S	
E1	
E2	
For Office Use Only	

Masters Project Final Report (MCS) 2019

Project Title	Sinhala Chatbot for Train Information
Student Name	S. A. D. U. Harshani
Registration No. & Index No.	16440394
Supervisor's Name	Dr. A. R. Weerasinghe

For Office Use ONLY

2019

Sinhala Chatbot for Train Information

A dissertation submitted for the Degree of Master of Computer Science

S. A. D. U. Harshani

University of Colombo School of Computing

2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: S. A. D. U. Harshani

Registration Number: 2016/MCS/039

Index Number: 16440394

____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr./Ms. S. A. D. U. Harshani under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Dr. A. R. Weerasinghe

____

Signature: Date:

Acknowledgement

Upon the completion of this research, I would extend heartfelt gratitude to many people.

First of all, I would like to gratefully and sincerely thank research supervisor, Dr Ruwan Weerasinghe for his guidance, understanding, patience, motivation and most importantly their kindness during this research work.

Then I would also like to thank Dr Kasun Gunawardana, Co-coordinator of MCS individual research projects for his valuable support. Next, I would like to thank the management of the Postgraduate unit of UCSC for allowing doing the MCS degree and for their excellent support during the entire period of this research.

Then I like to thank all the senior lectures of UCSC, for their assistance. I also like to thank all the lecturers, Academic, non-academic staff of UCSC for their great support throughout the Postgraduate degree.

I am very grateful to my friends who followed MCS and MSC with me for their support and most importantly for their friendship. During these two years of the Master's degree program, we worked as a family. Thank you very much for everything. I am very thankful to all batchmates for their support and friendship.

I want to thank Codegen International which is my working place for all the encouragement they have given. Moreover, also I would like to thank my Team members of Codegen International are for making me comfortable to follow the Master's degree program even in the critical deadlines of the projects.

I also like to thank all the Senior Lectures, lecturers, Academic, the non-academic staff of the department of computer science, University of Ruhuna, for their great support throughout my first degree.

Then I would like to all my teacher of St. Pauls‟ Girls‟ school, Udawalawa M.V. and the other teacher I met during my life, without them, I would not be able to obtain a university entrance.

Then, most importantly, I would like to thank my mother Amara Sepali and father S. D. Rathnasiry for their unconditional support and encouragement. Moreover, I like to thank my loving brother, Janith, for his excellent support during my Master's degree period.

Even though some names may not be mentioned, but I would like to thank everyone who helped me in various ways, supported me, make me stronger, and encourage me. I thank you All !!!

Table of Content

Table of Figures

Abstract

A chatbot is an essential point at the natural evolution of a Question Answering system in parallel to Natural Language Processing (NLP) improvements. Chatbot applications enhance the interactions between people and services with enhancing customer experience. More accurate chatbot systems offer companies new opportunities to improve the customer's engagement with the service by reducing the typical cost of customer service. This research is on the design and implementation of the domain-specific Sinhala Chatbot System that can communicate between computer and user, through the Sinhala language.

The entire system has been developed using the JAVA programming language. The main objective behind this approach was to understand user-required information and provide the ideal answers considering various user circumstances. The proposed framework includes an NLP service and a context manager. NLP service contains Intent classifier, Entity identifier and Response generator, implemented for Sinhala Language processing with Supervised Learning in Neural Network, trained with the data, collected from the end users.

We have introduced a framework that can use for the implementation of a chatbot in Sinhala for any domain. By using the proposed framework, we present a simple chatbot system which accepts user input in Sinhala and generate replies extracted from the training data set in the domain of Train information in the Sinhala language.

Chapter One

Introduction

The computer-based chat system is a prevalent communication method in the modern world. Most of these chat systems use a limited capability of responding to user requests based on the service domain.

Mainly there are two types of chat systems.

1. Human - Human chat system
2. Human-Computer char systems.

Human - Human chat systems act as a mediator between two humans who communicate with each other. It can consider a simple conversation that occurs between two parties.

Human-Computer Chat systems are more challenging. Term Chat Bot use for this kind of chat systems. Chatbots have known to be one of the most emerging technologies in information technology. Most of the companies and services have already trying out Chatbots in their customer service lines to provide their customers with a better service and optimise their business process. Chatbots is a technical tool which enables us to replace "human" representative of the conversation scenario. In order to achieve this, unique mechanisms such as NLP (Natural Language Processing), Machine Learning techniques have used.

Both of these two types of chat systems are available in the English language. Among these two, Chatbot has become an essential part of many business processes. Most of the Chatbots developed for a specific domain. Such as Hotel Reservation, Flight Booking, Customer support.

There are only a few Sinhala Chat bots implementation attempts are available. Among them implementing a domain-specific Sinhala Chatbot is still in the research level. The language has become a significant barrier since most of the supplementary algorithms; techniques are developed based on the English language.

In this Research project, we wanted to implement a Domain Specific Sinhala Chatbot. As the domain of this Chatbot, we have selected Train Information.

In this research project, we hope to implement a Sinhala Chatbot which can be used to communicate through Sinhala natural language for the Train Information retrieval domain.

In Sri Lanka trains have been one of the major public transportation methods. However, in most cases, people have to face many difficulties in using trains because they are unable to access train schedules and other related information. Most of the information that they get is outdated or incorrect. Because of these issues, people miss out of using trains for their daily transportation.

Today some people tend to retrieve train schedules by going through the Sri Lanka Railway Website or by contacting nearest Railway station or Government information centre of Sri

Lanka. Both of these methods are time-consuming. In some cases, they will not even get the relevant information they require.

In order to overcome all these difficulties and provide a reliable, efficient service for all the Sri Lankans, we propose a solution of implementing a Chatbot which is capable of providing train schedules, railway station information and any other related information that would matter for the train users

1.1 Background

In 1950, Alan Turing's famous article "Computing Machinery and Intelligence" was published, which proposed what is now called the Turing test as a criterion of intelligence. This article is the first idea of the intelligent machine. Stimulated by the suggested Turing test, Joseph Weizenbaum's program ELIZA, published in 1966, which seemed to be able to fool users into believing that they were conversing with a real human. It was able, at least for a time to pass the Turing Artificial Intelligence test. However Weizenbaum himself did not claim that ELIZA was genuinely intelligent, and the Introduction to his paper presented it more as a debunking exercise:

ELIZA's principal method of operation (copied by chatbot designers ever since) involves the recognition of cue words or phrases in the input and the output of corresponding pre-prepared or pre-programmed responses that can move the conversation forward in a meaningful way.

Most people prefer to engage with human-like programs, and this gives chatbot-style techniques a potentially useful role in interactive systems that need to elicit information from users, as long as that information is relatively straightforward and falls into predictable categories. Thus, for example, online help systems can usefully employ chatbot techniques to identify the area of help that users require, potentially providing a "friendlier" interface than a more formal search or menu system. This sort of usage holds the prospect of moving chatbot technology.

1.2 Problem Definition

At present, the majority of these chat systems are available in the English language. Therefore people who do not speak fluent in the English Language, cannot use these chat systems due to the apparent reason for the Language barrier. The language barrier has been an issue not only for communicating with the Chatbot system but also contributing to the discovery of knowledge by the persons whose mother tongue is different from English.

The foundation of conducting this research in order to address our main purpose is to develop a Chatbot system which is used the Sinhala language as the communication language.

1.3 Motivation

A chatbot is an intelligent system. There are so many useful applications available to developing intelligent Chatbots. This is mainly because; humans want to communicate with various resources through appropriate interfaces.

Domain-specific chatbots are heavily used by competitive fields, such as travel agencies, hotel booking services, and other businesses built around travel and tourism. In this research, we are developing a chatbot to retrieve information about trains and other details related to the Sri Lanka Railway.

Currently, Sri Lanka railway does not have a call centre which can assist Train passengers. They only have the option to contact the nearest Railway station via a Telephone call. However, since most of the Railway stations have not assigned a telephone operator, passengers have faced many difficulties to contact Railway stations. From the other side, people who worked in the Railway station has to put extra effort to answer all of those telephone calls, and it will be annoying for them as well. Because of these reasons, passengers will not be able to receive friendly and useful service from them.

Another option is to retrieve details from the Sri Lanka Railway website. However, search and find information from a website is very time-consuming.

As an alternative way, frequent train passengers have got together, and created communities using social media and they try to send information and retrieve information via these community chats. However, it is not 100% accurate. Because getting information from those kinds of communities are depending on the knowledge of the members of the community. Sometime we will not be able to get sufficient information and also most of the people exchanging information, and it is hard to find the information we required on time. Another problem is that most of the people scared to use these kinds of communities.

In order to address the above issues, we have developed a chatbot which can use to retrieve information about Trains. When using these kinds of chatbot, the major problem is the language. Since everyone is not able to use the English language, we have decided to implement this chatbot in a way the user can communicate using the Sinhala language. This chatbot can answer the user’s question faster and politely in Sinhala.

Since chatbot is an emerging technology which is heavily using in various types of businesses, we have decided that chatbot will be the best solution to address the problems of Trains travellers.

1.4 Objectives

The main objectives of the project are to develop an algorithm that will be used to identify answers related to user-submitted questions and mapped them into the Sinhala language. To develop a database to store all the related data such as questions, answers, keywords, logs and feedback messages and to develop a web interface. The web interface developed had two parts, one for simple users and one for the administrator.

In order to archive the primary objective, some other objectives need to be archived.

1. Implement an Intent identification for the Sinhala language
2. Implement an entity recognition method
3. Develop an appropriate method to generate a response or the identified intents and entities.

A chatbot will read input sentence from user and analyses Syntax and Semantic of the sentence. By using analysed results, it will extract the knowledge and identify the intent of the user and identify the appropriate answer.

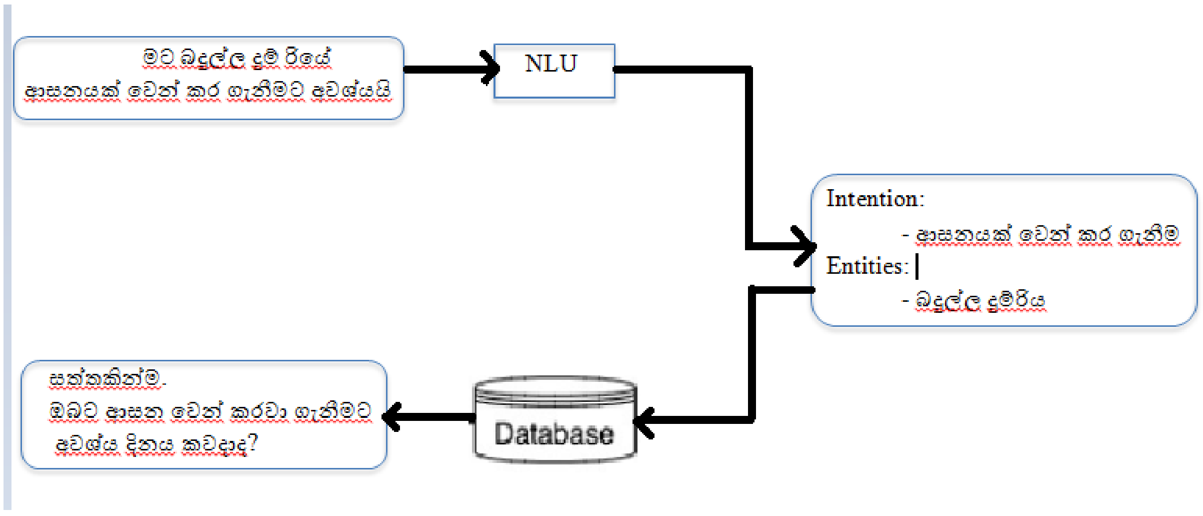

මට බදුල්ල දුම් රියේ ආසනයක් වෙන් කර ගැනීමට අවශ්‍යයි
NLU
Intention:
- ආසනයක් වෙන් කර ගැනීම
Entities:
- බදුල්ල දුම්රිය
Database
සත්තකින්ම.
ඔබට ආසන වෙන් කරවා ගැනීමට අවශ්‍ය දිනය කවදාද?

Figure 1: Basic Structure of the Chatbot

Intent Identification

One of the significant tasks of chatbots is intent classification. As simple as it may sound, it is quite a complicated process. For that, generally, a model is trained either using a supervised or unsupervised approach based on the number of labelled user queries or spoke utterances available in order to be able to detect the intent of input user query/input. The challenge of this research is to conduct this process for the Sinhala language.

The goal of the Intent identification is to determine the entities and intent of natural language sentences. For instance, if we analyse the following sentence: “මට බදුල්ල දුම්රියේ ආසනයක් වෙන් කර ගැනීමට අවශ්‍යයි.”, an intent-based NLP algorithm can be used to determine the intent of the sentence as “ආසනයක් වෙන් කර ගැනීම” while the main entity remains “බදුල්ල දුම්රිය”.

Generate Response

In this research, we are building a chatbot framework to process responses according to the user intent and the knowledge identified from the user queries.

1.5 Scope

Chatbot we are going to implement from this project is a text-based AI (Artificial Intelligence) chatbot that receives questions from users in the Sinhala language, tries to understand the question, and provides appropriate answers. The mechanism behind this process is identifying the user's intent through intent classification and provides the relevant answer for the user request. It does this by converting a Sinhala sentence into a machine-friendly query, then going through relevant data to find the necessary information, and finally returning the answer in a natural language sentence. In other words, it answers user's questions like a human does, instead of giving the list of websites that may contain the answer for every question. For example, when it receives the question " රුහුණු කුමාරි දුම් රිය මාතරින් පිටත් වන්නේ කීයටද?", it will give a response "රුහුණු කුමාරි දුම් රිය දිනපතා මාතරින් පෙරවරු 6.05ට පිටත්වේ."

This chatbot will be implemented to retrieve train information. This chatbot will be implemented to answer the questions which are relevant to the facilities available in the Sri Lanka railways website.

If the user asks about something which cannot be replied by the Chatbot or which is not related to the domain, it will navigate the user to the Sri Lanka railway website to search required information by them.

Mainly there are two types of chatbots, Voice base and Text based. In this approach, we are only focusing on implementing a Text-based chatbot.

Chapter Two

Related Research or Previous Work

Chatbots are becoming popular as a means for interactive communication between human and machines. Due to their interactivity, chatbots are much better than standard machine translation systems, which may provide unrealistic solutions when the system cannot perform without user intervention.

2.1. Existing Chatbot Systems

ELIZA is an early Artificial Intelligent program to simulate a non-directive psychotherapist[1] In this chatbot, Input sentences analysed by decomposition rules. However, ELIZA had minimal natural language processing capabilities.

ELIZA is a program operating within the MAC time-sharing system at MIT which makes certain kinds of natural language conversation between man and computer possible. Input sentences analysed by decomposition rules which are triggered by keywords appearing in the input text. Responses generated by reassembly rules associated with selected decomposition rules. The fundamental technical problems with which ELIZA is concerned are:

- The identification of keywords
- The discovery of minimal context
- The choice of appropriate transformations
- Generation of responses in the absence of keywords
- the provision of an editing capability for ELIZA "scripts."

A discussion of some psychological issues relevant to the ELIZA approach as well as of future developments concludes the paper[1]

Elizabeth is another Chatbot system that adaptation of the Eliza[2] She uses to store knowledge as a script in a text file. Each line started with a script command notation. Moreover, Elizabeth can produce a grammar structure analysis of a sentence using a set of input transformation rules to represent grammar rules.

ALICE is another chatbot system that can be used to chat with using natural language[3]. It uses AIML files to implement its knowledge. It was developed to enable people to input dialogue pattern knowledge into Chat bots based on the ALICE free software technology. ALICE uses pattern-matching algorithms to identify user input, and this algorithm uses depth-first search techniques.

User interfaces for software applications can come in a variety of formats, ranging from command-line, graphical, web application, and even voice. While the most popular user interfaces include graphical and web-based applications, occasionally the need arises for an alternative interface. Whether due to multi-threaded complexity, concurrent connectivity, or details surrounding the execution of the service, a chatbot based interface may suit the need.[4]

Chatbots typically provide a text-based user interface, allowing the user to type commands and receive a text as well as text to speech response. Chatbots are usually stateful services, remembering previous commands (and perhaps even conversation) in order to provide functionality. When chatbot technology integrated with popular web services, it can be utilised securely by an even larger audience[4]

There is a Sinhala Chatbot system already implemented[5]. It can be used to communicate through Sinhala Natural language. It was implemented to answer simple questions and also it was trained to do some simple operations.

2.2. Chatbots in other languages (French and Arabic)

Software to machine-learn conversational patterns from a transcribed dialogue corpus has been used to generate a range of chatbots speaking various languages and sublanguages including varieties of English, as well as French, Arabic and Afrikaans.[6]

"Using corpora in machine-learning chatbot systems"[6] present a program to learn from spoken transcripts of the Dialogue Diversity Corpus of English, the Minnesota French Corpus, the Corpus of Spoken Afrikaans, the Qur'an Arabic-English parallel corpus, and the British National Corpus of English; They[6] have discussed the problems which arose during learning and testing.

