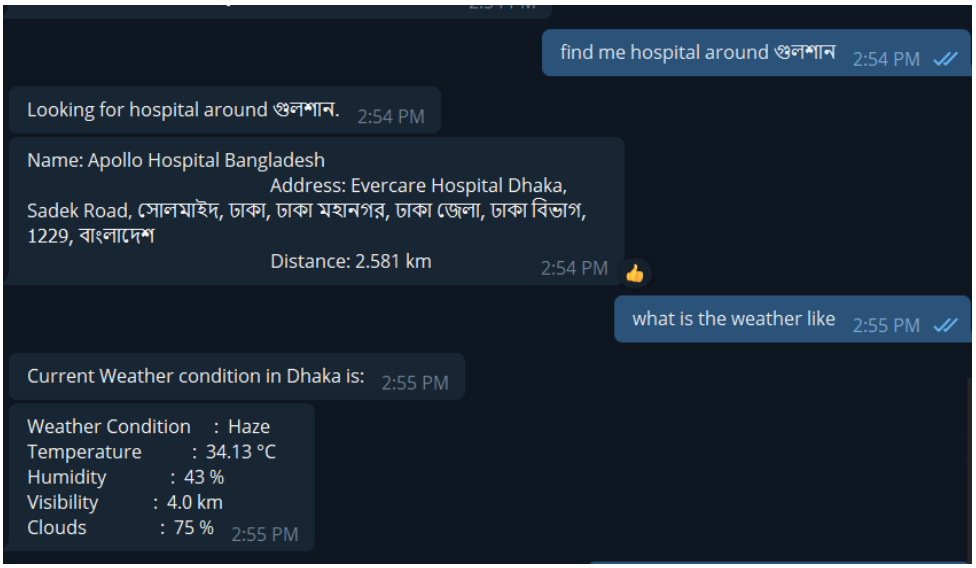


Figure 5.6: Our system result (Showing results for Hospital search and Weather Query)