Two main goals achieved from the automation process.[6] One was the ability to generate different versions of the chatbot in different languages, bringing chatbot technology to languages with few if any NLP resources: the corpus-based learning techniques transferred straightforwardly to develop chatbots for Afrikaans and Quranic Arabic. The second achievement was the ability to learn a considerable number of categories within a short time, saving effort and errors in doing such work manually: we generated more than one million AIML categories or conversation-rules from the BNC corpus, 20 times the size of existing AIML rule-sets, and probably the biggest AI Knowledge-Base ever.

There are chatbot systems implemented in the Arabic language. They have presented machine learning techniques used to generate an Arabic chatbot, which accepts user input in Arabic and generates replies extracted from Qur'an[7].

This system[7] is to learn conversational patterns from a Corpus of transcribed conversation have been used to generate a range of chatbots speaking different languages including English, French and Afrikaans. They[7] have reviewed aspects of the Arabic language, which pose problems for chatbot-learning, and they[7] have discussed the revised process to handle Arabic training text and input/output. In principle, the Qur'an provides guidance and answers to religious and other questions. Therefore they have used the Qur'an as a training corpus for their chatbot.

Since the Qur'an is not a transcription of a conversation, they[7] have adopted the learning process to cope with the structure of the Qur'an in terms of sooras and ayyas. Their resulting system[7] accepts user input in Arabic and answers with appropriate ayyas from Qur'an.

2.3. Components of a Chatbot

There is a specific vocabulary for Chatbots and Natural language processing[8]. Learn this vocabulary to make the user learn Natural language processing and Chatbot developments. Machine learning can be used to build Neural conversational models and improve the usefulness of the Chatbot systems[9]

The Log Answer system is an application of automated reasoning to the field of open domain question answering.[10] In order to find answers to natural language questions regarding arbitrary topics, the system integrates an automated theorem prover in a framework of natural language processing tools.

LogAnswer[10] is a QA system supporting the German language. The system originates from the logic based answer validator MAVE[11] (developed for the multi-stream QA system IRSAW).[12] The validator was a redesign for real-time question answering and extended to a full QA system by adding a logic-based answer extractor, a user interface and other components.

LogAnswer works with a knowledge base derived from textual sources namely the German Wikipedia and corpus of newspaper articles. Answers are produced using a combination of deep linguistic processing and automated theorem proving[10]

Chatbots are a huge trend. Big name brands are jumping at the opportunity to meet their customers where they are already spending time—in messaging apps. That is why we are seeing a massive number of bots appearing in various industries, from e-commerce and fashion to more conservative sectors like banking.[13]

Advancements in conversational UIs and artificial intelligence (AI) in the events industry may not be as fast and impactful yet as in some other sectors, but these two technologies have the potential to redefine attendee experience and enable smarter event management for the organizers.

Because of the three major trends that are creating the perfect storm for the rise of chatbots:[9]

- the ever-present app fatigue
- the explosion of messaging apps

- the increased commoditization of AI algorithms.

Companies are building their chatbots leveraging AI for their domain of knowledge and ability to personalise conversations. These branded bots live on existing messaging apps, such as Facebook Messenger, Telegram, Kik, Viber and more.

Chatbots we interact with today may be built to do simple things, but as bot makers start to leverage advanced machine learning technologies, bots will become more intelligent and capable of interacting with users in more contextually relevant ways.

Chatbots learn to do new things by trawling through a vast swath of information. They are designed to spot patterns and repeat actions associated with them when triggered by keywords, phrases or other stimuli. When Chatbots are used to gather customer insight they can improve on all stages of the marketing funnel. Chatbots can store information about the kind of questions that the customer asks -> respond with the backdated information available & by reading through a large amount of data in a quick amount of time -> predict the user actions -> escalate the conversation if need be and resolve it at best possible pace. [14]

Chatbot developers usually use two technologies to make the bot understand the meaning of user messages[15]

1. machine learning
2. hardcoded rules

Machine learning can help to identify the intent of the message and extract named entities. It is quite powerful, but it requires lots of data to train the model. If there were no enough labelled data, then handcrafting rules can be used to identify the intent of a message. Rules can be as simple as “if a sentence contains words ‘pay’ and ‘order’ then the user is asking to pay for an order”. [15]

Chatbot needs a preprocessing NLP pipeline to handle typical errors. It may include these steps:

- **Spellcheck**:- Get the raw input and fix spelling errors. Can do something very simple or build a spell checker using deep learning.
- **Split into sentences**:- It is beneficial to analyse every sentence separately. Splitting the text into sentences is easy, can use one of NLP libraries
- **Split into words**:- This is also very important because hardcoded rules typically operate with words. Same NLP libraries can do it.
- **POS tagging**:- Some words have multiple meanings, for example, “charge” as a noun and “charge” as a verb. Knowing a part of speech can help to disambiguate the meaning. Can use the same NLP libraries,

or Google SyntaxNet, that is a little bit more accurate and supports multiple languages.

- **Lemmatize words**:- One word can have many forms: “pay”, “paying”, “paid”. In many cases, an exact form of the word is not essential for writing a hardcoded rule. If the preprocessing code can identify a lemma, a canonical form of the word, it helps to simplify the rule.
- **Entity recognition: dates, numbers, proper nouns**:- Dates and numbers can be expressed in different formats: “3/1/2016”, “1st of March”, “next Wednesday”, “2016–03–01”, “123”, “one hundred”. It may be helpful to convert them to a unified format before doing pattern matching.
- **Find concepts/synonyms**:- WordNet can be used to identify familiar concepts. It might need to add domain specific concept libraries

This paper[16] describes a flexible method of teaching introductory artificial intelligence (AI) using a novel, Java-implemented, simple agent framework developed specifically for this course. Although numerous agent frameworks have proposed in the vast body of literature, none of these available frameworks proved to be simple enough to be used by first-year students of computer science.

Hence, the authors set out to create a novel framework that would be suitable for the aims of the course, for the level of computing skills of the intended group of students and the size of this group of students.[16] The content of the introductory AI course in question is a set of assignments that require the students to use intelligent agents and other AI techniques to monitor, filter and retrieve relevant information from the World Wide Web. It represents, therefore, a synthesis of the traditional objectivist approach and a real-world-oriented, constructivist approach to teaching programming to novices.[16] The main aim of implementing such a pedagogy was to engage the students in learning to which they relate while attaining intellectual rigour. Classroom experience indicates that students learn more effectively when the traditional objectivist approach combined with a constructivist approach than when this orthodox approach to teaching programming to novices is used alone.

2.4. Technology Frameworks for Chatbots

Dialogflow (formerly Api.ai, Speaktoit) is a Google-owned developer of human-computer interaction technologies based on natural language conversations.[17] Api.ai is a service that allows developers to build speech-to-text, natural language processing, artificially intelligent systems that you can train up with your custom functionality.

They have a range of existing knowledge bases that systems built with Api.ai can automatically understand called “Domains” — which is what we will be focusing on in this article. Domains provide a whole knowledge base of encyclopedic knowledge, language translation, weather and more.[17]

Watson is an IBM supercomputer that combines artificial intelligence (AI) and sophisticated analytical software for optimal performance as a "question answering" machine. The supercomputer named for IBM's founder, Thomas J. Watson[18].

There are plenty of easy to use bot building frameworks developed by big companies. Dialogflow (formerly known as API.AI; developed by Google) and Bot Framework (developed by Microsoft) are some examples. Both Dialogflow and Bot Framework have pre-built custom language understanding modes. These frameworks seem to be great tools to get started quickly, especially if they do not have existing chat logs that can be used as training data. In order to develop a chatbot for a business and this chatbot will be receiving potentially sensitive or confidential information from its users. In such a case, it may be more comfortable keeping all the components of the chatbot in the house. In this case, we can use the Rasa platform[19].

Rasa[20] is an open source bot building framework. It does not have any pre-built models on a server that can be called using an API, which means that it will take more work to get it running. Rasa stack consists of two major components: Rasa NLU and Rasa Core. Rasa NLU is responsible for natural language understanding of the chatbot. Its primary purpose is, given an input sentence, predict the intent of that sentence and extract useful entities from it. Rasa Core is the next component in the Rasa stack pipeline. It takes structured input in the form of intents and entities (output of Rasa NLU or any other intent classification tool) and chooses which action the bot should take using a probabilistic model (to be more specific, it uses LSTM neural network implemented in Keras).

Many applications are incorporating a human appearance and intending to simulate human dialogue, but in most of the cases, the knowledge of the conversational bot is stored in a database created by human experts.[21] However, very few researches have investigated the idea of creating a chat-bot with an artificial character and personality starting from web pages or plain text about a certain person. This paper describes an approach to the idea of identifying the most critical facts in texts describing the life (including the personality) of a historical figure for building a conversational agent that could be used in middle-school CSCL scenarios.

MILABOT[22] is a deep reinforcement learning chatbot developed by the Montreal Institute for Learning Algorithms (MILA) for the Amazon Alexa Prize competition. MILABOT[22] is capable of conversing with humans on popular small talk topics through both speech and text.

The system[22] consists of an ensemble of natural language generation and retrieval models, including template-based models, bag-of-words models, sequence-to-sequence neural network and latent variable neural network models. By applying reinforcement learning to crowd-sourced data and real-world user interactions, the system[22] has been trained to select an appropriate response from the models in its ensemble.

The system[22] has evaluated through A/B testing with real-world users, where it performed significantly better than many competing systems. Due to its machine learning architecture, the system is likely to improve with additional data.

ONTBOT[23] is another approach for chatbot implementation. It is a working model of Ontology-based chatbot, and it proposed that handles queries from users for an E-commerce website. It is mainly concerned with providing a user with total control over the search result on the website.

This chatbot[23] helps the user by mapping relationships of the various entities required by the user, thus providing detailed and accurate information thereby overcoming the drawbacks of traditional chatbots.

The Ontology template is developed using Protégé which stores the knowledge acquired from the website APIs while the dialogue manager is handled using Wit.ai. The integration of the dialogue manager and the ontology template managed through Python.[23] The related response to the query will be formatted and returned to the user on the dialogue manager.

There is an approach[24] which considers incorporating topic information into the sequence-to-sequence framework to generate informative and interesting responses for chatbots. To this end, They have proposed a topic aware sequence-to-sequence (TA-Seq2Seq), model.

The model[24] utilises topics to simulate prior knowledge of human that guides them to form informative and interesting responses in conversation and leverages the topic information in the generation by a joint attention mechanism and a biased generation probability. The joint attention mechanism summarises the hidden vectors of an input message as context vectors by message attention, synthesises topic vectors by topic attention from the topic words of the message obtained from a pre-trained LDA model, and let these vectors jointly affect the generation of words in decoding. To increase the possibility of topic words appearing in responses, the model[24] modifies the generation probability of topic words by adding an extra probability item to bias the overall distribution. An empirical study on both automatic evaluation metrics and human annotations shows that TA-Seq2Seq can generate more informative and interesting responses, and significantly outperform the-state-of-the-art response generation models.

2.5. Evaluation of Chatbot Systems

For the annual Loebner Prize contest, rival chatbots have assessed in terms of ability to fool a judge in a restricted chat session. Research on Different measurement metrics to evaluate a chatbot system has been investigated methods to train and adopt a chatbot to a specific user's language use or application, via a user-supplied training corpus[25].

They[25] have advocated open-ended trials by real users, such as an example Afrikaans chatbot for Afrikaans-speaking researchers and students in South Africa. This[25] evaluated in terms of "glass box" dialogue efficiency metrics, and "black box" dialogue quality metrics and user satisfaction feedback. Another example presented in this paper[25] was the Qur'an and the FAQchat prototypes. Their[25] general conclusion is that evaluation should be adapted to the application and user needs.

Another approach[26] on evaluating chatbot systems are, investigated if and how an artificially intelligent chat agent (chatbot) that answers questions about sex, drugs, and alcohol is used and evaluated by adolescents, especially in comparison with information lines and search engines.

Another evaluation approach[27] was investigating the linguistic worth of current ‘chatbot’ programs – software programs which attempt to hold a conversation or interact, in English – as a precursor to their potential as an ESL (English as a second language) learning resource.

After some initial background to the development of chatbots, and a discussion of the Loebner Prize Contest for the most ‘human’ chatbot (the ‘Turing Test’), this paper[27] describes an in-depth study evaluating the linguistic accuracy of many chatbots available online. Since the ultimate purpose of the current study concerns chatbots' potential with ESL learners, the analysis of language embraces not only an examination of features of language from a native speaker's perspective (the focus of the Turing Test) but also aspects of language from a second-language users perspective

Analyses[27] indicate that while the winner of the 2005 Loebner Prize is the ablest chatbot linguistically, it may not necessarily be the chatbot most suited to ESL learners. The paper concludes that while substantial progress has made in terms of chatbots' language-handling, a robust ESL ‘conversation practice machine’ (Atwell, 1999) is still some way off being a reality.

When considering the evaluation, Quality is one aspect which needs to evaluate. Since chatbots are now easier to train and implement due to plentiful open source code, widely available development platforms, and implementation options via Software as a Service (SaaS). In addition to enhancing customer experiences and supporting learning, chatbots can also be used to engineer social harm - that is, to spread rumours and misinformation, or attack people for posting their thoughts and opinions online.[28]

This paper[28] presents a literature review of quality issues and attributes as they relate to the contemporary issue of chatbot development and implementation. Finally, quality assessment approaches reviewed, and a quality assessment method based on these attributes and the Analytic Hierarchy Process (AHP) proposed and examined

Another research[29] has been used the ALICE/AIML chatbot architecture as a platform to develop a range of chatbots covering different languages, genres, text-types, and user-groups, to illustrate qualitative aspects of natural language dialogue system evaluation.

They[29] have presented some of the different evaluation techniques used in natural language dialogue systems, including black box and glass box, comparative, quantitative, and qualitative evaluation. Four aspects of NLP dialogue system evaluation are often overlooked: "usefulness" in terms of a user's qualitative needs, "localizability" to new genres and languages, "humanness" or "naturalness" compared to human-human dialogues, and "language benefit" compared to alternative interfaces. They[29] have illustrated those aspects for their work on machine-learnt chatbot dialogue systems;

2.6. Implementing Chatbot in Sinhala

Since we are going to implement our chatbot system in Sinhala, we have searched about the methods that can be used to archive our goal.

One approach we have thought about is machine translation[30]. Machine translation is a challenging task in natural language processing. Out-of-vocabulary, handling proper nouns and technical terms are some significant issues which are common to all machine translation systems. This paper[30] presents a transliteration approach to machine translation from English to Sinhala. They[30] have used finite state automaton to develop transducers for English to Sinhala transliteration. This approach[30] can transliterate the text in original English and Sinhala words that are written using English letters. The transliteration system has been developed using SWI-PROLOG and prologue server page (PSP). English WorldNet and Sinhala Chatbot are used to test the transducers, and reasonable results achieved.

In addition to translation, the other option we have is to implement and train the chatbot in Sinhala using a rule-based method. In order to use this approach, it is critical to handle the Sinhala language. Identifying syllables will be very important.

There is a paper we have found which presents a study of Sinhala syllable structure and an algorithm for identifying syllables in Sinhala words[31]. After a thorough study of the Syllable structure and linguistic rules for syllabification of Sinhala words and a survey of the relevant literature, a set of rules was identified and implemented as a simple, easy to implement the algorithm. The algorithm was tested using 30,000 distinct words obtained from a corpus and compared with the same words manually syllabified. The algorithm performs with 99.95 % accuracy.

Chapter Three

Problem Analysis and Methodology

In this section, all the steps carried out in each approach to find solutions to the research questions will describe. In order to find the solution to our research question we mainly focus on implementing a chatbot which use the Sinhala language as the communication language to help train passengers to retrieve information about Trains such as time table, ticket price and reservation facilities. In this section, we mainly focus on describing the method we have used on implementing a domain-specific Chatbot in Sinhala.

In section 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6 describe the methodology we use to find the solutions for the subquestions mentioned in section 1.2.

3.1. Problem Analysis

A chatbot (also known as a talkbot, chatterbot, Bot, IM bot, interactive agent, or Artificial Conversational Entity) is a computer program which conducts a conversation in natural language via auditory or textual methods, understands the intent of the user, and sends a response based on business rules and data of the organisation.

The idea of chatbots appeared first in the 1960s. However, only after more than half a century passed we can confess that the world is ready for their implementation into the real life, being a result of the rapid progress in natural language processing, AI, and the global presence of text messaging applications.

The tendencies in communication technologies indicate that text communication became a socially acceptable form of personal interaction. People increasingly prefer chatting rather than personal contacts or even making phone calls.

All the technology high rollers have created open platforms and interfaces for the chatbot acceptance by society to pass easier and faster. Microsoft, Facebook, Google, Amazon, IBM, Apple and Samsung, they all have worked to create their chatbots.

Siri was introduced in 2010, IBM Watsons started in 2011, the pilot version of the Bixby Samsung voice assistant appeared in smartphones in 2012. Alexa have been learning to answer the questions since 2014, and the Google Assistant had gained its modern shape in 2016.

It is 2018 now, but still, the majority of these chat systems are available in the English language. Therefore people who do not speak fluently in the English Language, cannot use these chat systems due to the obvious reason for the Language barrier. The language barrier has been an issue not only for communicating with the Chatbot system but also contributing to the discovery of knowledge by the persons whose mother tongue is different from English.

The foundation of conducting this research in order to address our primary purpose is to develop a Chatbot system which is used the Sinhala language as the communication language.

3.2. Methodology

Create a conversational user interface between users and information assistant (Bot) in order to provide information based on user requirements. The primary objective behind this approach is to understand user-required information and provide the ideal answers considering various user circumstances.

Chatbots can divide into two groups: **scripted** and **AI chatbots**. The first group describes chatbots that can conduct a guided conversation; however, their actions are limited by rules. Scripted chatbots are widely used in customer service as they are a safe tool that can interact with clients according to a planned scenario.

When it comes to AI chatbots, they represent more advanced technology. They can understand more than scripted bots and can deal with notions they were not prepared. In this section, we are describing the methodology we have followed in implementing an AI chatbot which uses the communication language as the Sinhala language.

3.2.1. Data Collection

The standard process of the data collection is to collect data from various chat applications which are already available online. Most of the companies have made these substantial conversational data sets freely available.

However, in this case, we would not be able to use the same approach. Because since we need data in the Sinhala language, there are no data sources we can collect appropriate data. Therefore we have used the survey method to collect data from the daily train travellers. Apart from these data sets some amount of data can be collected from the dialogue management systems and translated them to Sinhala as well.

3.2.2. Data Cleaning

Since we have collected data directly from end-user data was not formatted. This data collection had garbage data such as Sinhala text written using English alphabet letters; Sentences contain a mix of both Sinhala and English words, incomplete sentences.

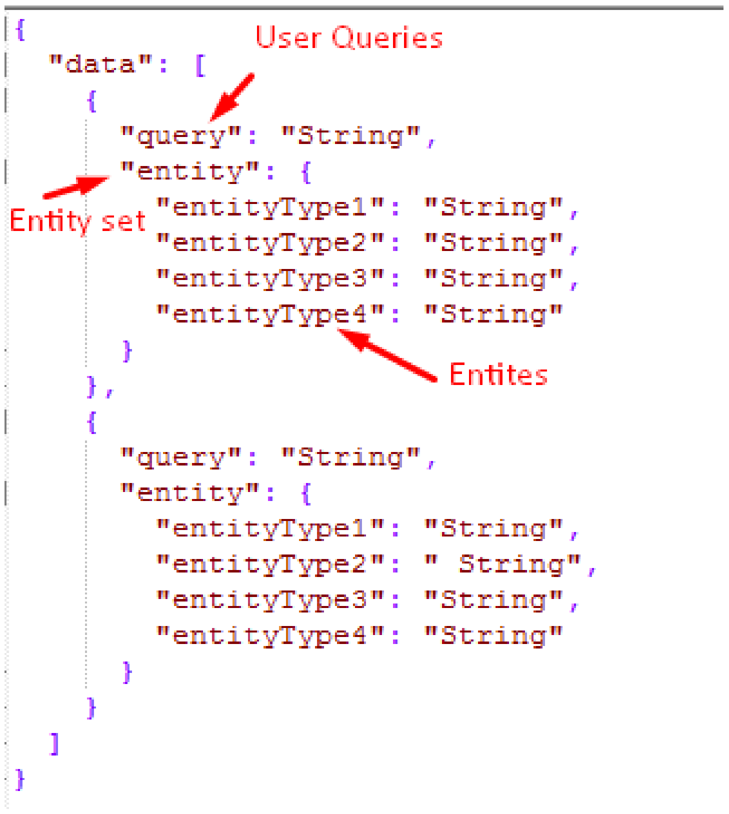
Because of the reasons mentioned above, we had to use a data cleaning process. This process includes the following steps.

1. Remove sentences which contain words which were written in other languages except for Sinhala.
2. Remove incomplete sentences
3. Remove data which is not related to the domain

Since there was no any tool implemented to support the above-mentioned cleaning steps in the Sinhala language, we were conducted manually.

3.2.3. Data Preprocessing

In order to prepare the data to be used in NLP service, we need to format the collected data set properly. In this research, we have created data in JSON format.



```
{
  "data": [
    {
      "query": "String",
      "entity": {
        "entityType1": "String",
        "entityType2": "String",
        "entityType3": "String",
        "entityType4": "String"
      }
    },
    {
      "query": "String",
      "entity": {
        "entityType1": "String",
        "entityType2": " String",
        "entityType3": "String",
        "entityType4": "String"
      }
    }
  ]
}
```

Figure 2: Structure of the training data file

Figure 2 illustrates the structure of the Training data file we have created by using the collected data set. This file contains the user quarries about the train information, and we have identified the entities inside those files depending on each intent classification group.

We have created a separate folder structure for the training data set. In this folder set, we have created a separate folder for each intent classification group. Initially, we have identified three main intent groups.

1. Greetings
2. Search Train
3. Search Info
4. Ticket Price
5. Other

In each folder we have created for Intent group had contained JSON file which contains data sets formatted according to the structure as mentioned above.

Query section contains the questions we have collected from the Train passengers. Then we have identified the entity set which was necessary for knowledge identification about the user query. All the entities we have identified has assigned to a separate parameter, and the whole set of entities has assigned to a parameter named “entity”.

Since we were unable to find the prior formatted data set, we have to format all the data we have collected manually.

3.2.4. Implementation of Sinhala Chat Bot

This system designs to answer simple questions with the domain of Train Information. Currently, there are several platforms which can be used to implement Chatbots. However, most of them are created only to support the English language.

Implementing a Chatbot can be divided into several steps. Figure 3 illustrates the high-level structure of the Proposed Chatbot system. The entire system has been developed using Java. This system is designed to work on the client-server model. All the controllers, Servers, Data Source connections and Engines are available in the server side. Client-side can connect to the Server side through the Network. In this research, we are mainly focusing on the implementation of the Server side. This chatbot application can implement as a Web service architecture.

The user can be sent their queries as a request to the Server. Hander in the server side read the request from the client side and sends it into the NLP Service for the Intent Classification and Entity identification. NLP Service will identify the knowledge which is required to generate the appropriate response. Response generation will be done by the Response Service together with the Data Service. The Controller has controlled all of this process. It had managed the connection between NLP Service, Response service and the Data service.

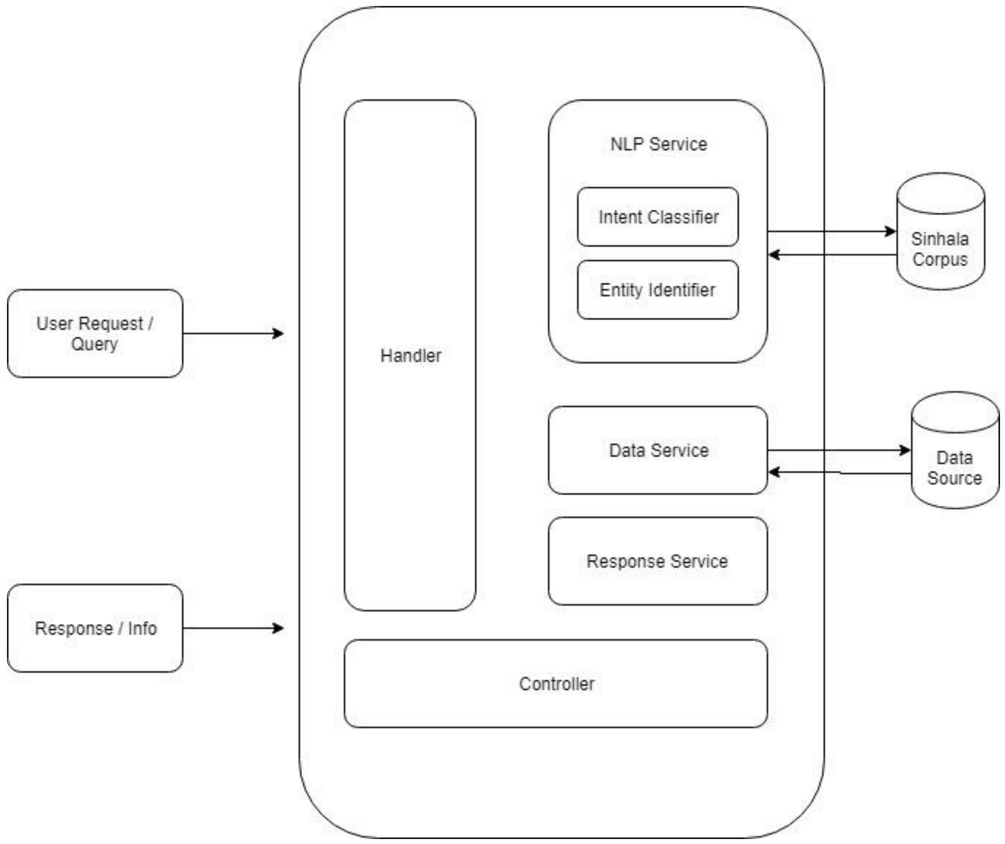


Figure 3: Structure of the Chatbot system

2.2.5. NLP Service

NLP Service is the place where read the user queries from the handler and identifies the appropriate knowledge which required to generating a suitable response. The first step of Knowledge identification is Intent classification.

3.2.5.1. Intent Classifier

The intent classifier has been implemented to identify the intent of the user query or request. There are several third-party libraries available for this purpose, such as IBM Watson, Dialogue Flow, RASA NLU, WEKA Stable java library. Any of these libraries has not been supported for the Sinhala language. Therefore we have decided to implement the Intent classifier from the sketch.

Intent classification is also similar to Text Classification. We have selected the Naïve Bays classification algorithm for the Intent classification.

We have used the Naïve bays text classification system implemented for the Chinese language[32] Modify it in a way to use for the Sinhala language.

The naïve Bays classification algorithm

Naïve bays algorithm is based on Bayes‟ theorem.

Bayes' Theorem

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

- P(A|B) = conditional probability of A given B
- P(A) = priory probability of A
- P(B) = priory probability of B
- P(B|A) = conditional probability of B given A

We can write the equation as follow,

$$P(\,Label \mid feature_1, afeature_2, \ldots feature_n) = \frac{P(\,feature_1, feature_2, \ldots, feature_3 \mid Label\,)\, P(\,Label)}{P(\,feature_1, feature_1, \ldots feature_3\,)}$$

$$Label_{MAP} = \arg \max_{v_j \in Label} P(\,v_j \mid feature_1, feature_2, \ldots feature_n\,)$$

$$Label_{MAP} = \arg \max_{v_j \in Label} \frac{P(feature_1, feature_2, \ldots, feature_n \mid v_j)\, P(\,v_i\,)}{p(\,feature_1, feature_2, \ldots, feature_n\,)}$$

$$Label_{MAP} = \arg \max_{v_j \in Label} P(\,v_j \mid feature_1, feature_2, \ldots feature_n\,) P(\,v_i\,)$$

Naïve Bays assumption:

$$P(feature_1, feature_2, \ldots, feature_n \mid v_j) = \prod_i P(\,feature_i \mid v_j\,)$$

Which gives the Naïve Bays classifier

$$Label_{NB} = \arg \max_{v_j \in Label} P(\, v_i\,) \prod_i P(\, feature_i \,|v_j\,)$$

Figure 4: Equation of Predicting the label

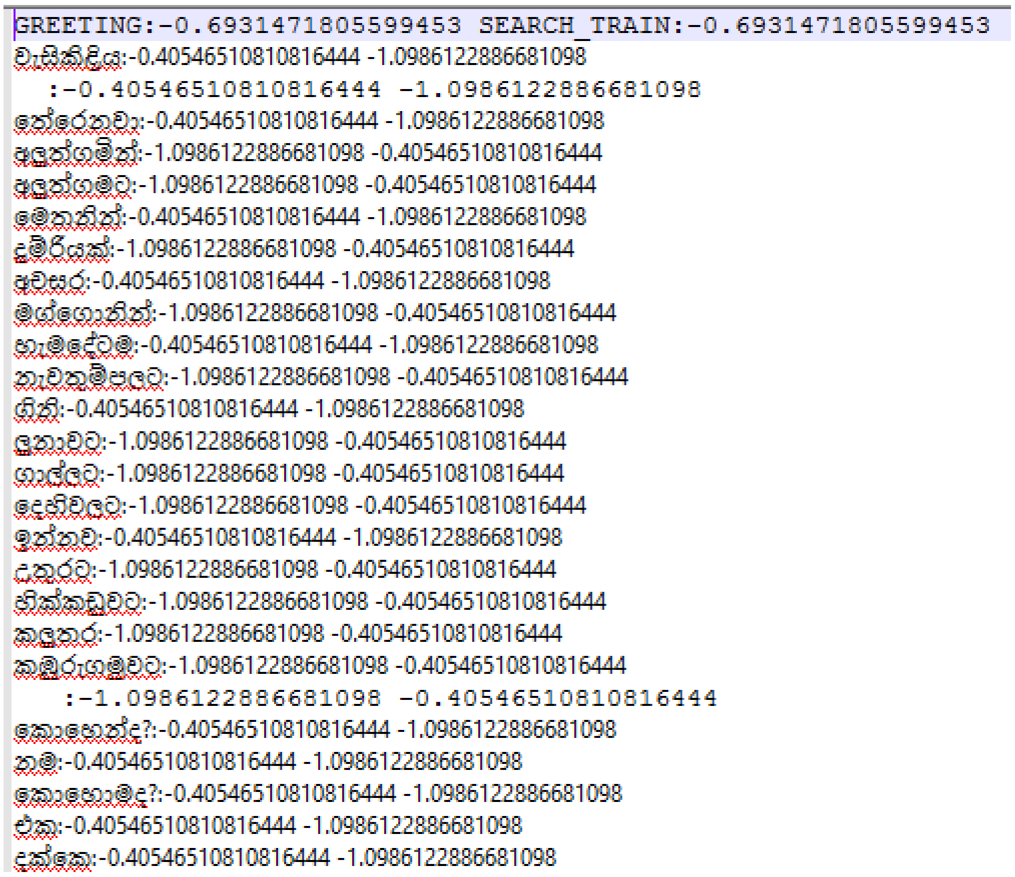
We use his classification theorem for text classification[33] [34]. Because it is pretty fast, it can handle a large number of features (words), and it is effective[35].

Training of Intent Classifier

Next step of this research was to Train the Intent classifier and create a model to use for Intent classification. For this training process, we were using data collected in section 3.2.1 and formatted in section 3.2.2 and 3.2.3

The Intent Classifier we have implemented had been created a data objects from the training data set we have created in JSON format. Then part of speech is used to filter noise words. We have removed stop words, and unwanted symbols from the data set. If the word segmentation does not provide part-of-speech tagging Function, the default can be all labelled n, and then use other filtering methods. At the end of the training, the model file is generated into a separate location in the project.

Figure 7 is a sample of a model file generated by the Naïve Bayes classifier which was trained for two intent classification groups such as Greeting and Search Train



```
GREETING:-0.6931471805599453 SEARCH_TRAIN:-0.6931471805599453
වැසිකිළිය:-0.40546510810816444 -1.0986122886681098
  :-0.40546510810816444 -1.0986122886681098
තේරෙනවා:-0.40546510810816444 -1.0986122886681098
අලුත්ගමින්:-1.0986122886681098 -0.40546510810816444
අලුත්ගමට:-1.0986122886681098 -0.40546510810816444
මෙතනින්:-0.40546510810816444 -1.0986122886681098
දුම්රියක්:-1.0986122886681098 -0.40546510810816444
අවසර:-0.40546510810816444 -1.0986122886681098
මග්ගොනින්:-1.0986122886681098 -0.40546510810816444
හැමදේටම:-0.40546510810816444 -1.0986122886681098
නැවතුම්පලට:-1.0986122886681098 -0.40546510810816444
ගිනි:-0.40546510810816444 -1.0986122886681098
ලුනාවට:-1.0986122886681098 -0.40546510810816444
ගාල්ලට:-1.0986122886681098 -0.40546510810816444
දෙහිවලට:-1.0986122886681098 -0.40546510810816444
ඉන්නව:-0.40546510810816444 -1.0986122886681098
උතුරට:-1.0986122886681098 -0.40546510810816444
හික්කඩුවට:-1.0986122886681098 -0.40546510810816444
කලුතර:-1.0986122886681098 -0.40546510810816444
කඹුරුගමුවට:-1.0986122886681098 -0.40546510810816444
   :-1.0986122886681098 -0.40546510810816444
කොහෙන්ද?:-0.40546510810816444 -1.0986122886681098
නම:-0.40546510810816444 -1.0986122886681098
කොහොමද?:-0.40546510810816444 -1.0986122886681098
එක:-0.40546510810816444 -1.0986122886681098
දක්කෙ:-0.40546510810816444 -1.0986122886681098
```

Figure 5: Structure of a model file generated from Naive Bayes classifier

Intent Classification

After training the Intent classifier, the generated model can be used for intent classification. We have used the equation mentioned in Figure 6 to predict the Intent group. Valued required for the calculation was taken from the generated model file.