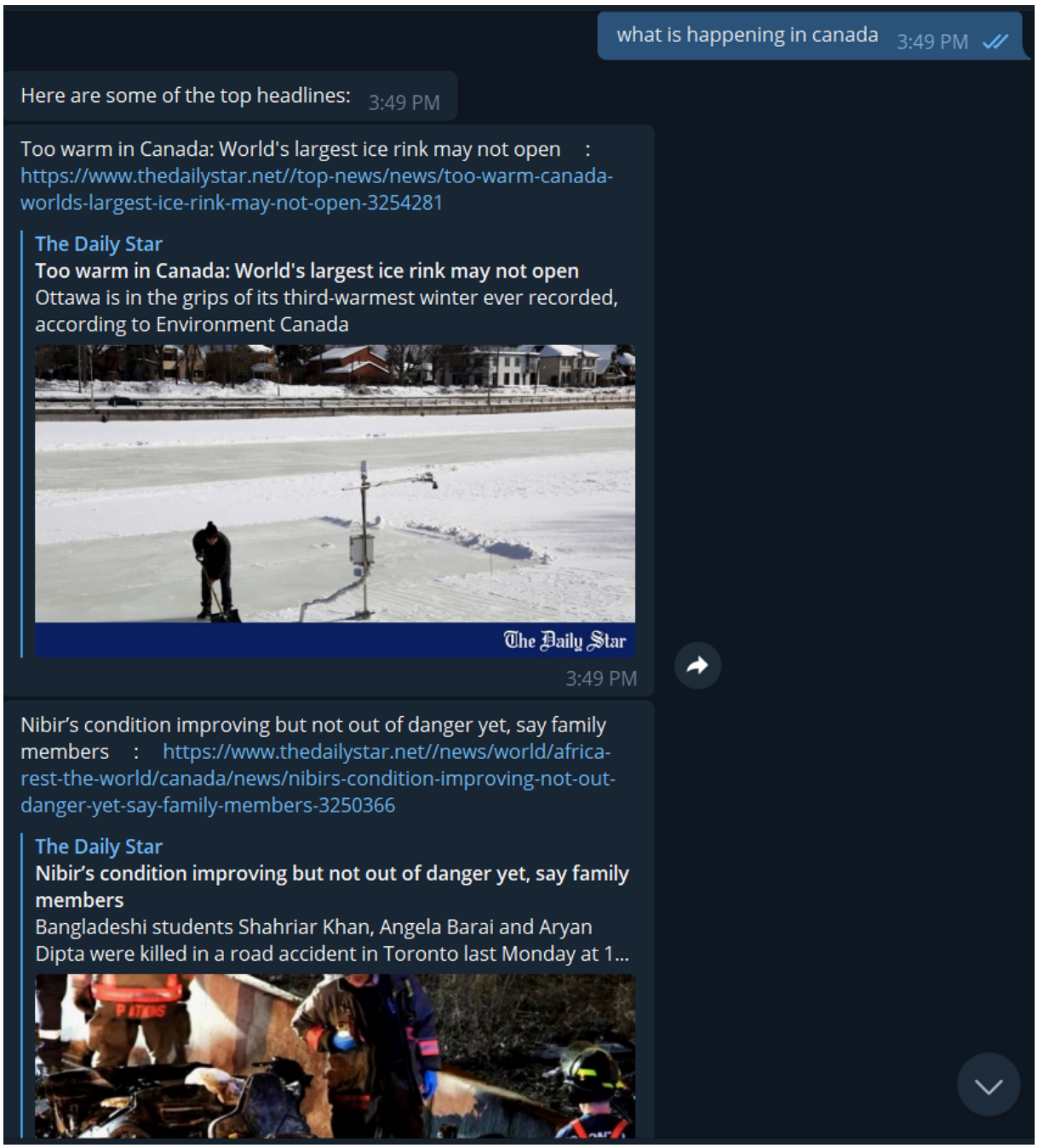


Figure 5.7: Our system results (Showing News search)

Chapter 6

Conclusion

The goal of this research work was to develop a conversational agent which is task oriented and multilingual. First of all, deep-learning models for natural language understanding and automatic speech recognition were developed and experimented with different configuration and analyse the results. By analysing the results with different models for Natural Language Understanding (NLU) and Automatic Speech Recognition (ASR) we selected our base models for these task in our system. As our system is a multilingual system we used a language identification model to distinguish between different language of the given command. We also analyse the system with and without the language identification model. For NLU system we decide to use joint intent and entity recognition approach as this approach give us satisfactory results understanding the command and also need only one model for understanding the command, which helps our system to response quickly. After analysing our machine learning models, we then use dialogue management system to control our system's conversational flow. We used defined rule based method for giving response to user. Various back-end services, such as APIs, are utilised to produce responses by retrieving data from the internet. Following the experimentation of each individual unit, the systems are subsequently integrated into an end-to-end system. The Rasa framework was employed to integrate these discrete components. Combining these units was one of the toughest part of our work, as different units depend on distinct dependencies and required different configuration. In the course of our research, we engage with a total of 18 distinct categories of commands and corresponding responses. The system has the potential to accommodate additional features based on user preferences and can be deployed at the user level to provide services to users.

Bibliography

[1] David Griol Michael McTear, Zoraida Callejas. *The Conversational Interface*. 2016.

[2] Mattias Wahde and Marco Virgolin. Conversational agents: Theory and applications. In *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*, pages 497–544. World Scientific, 2022.

[3] Pritika Parmar, Jina Ryu, Shivani Pandya, João Sedoc, and Smisha Agarwal. Health-focused conversational agents in person-centered care: a review of apps, Feb 2022.

[4] Martin Adam, Michael Wessel, and Alexander Benlian. Ai-based chatbots in customer service and their effects on user compliance - electronic markets, Mar 2020.

[5] Gundeep Singh, Sahil Sharma, Vijay Chahar, Manjit Kaur, Mohammed Baz, and Mehedi Masud. Spoken language identification using deep learning. *Computational Intelligence and Neuroscience*, 2021, 09 2021.

[6] Pawan Kumar, Astik Biswas, Achyuta Nand Mishra, and Mahesh Chandra. Spoken language identification using hybrid feature extraction methods. *arXiv preprint arXiv:1003.5623*, 2010.

[7] Shauna Revay and Matthew Teschke. Multiclass language identification using deep learning on spectral images of audio signals. *arXiv preprint arXiv:1905.04348*, 2019.

[8] Dong Yu and Li Deng. *Automatic speech recognition*, volume 1. Springer.

[9] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[10] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, et al. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*, 2021.