By using this process, we were able to categorise the queries sending by the user according to their intent. This categorisation has used in the next steps to identify the knowledge and also for generating an appropriate response.

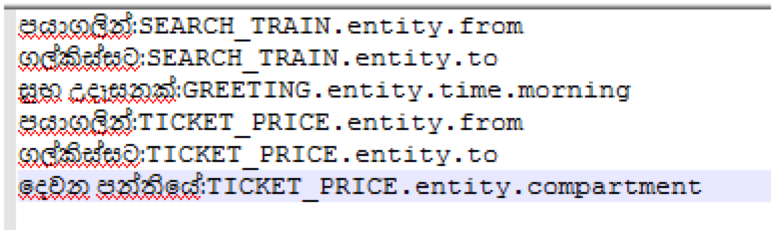
3.2.5.2. Entity Identification

AI chatbots owe their skill to NLP (natural language processing) that examines human speech and makes use of knowledge about sentence structure as well as the machine-learned pattern recognition. Chatbots can connect human speech with the intent they were equipped with, and that helps them to find answers and deliver a speech that sounds as humans produced it.

Entities (slots) are data buckets used in conversations. Entities are the fields, data, or words the developer designates necessary for the chatbot to complete a task: a date, time, person, location, description of an item or a product, or any number of other designations.

In order to identify the knowledge inside the user queries, it is necessary to implement an Entity Identifier during the implementation of Chatbots. In this research, we have implemented a simple Entity Identifier. Challenge was to identify the Sinhala words and extract the entities from the received sentences.

We have generated an entity model by using the training data.



```
පයාගලින්:SEARCH_TRAIN.entity.from
ගල්කිස්සට:SEARCH_TRAIN.entity.to
සුබ උදෑසනක්:GREETING.entity.time.morning
පයාගලින්:TICKET_PRICE.entity.from
ගල්කිස්සට:TICKET_PRICE.entity.to
දෙවන පන්තියේ:TICKET_PRICE.entity.compartment
```

Figure 6: Entity Model

A sample of the generated Entity model file is in Figure 8.

When a client sends a request with the user query, what we are doing first is to identify the Intent of the query. Then it was sent to the Entity Identifier. Entity identifier chooses the Entity model created for the corresponding Intent. Next step of the Entity identifier is to read the Entities in the model one by one and check which entity is available in the user query/ info.

For each intent, we have created the criteria object, which can store all the necessary knowledge for response generation.

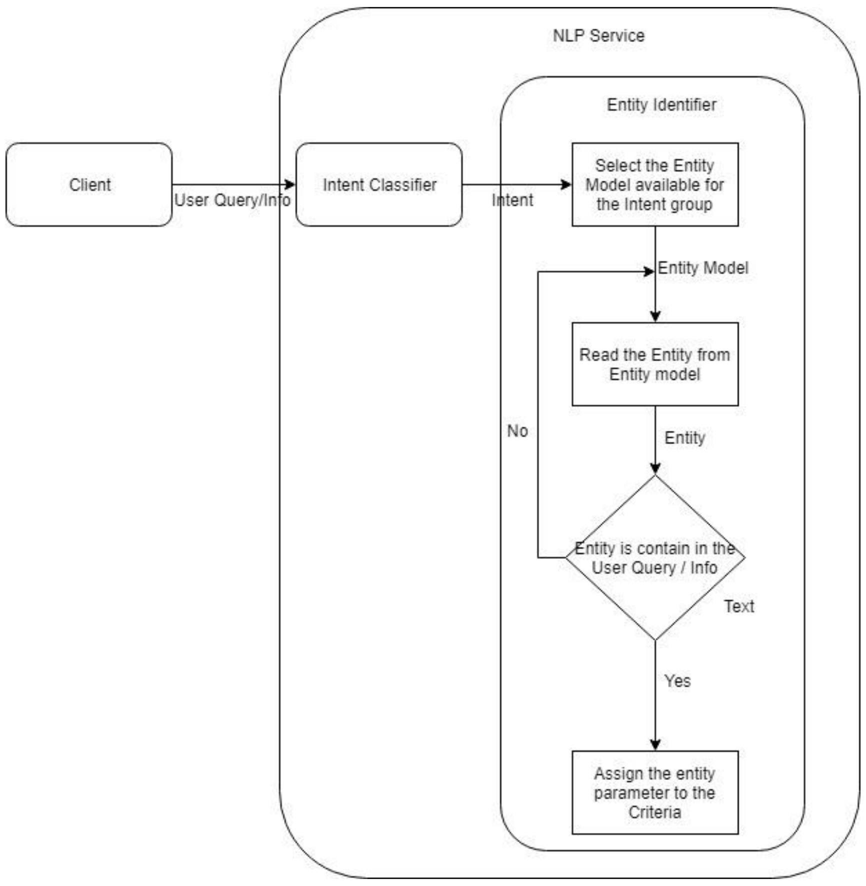

NLP Service
Entity Identifier
Client
User Query/Info
Intent Classifier
Intent
Select the Entity Model available for the Intent group
Entity Model
Read the Entity from Entity model
No
Entity
Entity is contain in the User Query / Info
Text
Yes
Assign the entity parameter to the Criteria

Figure 7: Structure of the Entity Identifier

3.2.5.3. Response Generation

Once the chatbot understands the user's message, the next step is to generate a response. In this research, we have used the Rule-based approach to generate the response. There are two types of responses generated by a chatbot.

1. The answer to the user with an appropriate message
2. Asking for some more data from the User.

Intent and identified entities parsed to the Response generator as inputs. By using those entities, Entity identifier has created a Criteria object, and it will be passed to the response generator. Separate ruler sets have implemented for each intent group, and according to the rules, the response will be created.

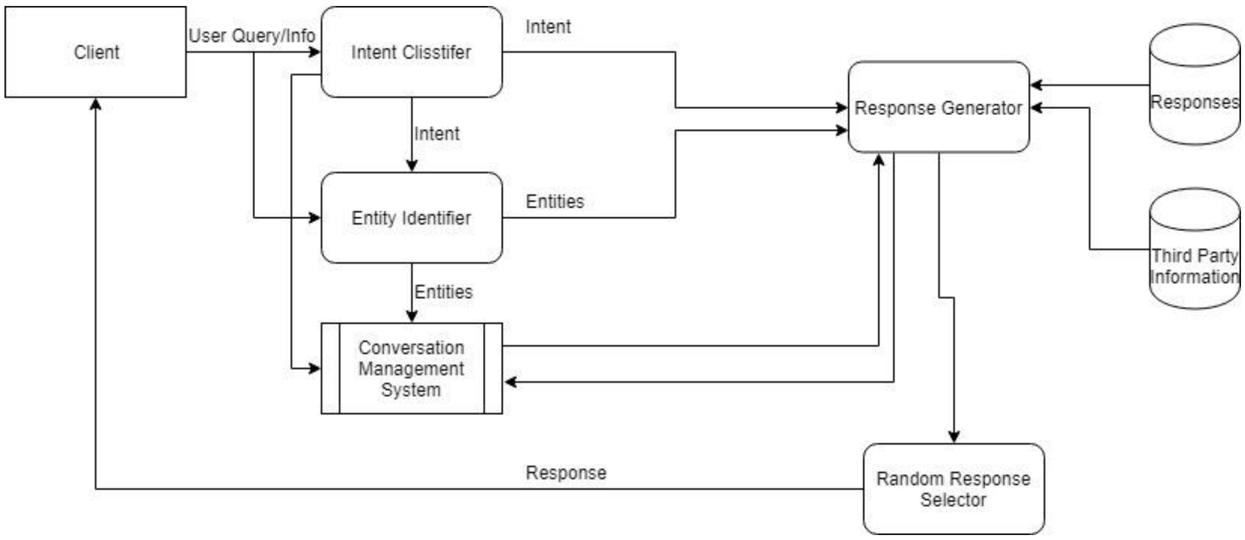


Figure 8: Process of Response generator

Figure 8 illustrates the process of the response generator including inputs and outputs.

When generating the response, we had to consider about few important points in order to keep the chatbot similar to the human-human conversation. In order to archive this goal, we have added some functionality to the system proposed with this research. Random response selector and Conversation Management are implemented to add this functionality to the system.

After getting user queries with its intent and entities, Response generator will check all the necessary information are available to generate the response according to the intent. If it has no sufficient information, Response generator will generate a separate response with asking missing information. When the system received missing information, it is required to store the existing information. For this purpose, we have implemented a conversation management system to store all information users sending to the system, and also it will store all the response system has given. Until the conversation session gets killed, these data will be stored in an object level storage, and at the end of the conversation, all the user queries and responses will be stored in a database.

In this research, we were able to implement the chatbot which can give random responses for the queries in the same intent. After receiving the intent and the entities, the response generator sent information through a set of rules and identifies the type of response which needs to send to the user. We have created a set of response templates for the response types. Since this was implemented in the Sinhala language, we have created those templates manually and store them separately for each response type. When the response generator identified the response type needs to be generated, it will calculate the random number and use this random number to select a response template. Then the system has added the appropriate values to the response template and sends the response to the client.

During the process of response generation, the system needs to be retrieving some data from third parties such as in order to answer the queries like Train time table information; the system needs to get data from the Sri Lanka railway department timetable. In order to retrieve that information, we have created a sample database which store sample data set of the Train time table and Train ticket prices.

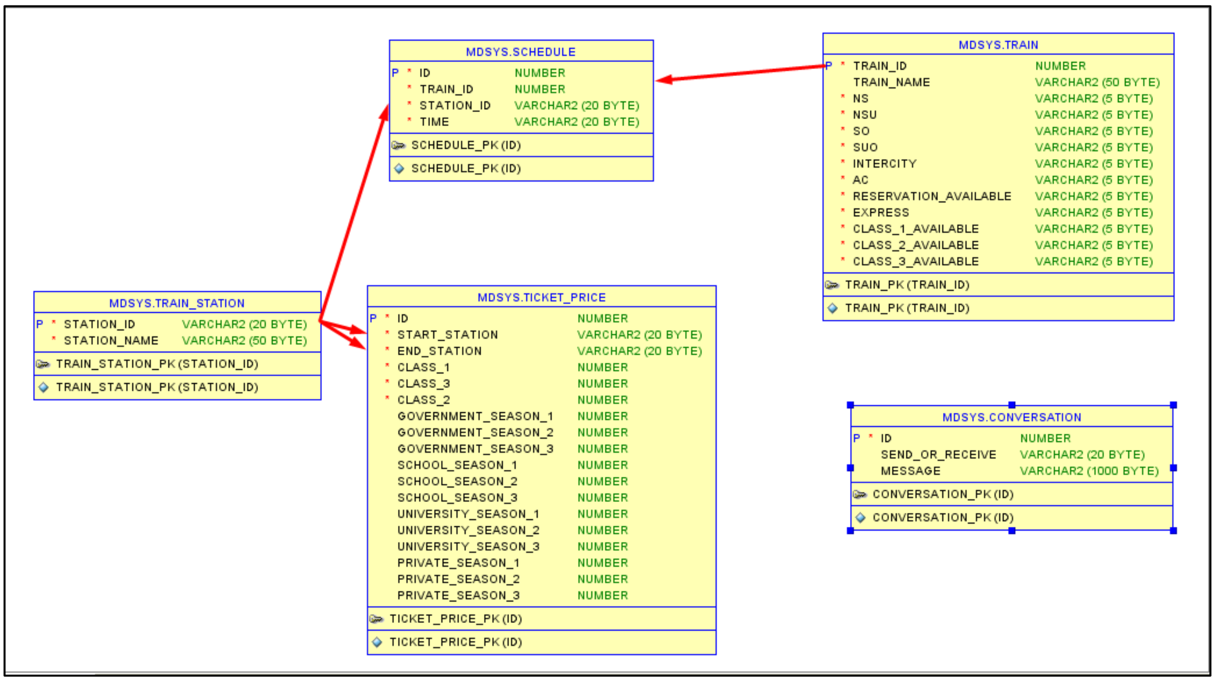

MDSYS.SCHEDULE
P * ID NUMBER
* TRAIN_ID NUMBER
* STATION_ID VARCHAR2 (20 BYTE)
* TIME VARCHAR2 (20 BYTE)
SCHEDULE_PK (ID)
SCHEDULE_PK (ID)
MDSYS.TRAIN
P * TRAIN_ID NUMBER
TRAIN_NAME VARCHAR2 (50 BYTE)
* NS VARCHAR2 (5 BYTE)
* NSU VARCHAR2 (5 BYTE)
* SO VARCHAR2 (5 BYTE)
* SUO VARCHAR2 (5 BYTE)
* INTERCITY VARCHAR2 (5 BYTE)
* AC VARCHAR2 (5 BYTE)
* RESERVATION_AVAILABLE VARCHAR2 (5 BYTE)
* EXPRESS VARCHAR2 (5 BYTE)
* CLASS_1_AVAILABLE VARCHAR2 (5 BYTE)
* CLASS_2_AVAILABLE VARCHAR2 (5 BYTE)
* CLASS_3_AVAILABLE VARCHAR2 (5 BYTE)
TRAIN_PK (TRAIN_ID)
TRAIN_PK (TRAIN_ID)
MDSYS.TRAIN_STATION
P * STATION_ID VARCHAR2 (20 BYTE)
* STATION_NAME VARCHAR2 (50 BYTE)
TRAIN_STATION_PK (STATION_ID)
TRAIN_STATION_PK (STATION_ID)
MDSYS.TICKET_PRICE
P * ID NUMBER
* START_STATION VARCHAR2 (20 BYTE)
* END_STATION VARCHAR2 (20 BYTE)
* CLASS_1 NUMBER
* CLASS_3 NUMBER
* CLASS_2 NUMBER
GOVERNMENT_SEASON_1 NUMBER
GOVERNMENT_SEASON_2 NUMBER
GOVERNMENT_SEASON_3 NUMBER
SCHOOL_SEASON_1 NUMBER
SCHOOL_SEASON_2 NUMBER
SCHOOL_SEASON_3 NUMBER
UNIVERSITY_SEASON_1 NUMBER
UNIVERSITY_SEASON_2 NUMBER
UNIVERSITY_SEASON_3 NUMBER
PRIVATE_SEASON_1 NUMBER
PRIVATE_SEASON_2 NUMBER
PRIVATE_SEASON_3 NUMBER
TICKET_PRICE_PK (ID)
TICKET_PRICE_PK (ID)
MDSYS.CONVERSATION
P * ID NUMBER
SEND_OR_RECEIVE VARCHAR2 (20 BYTE)
MESSAGE VARCHAR2 (1000 BYTE)
CONVERSATION_PK (ID)
CONVERSATION_PK (ID)

Figure 9: ER Diagram of the Database

Chapter Four

Results

The result of this research project, we were able to implement a simple chatbot system which was able to answer simple questions about Train information in the Sinhala language.

In this research, we intended to propose a framework to create domain-specific chatbot supported for the Sinhala language. As the first step, we have created a chatbot for Train information using our framework.

4.1. Collected Data set

According to the method describe in section 3.2.1, we collect 7293 clean Sinhala queries in Train Information domain. It includes Greetings, queries about the train time table, train availability, ticket prices, reservations. From the above data set, we have to kept 1/10 data for evaluation and other data used for Training

4.2. The result of Intent Classifier

We were able to implement an intent classifier using naïve Bayes classifier to support for the Sinhala language texts.

Initially, we have developed the naïve Bayes classifier for four intents. Moreover, for four entities classifier gives 84.14% accuracy. This accuracy was calculated using 306 test queries representing all four intent categories. We assumed that when the number of intents is increasing, accuracy will also get reduced but with using more training data to train the model will increase the accuracy of the classifier.