[11] Samiul Alam, Asif Sushmit, Zaowad Abdullah, Shahrin Nakkhatra, MD Ansary, Syed Mobassir Hossen, Sazia Morshed Mehnaz, Tahsin Reasat, and Ahmed Imtiaz Humayun. Bengali common voice speech dataset for automatic speech recognition. *arXiv preprint arXiv:2206.14053*, 2022.

[12] Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv.org*, Apr 2022.

[13] Jonathan Owens. Case and proto-arabic, part i. *Bulletin of the School of Oriental and African Studies*, 61(1):51–73, 1998.

[14] John McCarthy. What is artificial intelligence. 2007.

[15] Nahal Norouzi, Kangsoo Kim, Jason Hochreiter, Myungho Lee, Salam Daher, Gerd Bruder, and Greg Welch. A systematic survey of 15 years of user studies published in the intelligent virtual agents conference. In *Proceedings of the 18th international conference on intelligent virtual agents*, pages 17–22, 2018.

[16] Ulrich Gnewuch, Stefan Morana, and Alexander Maedche. Towards designing cooperative and social conversational agents for customer service. In *ICIS*, 2017.

[17] Rodrigo Bavaresco, Diórgenes Silveira, Eduardo Reis, Jorge Barbosa, Rodrigo Righi, Cristiano Costa, Rodolfo Antunes, Marcio Gomes, Clauter Gatti, Mariangela Vanzin, et al. Conversational agents in business: A systematic literature review and future research directions. *Computer Science Review*, 36:100239, 2020.

[18] Rainer Winkler, Sebastian Hobert, Antti Salovaara, Matthias Söllner, and Jan Marco Leimeister. Sara, the lecturer: Improving learning in online education with a scaffolding-based conversational agent. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–14, 2020.

[19] Himanshu Bansal and Rizwan Khan. A review paper on human computer interaction. 2018.

[20] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[21] Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

[22] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[23] Richard S Wallace. *The anatomy of ALICE*. Springer, 2009.

[24] Richard Wallace. The elements of aiml style. *Alice AI Foundation*, 139, 2003.

[25] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part II 16*, pages 373–383. Springer, 2020.

[26] Natasha E. Bajema. Artificial intelligence, china, russia, and the global order. Technical report, Air University Press, 2019.

[27] Jennifer Hill, W Randolph Ford, and Ingrid G Farreras. Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in human behavior*, 49:245–250, 2015.

[28] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[29] Justine Cassell. Embodied conversational agents: representation and intelligence in user interfaces. *AI magazine*, 22(4):67–67, 2001.

[30] Bhavika R Ranoliya, Nidhi Raghuwanshi, and Sanjay Singh. Chatbot for university related faqs. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1525–1530. IEEE, 2017.

[31] Sameera A. Abdul-Kader and John Woods. Question answer system for online feedable new born chatbot. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 863–869, 2017.

[32] Dana Abu Ali and Nizar Habash. Botta: An arabic dialect chatbot. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 208–212, 2016.

[33] Dana Al-Ghadhban and Nora Al-Twairesh. Nabiha: an arabic dialect chatbot. *International Journal of Advanced Computer Science and Applications*, 11(3), 2020.

[34] Trang Nguyen Thi Mai and Shcherbakov Maxim. Enhancing rasa nlu model for vietnamese chatbot. *International Journal of Open Information Technologies*, 9(1):31–36, 2021.

[35] Xuan Li, Huixin Zhong, Bin Zhang, and Jiaming Zhang. A general chinese chatbot based on deep learning and its' application for children with asd. *Int. J. Mach. Learn. Comput*, 10:1–10, 2020.

[36] Yuki Kataoka, Tomoyasu Takemura, Munehiko Sasajima, Naoki Katoh, et al. Development and early feasibility of chatbots for educating patients with lung cancer and their caregivers in japan: mixed methods study. *JMIR cancer*, 7(1):e26911, 2021.

[37] Varad Bhagwat, Mrunal Nagarkar, Pooja Paramane, Shrikant Jindam, and NG Kharate. Review of chatbot system in hindi language. *International Research Journal of Engineering and Technology (IRJET)*, 6, 2019.

[38] Md Kowsher, Farhana Sharmin Tithi, M Ashraful Alam, Mohammad Nurul Huda, Mir Md Moheuddin, and Md Golam Rosul. Doly: Bengali chatbot for bengali education. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pages 1–6. IEEE, 2019.