We have trained the system using 3819 texts in SEARCH_TRAIN intent, 70 texts in GREETING intent, 7175 texts in SEARCH_TICKET_PRICE intent and 40 texts in OTHER intents. The result of this training of Intent classifier is to generate a model contain a word with their frequencies probability for each intent. Figure 5 is an image of a sample model file generated from the Intent classifier

We have used this model to identify the intent of the user query or information.

4.3. The result of the Entity Identifier

Entity identifier will also create a model during the training of the model. This model contains entities that can be identified in the training data set. Separate models were created for all intents. Figure 6 is an image of a sample model generated for SEARCH_TRAIN intent. It contains the value and the parameter of the criteria. By using this model, the system was able to identify the entities contains in the queries sent by the client, and it will create a criteria object which is related to the particular intent and assign the values to the object.

Entity Extractor model was trained using 641 queries from SEARCH_TRAIN intent, 337 queries from SEARCH_TICKET_PRICE intent and 35 queries from GREETING intent.

Accuaracy of the Intent identifier was calculated using a test data set which contains 298 test data representing all four intents. Implemented entity identifier gives 79.94%.accuracy for four intent groups which was tested using above mentioned test data set.

4.4. Results of the Chatbot

In this research, we were not focusing on implementing a client-side or a user interface. The main focus of our research was to implement a framework which can be used to implement a Sinhala domain specific chatbot.

As a result of this approach, we were able to implement a chatbot which can answer simple queries related to the Train information domain.

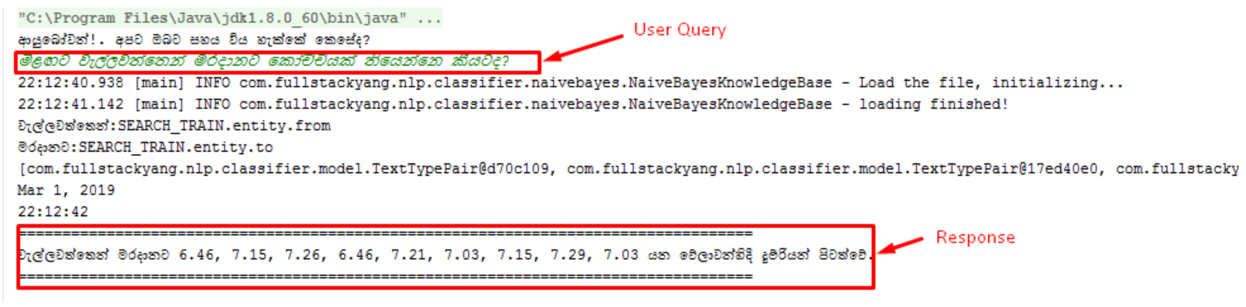

```
"C:\Program Files\Java\jdk1.8.0_60\bin\java" ...
ආයුබෝවන්!. අපට ඔබට සහය විය හැක්කේ කෙසේද?
මීළඟට වැල්ලවත්තෙන් මරදානට කෝච්චියක් තියෙන්නෙ කීයටද?
22:12:40.938 [main] INFO com.fullstackyang.nlp.classifier.naivebayes.NaiveBayesKnowledgeBase - Load the file, initializing...
22:12:41.142 [main] INFO com.fullstackyang.nlp.classifier.naivebayes.NaiveBayesKnowledgeBase - loading finished!
වැල්ලවත්තෙන්:SEARCH_TRAIN.entity.from
මරදානට:SEARCH_TRAIN.entity.to
[com.fullstackyang.nlp.classifier.model.TextTypePair@d70c109, com.fullstackyang.nlp.classifier.model.TextTypePair@17ed40e0, com.fullstacky
Mar 1, 2019
22:12:42
===================================================================================
වැල්ලවත්තෙන් මරදානට 6.46, 7.15, 7.26, 6.46, 7.21, 7.03, 7.15, 7.29, 7.03 යන වේලාවන්හිදී දුම්රියන් පිටත්වේ.
===================================================================================
```

Figure 10: Response generated by the system

Since we have not implemented a user interface, we have tested the project in the terminal as a basic java project. This system was able to generate responses in random formats.

When considering the accuracy of the system, the ultimate accuracy of the chatbot sytem is depending on the accuracy of the Intent identification and Entity extraction of the user queries. By using the accuracy calculated for the intent Identifier and the Entity extractor following confution metrix can be generated.

		Naïve Bays intent classifier	
		Positive	Negative
Entity Extractor	Positive	231.0	16.0
	Negative	26.0	25.0

Above values calculated over 298 test data. Based on the table, the ultimate accuracy of the intent classifier and the Entity extractor is 77.52%.

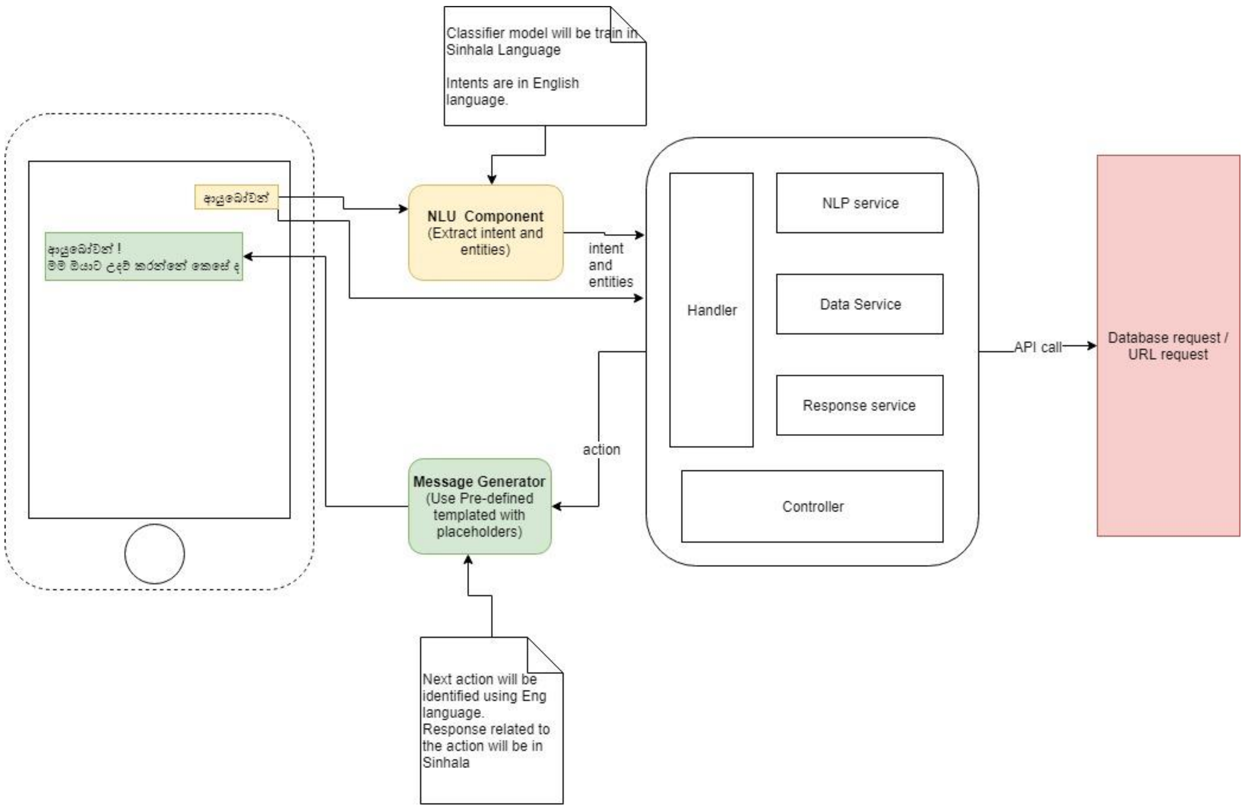
Chapter Five

Validation and Evaluation

Evaluation of the developed implemented conversational interface can be obtained by defining a series of user statements and measuring to which level the conversational assistant accurately identifies the user's intentions. The evaluation criteria can be further extended to measure whether the information assistant succeeds in providing relevant information in order to satisfy user Requests.

Different mechanisms are used to evaluate Spoken Dialogue Systems (SLDs). Turing test can be used to evaluate the chatbot systems. A Turing test, which evaluates the ability of the machine to fool people that they are talking to a human. In essence, judges are allowed a short chat (10 to 15 minutes) with the chatbot, and asked to rank them in terms of "naturalness". In order to use the Turing test to evaluate the chatbot is not a very simple task. Setup a Turing test is a huge task, and it is a very costly task. Therefore we have decided not to use the Turing test for the Evaluation.

Since we have implemented the Chatbot from the sketch, we have decided to evaluate each step separately and finally carrying out manual testing for the overall chat bot system.



Classifier model will be train in Sinhala Language
Intents are in English language.
ආයුබෝවන්
ආයුබෝවන් !
මම ඔයාට උදව් කරන්නේ කෙසේ ද
NLU Component (Extract intent and entities)
intent and entities
Handler
NLP service
Data Service
Response service
Controller
API call
Database request / URL request
action
Message Generator (Use Pre-defined templated with placeholders)
Next action will be identified using Eng language. Response related to the action will be in Sinhala

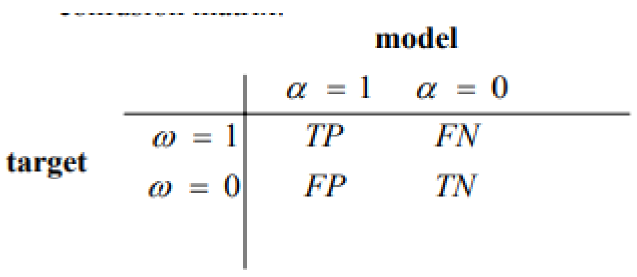
Figure 11: Chatbot response generation process

According to the above diagram of the system, we were able to identify two places which can evaluate, NLU component (Entity and Intent extractor) and a Message generator.

5.1. Evaluate Entity and Intent extractor

For the implementation of Entity and Intent extractor, we have used a Naive Bayes classification algorithm, which is a well-known test classification algorithm.
We have grouped the collected data into two groups as training data and testing data. We can train the naive bays classifier using trading data set and generate the model for the intent classification. Then we can use the testing data set for the evaluation. We can calculate the accuracy of the algorithm which has been trained using training data.

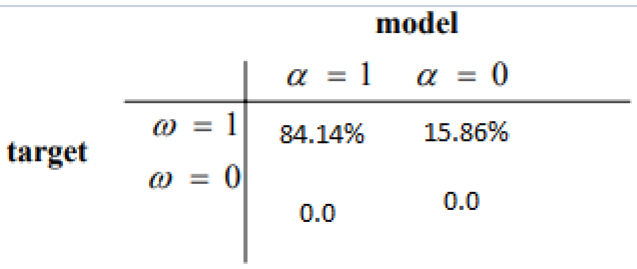
For any data set we used to test the model we can build a confusion matrix:
– Counts of examples with:
– Class label ω j that is classified with a label α i



		model	
		$\alpha = 1$	$\alpha = 0$
target	$\omega = 1$	TP	FN
	$\omega = 0$	FP	TN

Figure 12: Confusion Matrix

TP: True positive (hit)
FP: False positive (false alarm)
TN: True negative (correct rejection)
FN: False negative (a miss)



		model	
		$\alpha = 1$	$\alpha = 0$
target	$\omega = 1$	84.14%	15.86%
	$\omega = 0$	0.0	0.0

Figure 13: Confusion Matrix of the Intent Classifier

Figure 13 is the confusion Matrix of the intent classifier for two intents. We assumed that the accuracy would be reduced when the number of intent get increased. This calculation was done using java unit testing.

5.2. Evaluate Message Generator

In order to evaluate the Message generator, we used manually setup testing data set as similar to the above step. The difference of this step is that the message generator can dynamically change the reply of the message. Therefore we had to consider this dynamic behaviour during the calculation of the accuracy of the message generator.
The result of this step also similar to the intent classifier.

5.3. Manual Testing of Chatbot

Since it was hard to set up a Turing test, we have decided to test the overall output of the chatbot system manually.
We have used a set of question which can be asked from the chatbot and test whether it is providing the correct information.
We mainly focused on identify the user query and the extraction of the knowledge from the user queries.
A set of 298 queries is prepared as a test set representing all the 4 intents and tested against the classifier and the entity identifier. Based on the results obtained by the intent classifier the accuracy is determined.

		Naïve Bays classifier	
		Positive	Negative
Intent Identifier	Positive	True Positive	True Negative
	Negative	False Positive	False Negative

Based on the table, the ultimate accuracy can be calculated.

During the above process we have created a test data file which has a similar structure as the training data set created for the Intent extractor.(Figure 2: Structure of the training data file)

We have created a unit test to compare the intent identified by the Intent identifier and the intents defiened in the test data set.

Chapter Six

Conclusion and Future Work

In this section, we describe the main contribution of our research and possible future improvements.

In this research, we were able to create a framework that can be used for implementing domain specific Chatbot systems for the Sinhala language. By using this framework, we have implemented a Sinhala chatbot for Train information domain.

In this approach, we have not used any of the third party libraries for any NLP purpose. The reason is that most of the libraries do not support the Sinhala language or Unicode. We have used the neural network approaches for NPL service implementation. Therefore it was not required any knowledge about Sinhala grammar. We used only stop word list, corpus and prefixes and suffixes only. These data were taken from the Language Technology Research Library(LTRL) [36] of UCSC.

The main part of this implementation is the NLP Service. In addition to the NLP service, we have added Data Service, Controller, Handler and Conversation Management sections. Data service is used for handling database connections, data read and write to the Database. Handlers and controllers are implemented to manage the flow of the system and to manage the relationship between separate sections in this framework. Conversation Management was implemented to store the conversation detail in an object level storage and also at the end of the conversation it will store the requests and responses in the Database.

NLP Service contains three sections, Intent classifier, Entity identifier and Response generator. The intent classifier was implemented using text classification methodology. We have used the naïve Bayes algorithm for this implementation. Since it was supervise learning it was implemented to work as two parts, Learning and Classification. During the Training process of naïve Bayes classification, it generated a model file including all the information, and it is used for the classification process.

Entity identifier was also implemented under Neural Network supervised learning. During the process of learning of Entity identifier, it generated a model, and this model was used for knowledge identification of the user query.

Response generator is the place where the appropriate response was generated. It will take the inputs from intent classifier and entity recogniser and identified the knowledge inside the user query. Then it was generated the appropriate response for the user query. Response generator was able to generate answers in random patterns, and this functionality was implemented by using a random response selector.

We have proposed to implement this system as a client-server approach. However, in this research, we have implemented only the server side. We were not focusing on the implementation of the user interface because our main focus was to proposed a framework that can be used for implementation of Domain-specific Chatbot systems for the Sinhala language.

This system was able to answer simple questions related to the Train information such as Search train time table, Search train ticket prices, reservation methods. Our main focus was to correctly identify the intent and the knowledge inside the user query which sending in the Sinhala language.

Further work of this research includes expanding this chatbot to answer more complex questions. It required training the system with more data in different intents. When the number of intents is getting increased, we assumed that the accuracy of the intent classifier would be reduced and we need to improve the preprocessing part of the intent classifier as future work.

This Chatbot system was implemented for Train information domain, and this system and the code base will not be suitable for other domains. Therefore as a future work of this research, we hope to implement a chatbot system using the proposed framework from this project to use generally for any domain.

We can further improve the intent classifier we have implemented using the naïve Bayes algorithm as a general intent classifier to train with Sinhala text in any intent in any domain.

In this project, we have more focusing on implementing the Chatbot to identify the intent and the knowledge, and we did not consider the user interface and the accuracy of the response. Therefore implementing this framework as a REST full service and implementation of a User interface for this system will be future work. Moreover, also implement the response generator with real data to generate a more accurate response will be future work.

References

[1] J. Weizenbaum, “ELIZA — A Computer Program For the Study of Natural Language Communication Between Man And Machine,” *Commun. ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[2] “Using dialogue corpora to train a chatbot Bayan Abu Shawar (bshawar@comp.leeds.ac.uk) and Eric Atwell (eric@comp.leeds.ac.uk) School of Computing , University of Leeds , Leeds LS2 9JT England,” pp. 681–690.

[3] “Chatbot A.L.I.C.E. (A.L.I.C.E. A.I Foundation).” [Online]. Available: https://www.chatbots.org/chatbot/a.l.i.c.e.

[4] A. Tiwari, R. Talekar, and P. S. M. Patil, “College Information Chat Bot System,” vol. 5, no. 2, pp. 131–137, 2017.

[5] B. Hettige and A. S. Karunananda, “First Sinhala Chatbot in action,” no. September, pp. 4–10, 2006.

[6] B. A. Shawar and E. S. Atwell, “Using corpora in machine-learning chatbot systems,” *Int. J. Corpus Linguist.*, vol. 10, no. 4, pp. 489–516, 2005.

[7] E. S. Abu Shawar, B. A., & Atwell, “An Arabic Chatbot Giving Answers from the Qur’an,” *Proc. TALN04 XI Conf. sur le Trait. Autom. des Langues Nat. Vol. 2*, pp. 197–202, 2004.

[8] Chris McGrath, “Chatbot Vocabulary_ 10 Chatbot Terms You Need to Know.” [Online]. Available: https://chatbotsmagazine.com/chatbot-vocabulary-10-chatbot-terms-you-need-to-know-3911b1ef31b4.

[9] Dmitry Persiyanov, “Chatbots with Machine Learning_ Building Neural Conversational Agents.” [Online]. Available: https://blog.statsbot.co/chatbots-machine-learning-e83698b1a91e.

[10] U. Furbach, I. Glöckner, and B. Pelzer, “An application of automated reasoning in natural language question answering,” *AI Commun.*, vol. 23, no. 2–3, pp. 241–265, 2010.

[11] I. Gl, “University of Hagen at CLEF 2008 : Answer Validation Exercise,” vol. 1, 2008.

[12] I. Glockner, S. Hartrumpf, and J. Leveling, “Logical Validation, Answer Merging and Witness Selection - A Study in Multi- Stream Question Answering.,” no. January, 2007.

[13] Marijana Mrkalj, “Chatbots and AI: The Key Event Tech Trends for 2018,” *Chatbot Magazine*, 2018. [Online]. Available: https://chatbotsmagazine.com/chatbots-and-ai-the-key-event-tech-trends-for-2018-7b45cbc723a.

[14] P. Marupaka, “Chatbots (of) the future! – The Startup – Medium,” *medium.com*. [Online]. Available: https://medium.com/swlh/chatbots-of-the-future-86b5bf762bb4.

[15] P. Surmenok, “Natural Language Pipeline for Chatbots – Pavel Surmenok – Medium,” *medium.com*. [Online]. Available: https://medium.com/@surmenok/natural-language-pipeline-for-chatbots-897bda41482.

[16] M. Pantic, R. Zwitserloot, and R. J. Grootjans, “Teaching Introductory Artificial Intelligence Using a Simple Agent Framework,” *Ieee Trans. Educ.*, vol. 48, no. 3, pp. 382–390, 2005.

[17] Google, “Dialogflow,” *dialogflow*. [Online]. Available: https://dialogflow.com/.

[18] “IBM Watson _ IBM,” *IBM*. [Online]. Available: https://www.ibm.com/watson/.

[19] M. Lawnboy, “Building a Chatbot Using Rasa Stack_ Intro and Tips,” *hackernoon*. [Online]. Available: https://hackernoon.com/building-a-chatbot-using-rasa-stack-intro-and-tips-c6d1057d8536.

[20] Rasa Technologies, “Rasa: Open source conversational AI,” 2018. [Online]. Available: https://rasa.com/.

[21] E. Haller and T. Rebedea, “Designing a chat-bot that simulates an historical figure,” *Proc. - 19th Int. Conf. Control Syst. Comput. Sci. CSCS 2013*, no. May 2013, pp. 582–589, 2013.

[22] I. V. Serban *et al.*, “A Deep Reinforcement Learning Chatbot,” pp. 1–40, 2017.

[23] A. Vegesna, P. Jain, and D. Porwal, “Ontology-based Chatbot (For E-commerce Website),” *Int. J. Comput. Appl.*, vol. 179, no. 14, pp. 51–55, 2018.

[24] C. Xing *et al.*, “Topic Aware Neural Response Generation,” pp. 3351–3357, 2016.

[25] B. A. Shawar and E. Atwell, “Different measurements metrics to evaluate a chatbot system,” *Proc. Work. Bridg. Gap Acad. Ind. Res. Dialog Technol.*, no. April, pp. 89–96, 2007.

[26] R. Crutzen *et al.*, “An Artificially Intelligent Chat Agent That Answers Adolescents ’ Questions Related to Sex , Drugs , and Alcohol : An Exploratory Study,” 2010.

[27] F. Raby, “Evaluating the Language Resources of Chatbots for Their Potential in English As a Second Language,” vol. 19, no. 2, pp. 21–38, 2007.

[28] N. M. Radziwill and M. C. Benton, “Evaluating Quality of Chatbots and Intelligent Conversational Agents,” 2017.

[29] B. AbuShawar and E. Atwell, “Usefulness, localizability, humanness, and language-benefit: additional evaluation criteria for natural language dialogue systems,” *Int. J. Speech Technol.*, vol. 19, no. 2, pp. 373–383, 2016.

[30] B. Hettige and A. S. Karunananda, “Transliteration system for English to Sinhala machine translation,” *ICIIS 2007 - 2nd Int. Conf. Ind. Inf. Syst. 2007, Conf. Proc.*, no. October 2015, pp. 209–214, 2007.

[31] R. Weerasinghe, A. Wasala, and K. Gamage, “A Rule Based Syllabification Algorithm for Sinhala,” vol. 3651, no. May 2014, 2005.

[32] “GitHub - fullstackyang_article-classifier_ 基于朴素贝叶斯实现的一款微信公众号文章分类器,” *Github*. [Online]. Available: https://github.com/fullstackyang/article-classifier.

[33] J. Spencer and G. Uchyigit, “Sentimentor: Sentiment analysis of twitter data,” *CEUR Workshop Proc.*, vol. 917, pp. 56–66, 2012.

[34] Jiawen Le, “Master Thesis 2014 Cross-Domain Investigations of User Evaluations under the Multi-cultural Backgrounds,” 2014.

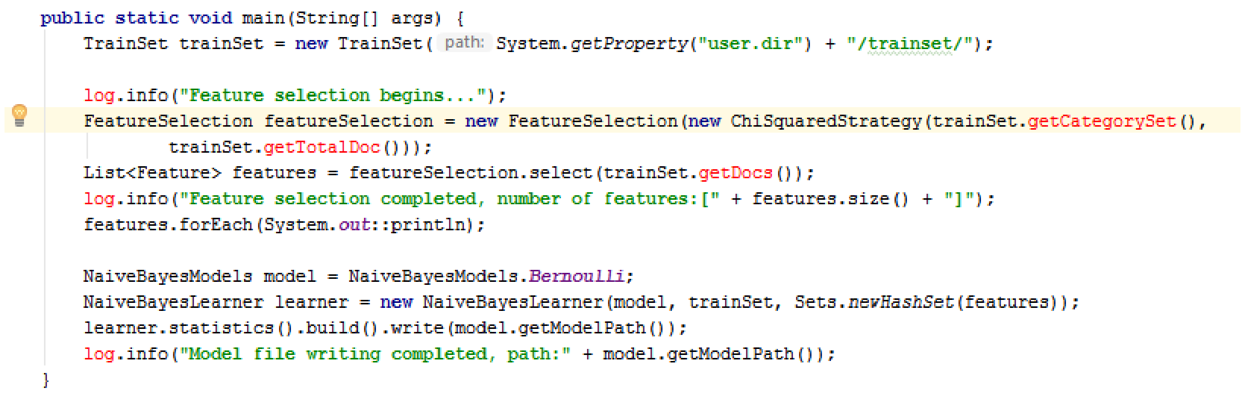
[35] A. Pak and P. Paroubek, “Twitter as a Corpus for Sentiment Analysis and Opinion Mining,” pp. 1320–1326.

[36] “Language Technology Research Lab.” [Online]. Available: http://ltrl.ucsc.lk/.

Appendix

Intent Identification

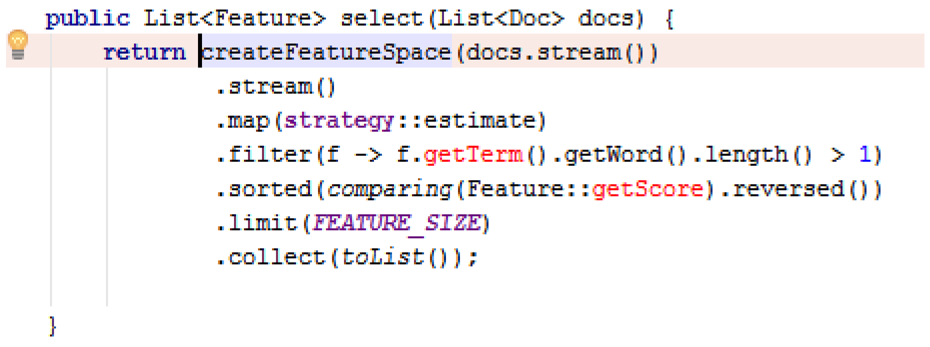
Naïve Bayes Algorithm - Training



```
public static void main(String[] args) {
    TrainSet trainSet = new TrainSet( path: System.getProperty("user.dir") + "/trainset/");

    log.info("Feature selection begins...");
    FeatureSelection featureSelection = new FeatureSelection(new ChiSquaredStrategy(trainSet.getCategorySet(),
            trainSet.getTotalDoc()));
    List<Feature> features = featureSelection.select(trainSet.getDocs());
    log.info("Feature selection completed, number of features:[" + features.size() + "]");
    features.forEach(System.out::println);

    NaiveBayesModels model = NaiveBayesModels.Bernoulli;
    NaiveBayesLearner learner = new NaiveBayesLearner(model, trainSet, Sets.newHashSet(features));
    learner.statistics().build().write(model.getModelPath());
    log.info("Model file writing completed, path:" + model.getModelPath());
}
```

Figure 14: Train the Intent classifier

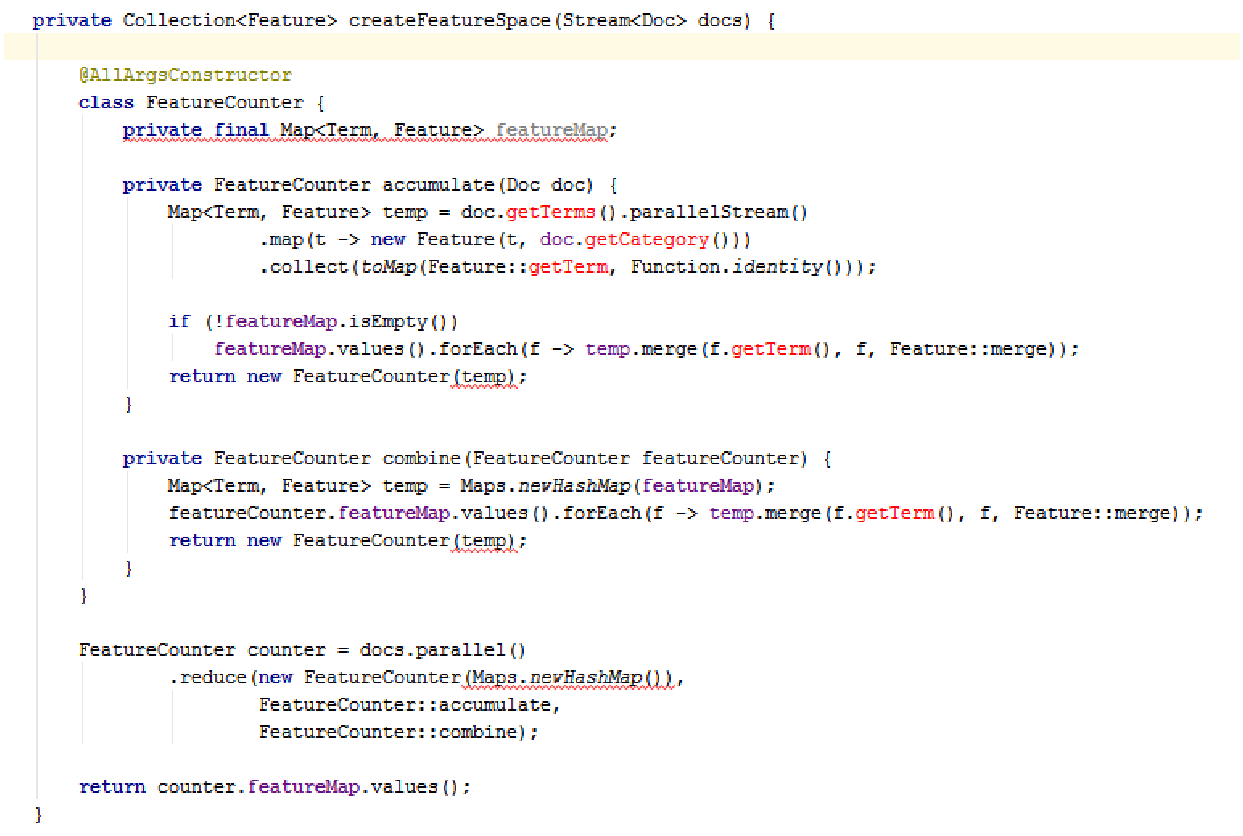
Feature Selection



```
public List<Feature> select(List<Doc> docs) {
    return createFeatureSpace(docs.stream())
            .stream()
            .map(strategy::estimate)
            .filter(f -> f.getTerm().getWord().length() > 1)
            .sorted(comparing(Feature::getScore).reversed())
            .limit(FEATURE_SIZE)
            .collect(toList());

}
```

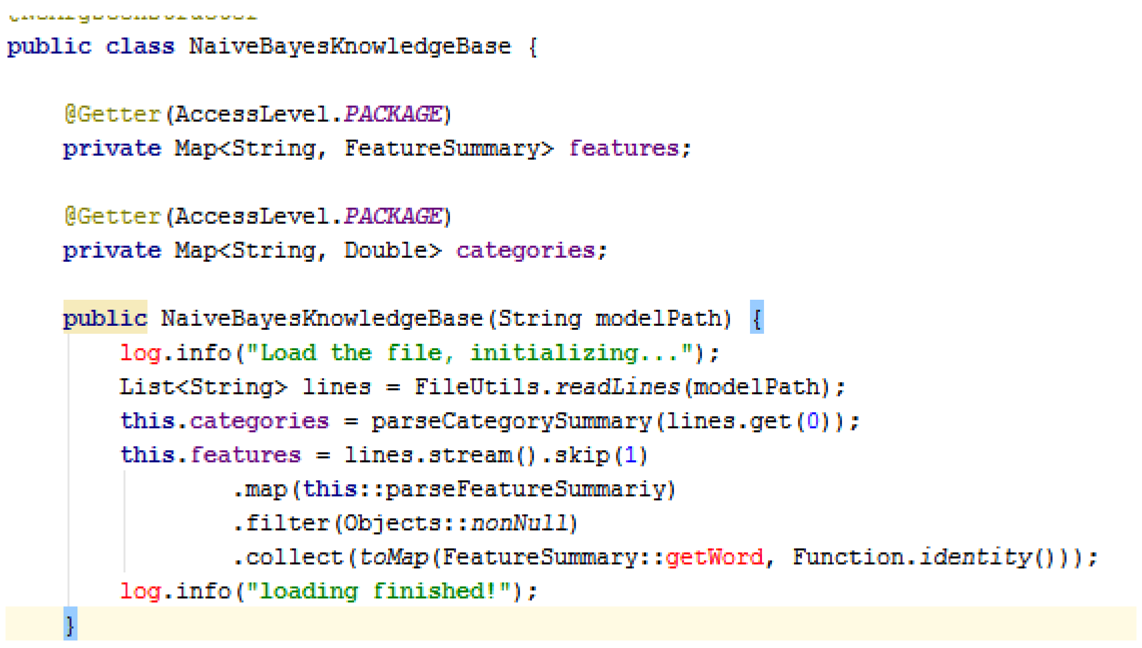
Figure 15: Naive Bayes Feature selection

Create Feature Map



```
private Collection<Feature> createFeatureSpace(Stream<Doc> docs) {

    @AllArgsConstructor
    class FeatureCounter {
        private final Map<Term, Feature> featureMap;

        private FeatureCounter accumulate(Doc doc) {
            Map<Term, Feature> temp = doc.getTerms().parallelStream()
                    .map(t -> new Feature(t, doc.getCategory()))
                    .collect(toMap(Feature::getTerm, Function.identity()));

            if (!featureMap.isEmpty())
                featureMap.values().forEach(f -> temp.merge(f.getTerm(), f, Feature::merge));
            return new FeatureCounter(temp);
        }

        private FeatureCounter combine(FeatureCounter featureCounter) {
            Map<Term, Feature> temp = Maps.newHashMap(featureMap);
            featureCounter.featureMap.values().forEach(f -> temp.merge(f.getTerm(), f, Feature::merge));
            return new FeatureCounter(temp);
        }
    }

    FeatureCounter counter = docs.parallel()
            .reduce(new FeatureCounter(Maps.newHashMap()),
                    FeatureCounter::accumulate,
                    FeatureCounter::combine);

    return counter.featureMap.values();
}
```