[39] Dimitrios Ververidis, Constantine Kotropoulos, and Ioannis Pitas. Automatic emotional speech classification. In *2004 IEEE international conference on acoustics, speech, and signal processing*, volume 1, pages I–593. IEEE, 2004.

[40] Santosh K Gaikwad, Bharti W Gawali, and Pravin Yannawar. A review on speech recognition technique. *International Journal of Computer Applications*, 10(3):16–24, 2010.

[41] Jinyu Li et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[42] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[43] Marc A Zissman and Kay M Berkling. Automatic language identification. *speech communication*, 35(1-2):115–124, 2001.

[44] Surabhi Punjabi, Harish Arsikere, Zeynab Raeesy, Chander Chandak, Nikhil Bhave, Markus Mueller, Sergio Murillo, Ariya Rastrow, Andreas Stolcke, Jasha Droppo, Sri Garimella, Roland Maas, Mat Hans, Athanasios Mouchtaris, and Siegfried Kunzmann. Joint asr and language identification using rnn-t: An efficent approach to dynamic language switching. In *ICASSP 2021*, 2021.

[45] Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. A survey of joint intent detection and slot filling models in natural language understanding. *ACM Computing Surveys*, 55(8):1–38, 2022.

[46] Liang Huang, Senjie Liang, Feiyang Ye, and Nan Gao. A fast attention network for joint intent detection and slot filling on edge devices, 2022.

[47] Di Wu, Liang Ding, Fan Lu, and Jian Xie. Slotrefine: A fast non-autoregressive model for joint intent detection and slot filling. *arXiv preprint arXiv:2010.02693*, 2020.

[48] Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. *arXiv preprint arXiv:1810.13327*, 2018.

[49] Hayet Brabra, Marcos Baez, Boualem Benatallah, Walid Gaaloul, Sara Bouguelia, and Shayan Zamanirad. Dialogue management in conversational systems: a review of approaches, challenges, and opportunities, Mar 2022.

[50] Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung-Hsien Wen, Nikola Mrkšić, and Pei-Hao Su. Training neural response selection for task-oriented dialogue systems. *arXiv preprint arXiv:1906.01543*, 2019.

[51] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

[52] Vladimir Vlasov, Akela Drissner-Schmid, and Alan Nichol. Few-shot generalization across dialogue tasks. *arXiv preprint arXiv:1811.11707*, 2018.

[53] Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*, 2019.

[54] Sara Perez-Soler, Sandra Juarez-Puerta, Esther Guerra, and Juan de Lara. Choosing a chatbot development tool. *IEEE Software*, 38(4):94–103, 2021.

[55] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

[56] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. Slurp: A spoken language understanding resource package. *arXiv.org*, Nov 2020.

[57] Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197, 2011.

[58] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. Rasa: Open source language understanding and dialogue management. *arXiv.org*, Dec 2017.

[59] Dialogflow | google cloud.

[60] Amazon lex.

[61] Microsoft bot framework.

[62] About chatterbot mdash; chatterbot 1.0.8 documentation.

[63] Sara Perez-Soler, Sandra Juarez-Puerta, Esther Guerra, and Juan de Lara. Choosing a chatbot development tool. *IEEE Software*, 38(4):94–103, 2021.

[64] Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. Diet: Lightweight language understanding for dialogue systems, 2020.

[65] Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, and Ivan Vulić. Convert: Efficient and accurate conversational representations from transformers. *arXiv.org*, 2020.

[66] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, October 2014.

[67] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[68] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[69] Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. How language-neutral is multilingual bert?, 2019.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[71] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[72] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding, Jul 2020.

[73] Vladimir Vlasov, Johannes E. M. Mosig, and Alan Nichol. Dialogue transformers. *arXiv.org*, Oct 2019.

[74] Samiul Alam, Tahsin Reasat, Asif Shahriyar Sushmit, Sadi Mohammad Siddique, Fuad Rahman, Mahady Hasan, and Ahmed Imtiaz Humayun. A large multi-target dataset of common bengali handwritten graphemes. In *International Conference on Document Analysis and Recognition*, pages 383–398. Springer, 2021.

[75] Training data format, rasa, May 2023.

[76] Stories, rasa, May 2023.

[77] Rules, rasa, May 2023.

[78] Precision and recall, Jul 2019.

[79] F-score, Dec 2018.

[80] Confusion matrix, Jul 2019.