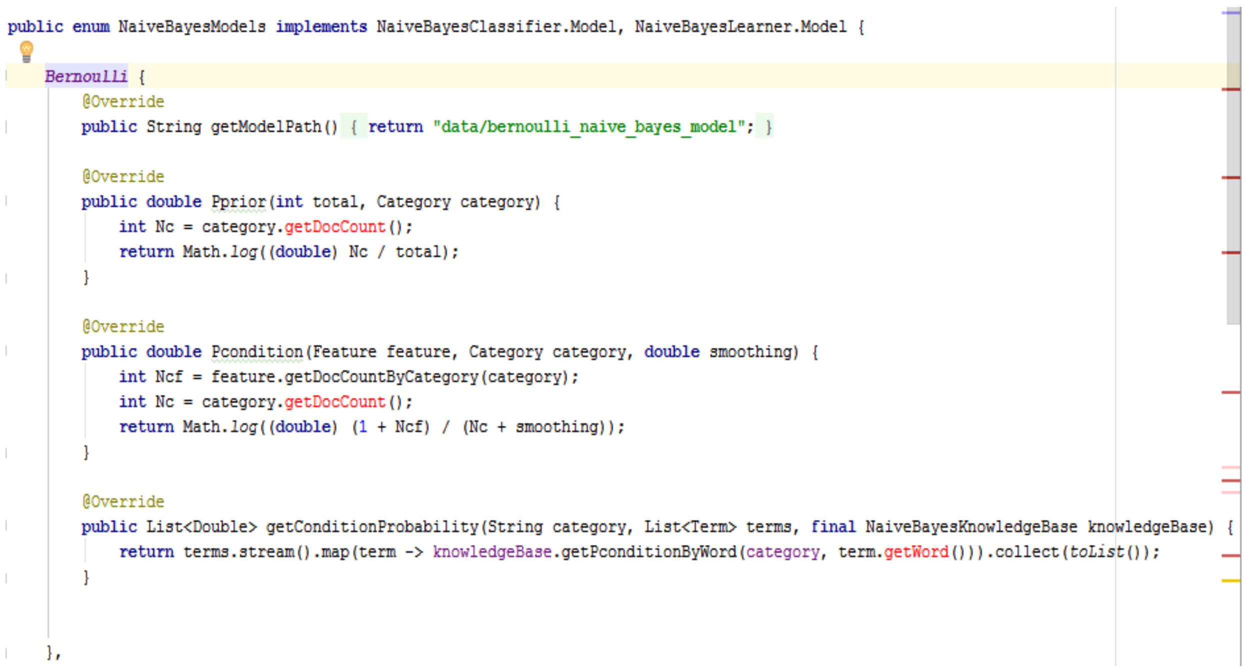
Figure 16: Feature Map for Naive Bayes classifier



```
[illegible]
public class NaiveBayesKnowledgeBase {

    @Getter(AccessLevel.PACKAGE)
    private Map<String, FeatureSummary> features;

    @Getter(AccessLevel.PACKAGE)
    private Map<String, Double> categories;

    public NaiveBayesKnowledgeBase(String modelPath) {
        log.info("Load the file, initializing...");
        List<String> lines = FileUtils.readLines(modelPath);
        this.categories = parseCategorySummary(lines.get(0));
        this.features = lines.stream().skip(1)
                .map(this::parseFeatureSummariy)
                .filter(Objects::nonNull)
                .collect(toMap(FeatureSummary::getWord, Function.identity()));
        log.info("loading finished!");
    }
```

Figure 17: Naive Bayes Knowledge Base - Store Feature and categories

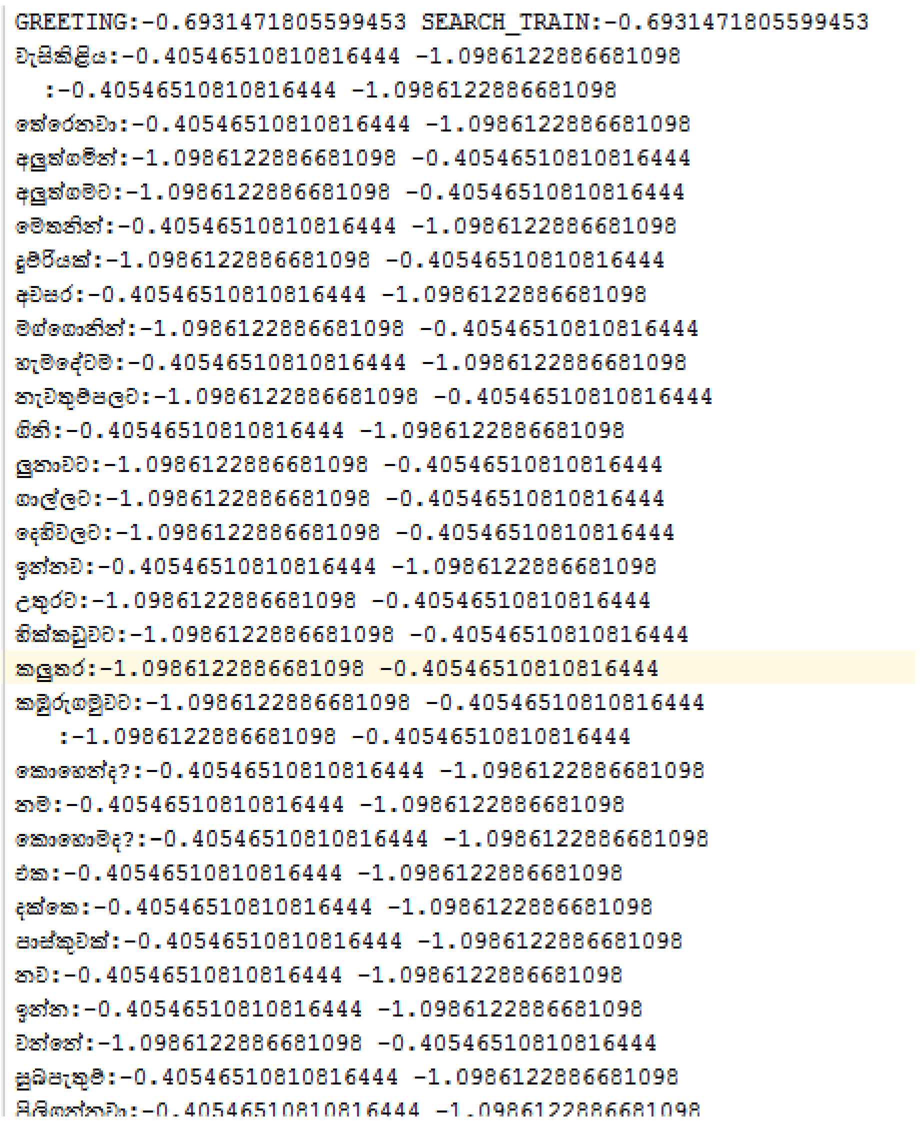
Naïve Bayes Bernoulli model



```
public enum NaiveBayesModels implements NaiveBayesClassifier.Model, NaiveBayesLearner.Model {

    Bernoulli {
        @Override
        public String getModelPath() { return "data/bernoulli_naive_bayes_model"; }

        @Override
        public double Pprior(int total, Category category) {
            int Nc = category.getDocCount();
            return Math.log((double) Nc / total);
        }

        @Override
        public double Pcondition(Feature feature, Category category, double smoothing) {
            int Ncf = feature.getDocCountByCategory(category);
            int Nc = category.getDocCount();
            return Math.log((double) (1 + Ncf) / (Nc + smoothing));
        }

        @Override
        public List<Double> getConditionProbability(String category, List<Term> terms, final NaiveBayesKnowledgeBase knowledgeBase) {
            return terms.stream().map(term -> knowledgeBase.getPconditionByWord(category, term.getWord())).collect(toList());
        }


    },
```

Figure 18: Naive Bayes Bernoulli model

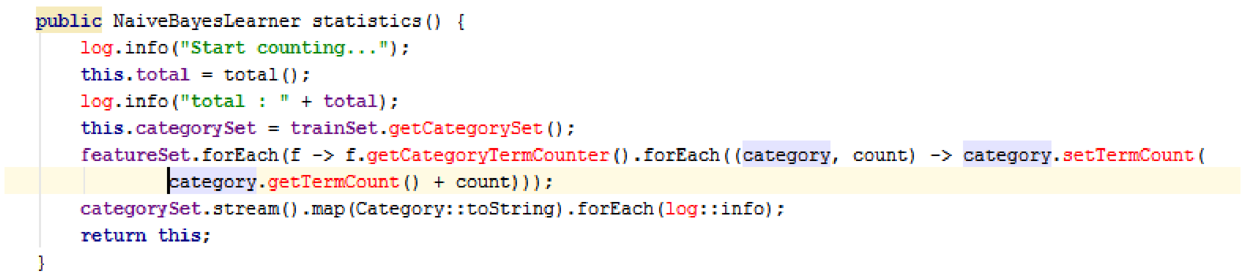
Generated Naïve Bayes model for two intents



```
GREETING:-0.6931471805599453 SEARCH_TRAIN:-0.6931471805599453
වැසිකිළිය:-0.40546510810816444 -1.0986122886681098
  :-0.40546510810816444 -1.0986122886681098
තේරෙනවා:-0.40546510810816444 -1.0986122886681098
අලුත්ගමින්:-1.0986122886681098 -0.40546510810816444
අලුත්ගමට:-1.0986122886681098 -0.40546510810816444
මෙතනින්:-0.40546510810816444 -1.0986122886681098
දුම්රියක්:-1.0986122886681098 -0.40546510810816444
අවසර:-0.40546510810816444 -1.0986122886681098
මග්ගොනින්:-1.0986122886681098 -0.40546510810816444
හැමදේටම:-0.40546510810816444 -1.0986122886681098
නැවතුම්පලට:-1.0986122886681098 -0.40546510810816444
ගිනි:-0.40546510810816444 -1.0986122886681098
ලුනාවට:-1.0986122886681098 -0.40546510810816444
ගාල්ලට:-1.0986122886681098 -0.40546510810816444
දෙහිවලට:-1.0986122886681098 -0.40546510810816444
ඉන්නව:-0.40546510810816444 -1.0986122886681098
උතුරට:-1.0986122886681098 -0.40546510810816444
හික්කඩුවට:-1.0986122886681098 -0.40546510810816444
කලුතර:-1.0986122886681098 -0.40546510810816444
කඹුරුගමුවට:-1.0986122886681098 -0.40546510810816444
   :-1.0986122886681098 -0.40546510810816444
කොහෙන්ද?:-0.40546510810816444 -1.0986122886681098
නම:-0.40546510810816444 -1.0986122886681098
කොහොමද?:-0.40546510810816444 -1.0986122886681098
එක:-0.40546510810816444 -1.0986122886681098
දක්කෙ:-0.40546510810816444 -1.0986122886681098
පාස්කුවක්:-0.40546510810816444 -1.0986122886681098
නව:-0.40546510810816444 -1.0986122886681098
ඉන්න:-0.40546510810816444 -1.0986122886681098
වන්නේ:-1.0986122886681098 -0.40546510810816444
සුබපැතුම්:-0.40546510810816444 -1.0986122886681098
සිහිගන්නවා:-0.40546510810816444 -1.0986122886681098
```

Figure 19: Naive Bayes training result model

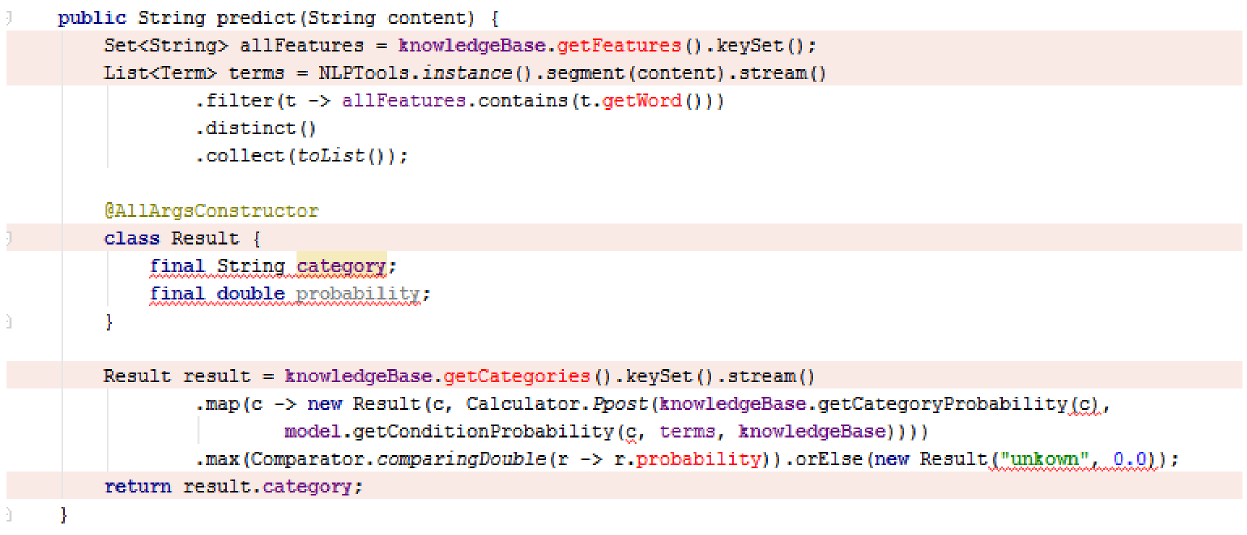
Calculate Statistics by Naive Bayes Learner



```
public NaiveBayesLearner statistics() {
    log.info("Start counting...");
    this.total = total();
    log.info("total : " + total);
    this.categorySet = trainSet.getCategorySet();
    featureSet.forEach(f -> f.getCategoryTermCounter().forEach((category, count) -> category.setTermCount(
            category.getTermCount() + count)));
    categorySet.stream().map(Category::toString).forEach(log::info);
    return this;
}
```

Figure 20: Calculate probabilities

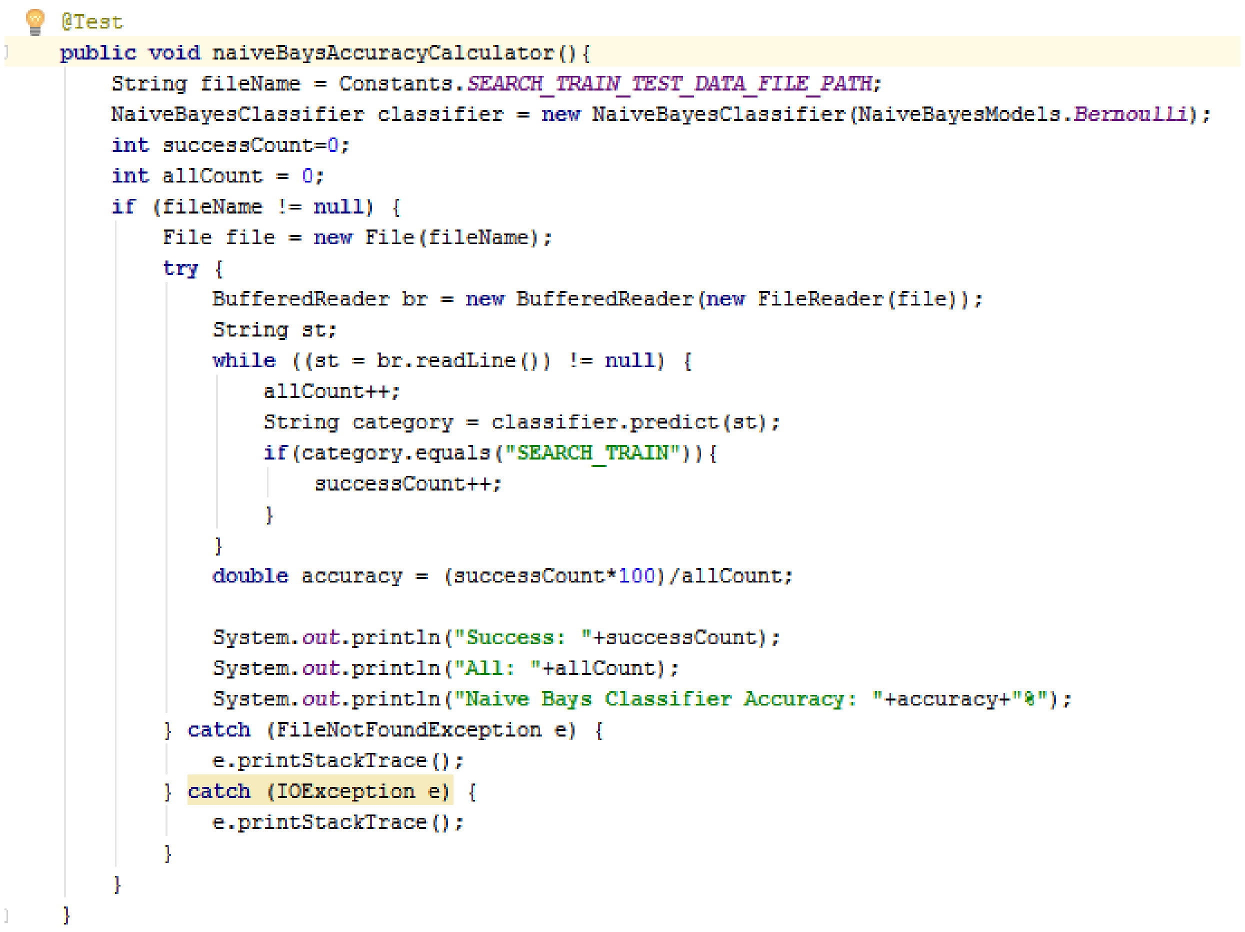
Intent Prediction of Intent classifier



```
public String predict(String content) {
    Set<String> allFeatures = knowledgeBase.getFeatures().keySet();
    List<Term> terms = NLPTools.instance().segment(content).stream()
            .filter(t -> allFeatures.contains(t.getWord()))
            .distinct()
            .collect(toList());

    @AllArgsConstructor
    class Result {
        final String category;
        final double probability;
    }

    Result result = knowledgeBase.getCategories().keySet().stream()
            .map(c -> new Result(c, Calculator.Ppost(knowledgeBase.getCategoryProbability(c),
                    model.getConditionProbability(c, terms, knowledgeBase))))
            .max(Comparator.comparingDouble(r -> r.probability)).orElse(new Result("unkown", 0.0));
    return result.category;
}
```

Figure 21: Intent classification

Evaluation of Intent classifier

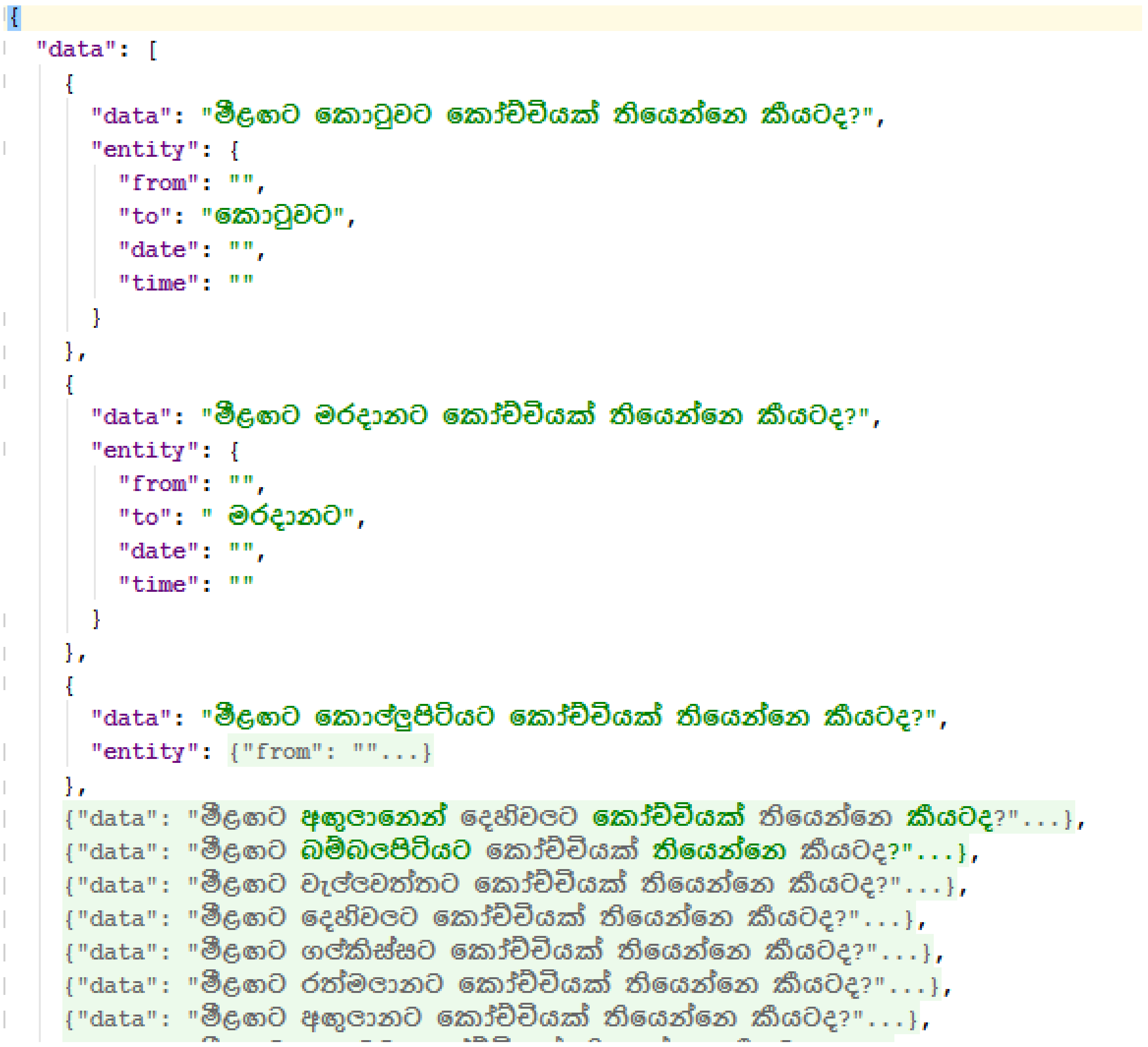


```java
@Test
public void naiveBaysAccuracyCalculator(){
    String fileName = Constants.SEARCH_TRAIN_TEST_DATA_FILE_PATH;
    NaiveBayesClassifier classifier = new NaiveBayesClassifier(NaiveBayesModels.Bernoulli);
    int successCount=0;
    int allCount = 0;
    if (fileName != null) {
        File file = new File(fileName);
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String st;
            while ((st = br.readLine()) != null) {
                allCount++;
                String category = classifier.predict(st);
                if(category.equals("SEARCH_TRAIN")){
                    successCount++;
                }
            }
            double accuracy = (successCount*100)/allCount;

            System.out.println("Success: "+successCount);
            System.out.println("All: "+allCount);
            System.out.println("Naive Bays Classifier Accuracy: "+accuracy+"%");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 22: Accuracy calculation of Intent classifier

Entity Identifier

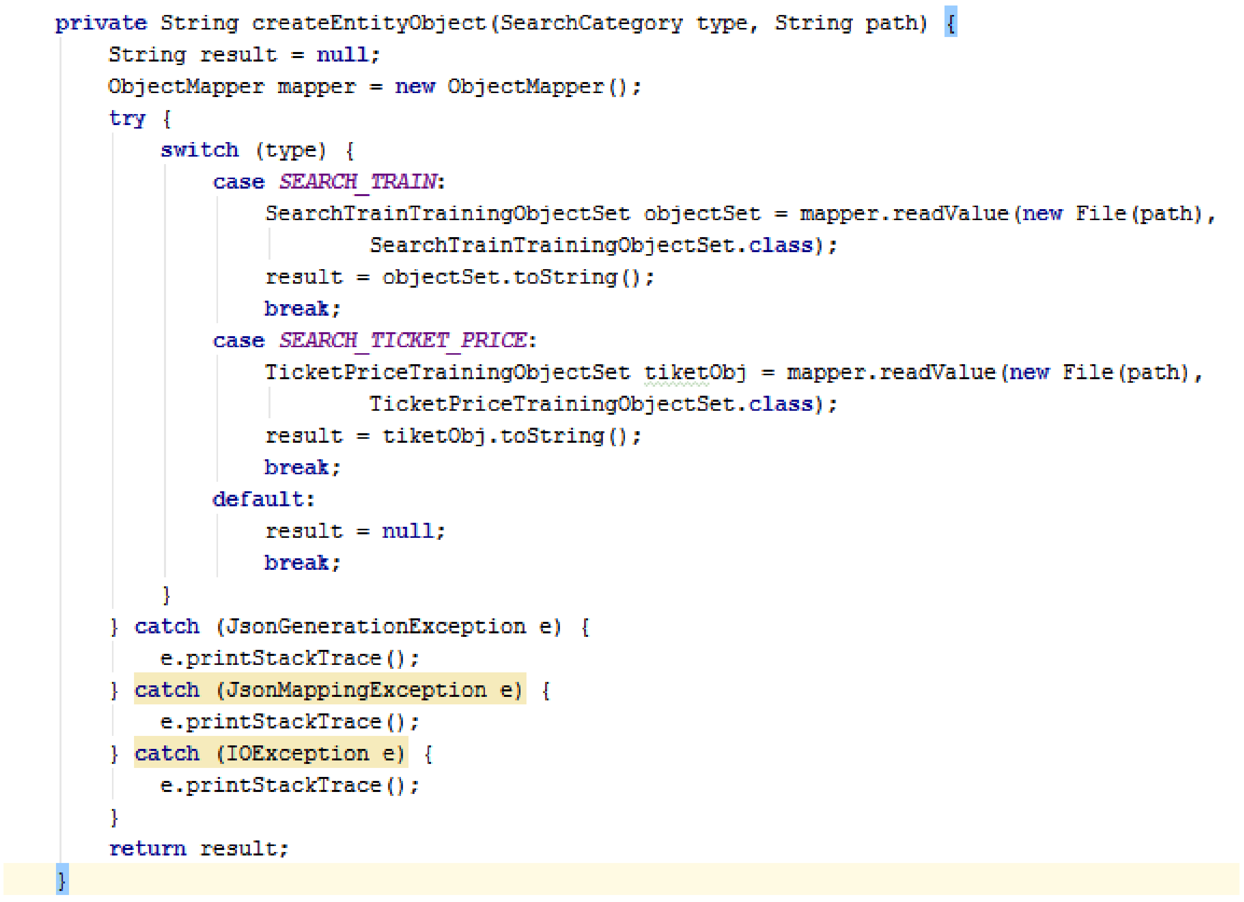
Training Data set for Entity Identifier



```
{
  "data": [
    {
      "data": "මීළඟට කොටුවට කෝච්චියක් තියෙන්නෙ කීයටද?",
      "entity": {
        "from": "",
        "to": "කොටුවට",
        "date": "",
        "time": ""
      }
    },
    {
      "data": "මීළඟට මරදානට කෝච්චියක් තියෙන්නෙ කීයටද?",
      "entity": {
        "from": "",
        "to": " මරදානට",
        "date": "",
        "time": ""
      }
    },
    {
      "data": "මීළඟට කොල්ලුපිටියට කෝච්චියක් තියෙන්නෙ කීයටද?",
      "entity": {"from": ""...}
    },
    {"data": "මීළඟට අඟුලානෙන් දෙහිවලට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට බම්බලපිටියට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට වැල්ලවත්තට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට දෙහිවලට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට ගල්කිස්සට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට රත්මලානට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
    {"data": "මීළඟට අඟුලානට කෝච්චියක් තියෙන්නෙ කීයටද?"...},
```

Figure 23: Training Data file

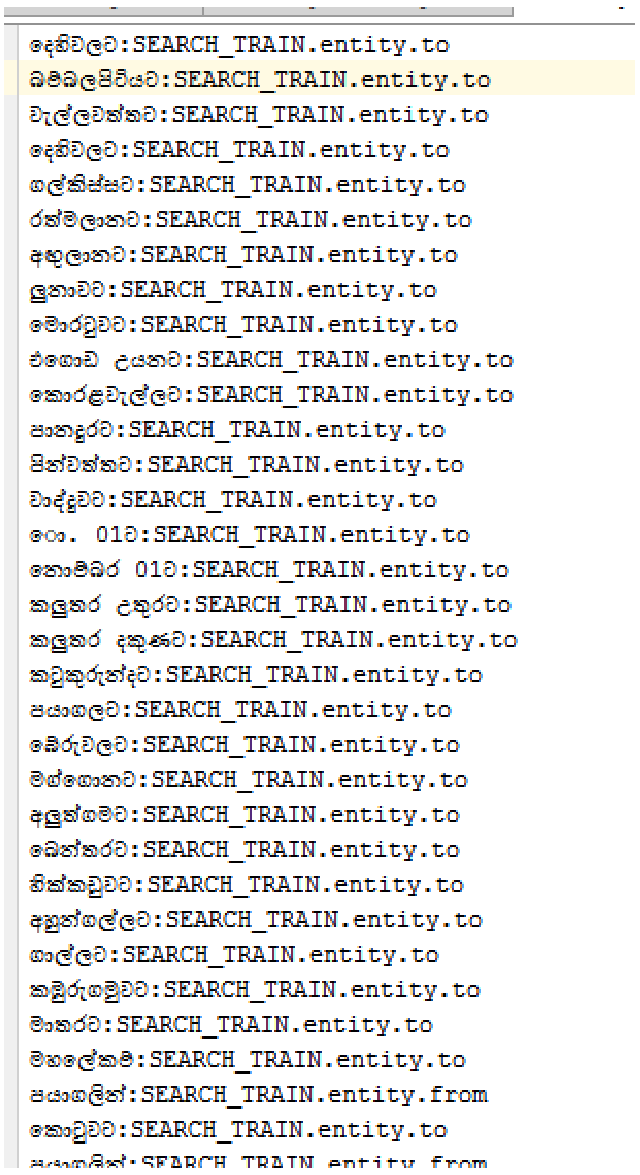
Entity Identifier Training



```java
private String createEntityObject(SearchCategory type, String path) {
    String result = null;
    ObjectMapper mapper = new ObjectMapper();
    try {
        switch (type) {
            case SEARCH_TRAIN:
                SearchTrainTrainingObjectSet objectSet = mapper.readValue(new File(path),
                        SearchTrainTrainingObjectSet.class);
                result = objectSet.toString();
                break;
            case SEARCH_TICKET_PRICE:
                TicketPriceTrainingObjectSet tiketObj = mapper.readValue(new File(path),
                        TicketPriceTrainingObjectSet.class);
                result = tiketObj.toString();
                break;
            default:
                result = null;
                break;
        }
    } catch (JsonGenerationException e) {
        e.printStackTrace();
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return result;
}
```

Figure 24: Train the Entity Identifier and Create data for the Entity model

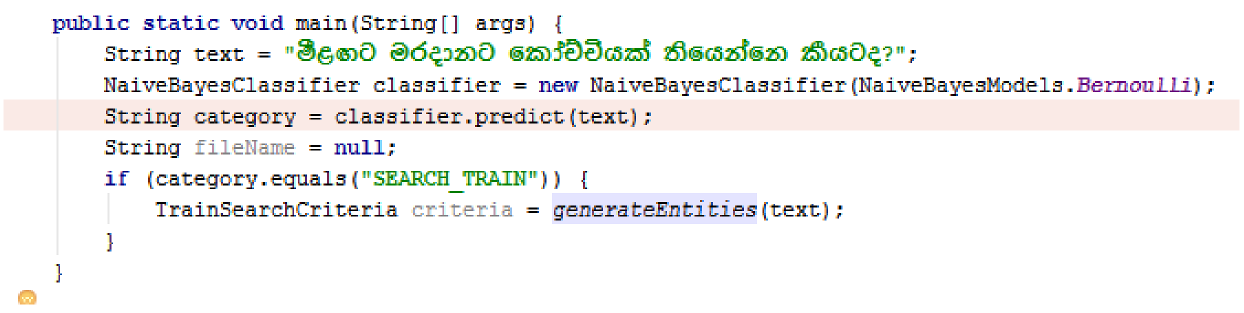
Result model of Intent identifier Training



```
දෙහිවලට:SEARCH_TRAIN.entity.to
බම්බලපිටියට:SEARCH_TRAIN.entity.to
වැල්ලවත්තට:SEARCH_TRAIN.entity.to
දෙහිවලට:SEARCH_TRAIN.entity.to
ගල්කිස්සට:SEARCH_TRAIN.entity.to
රත්මලානට:SEARCH_TRAIN.entity.to
අඟුලානට:SEARCH_TRAIN.entity.to
ලුනාවට:SEARCH_TRAIN.entity.to
මොරටුවට:SEARCH_TRAIN.entity.to
එගොඩ උයනට:SEARCH_TRAIN.entity.to
කොරළවැල්ලට:SEARCH_TRAIN.entity.to
පානදුරට:SEARCH_TRAIN.entity.to
පින්වත්තට:SEARCH_TRAIN.entity.to
වාද්දුවට:SEARCH_TRAIN.entity.to
කො. 01ට:SEARCH_TRAIN.entity.to
නොම්බර 01ට:SEARCH_TRAIN.entity.to
කලුතර උතුරට:SEARCH_TRAIN.entity.to
කලුතර දකුණට:SEARCH_TRAIN.entity.to
කටුකුරුන්දට:SEARCH_TRAIN.entity.to
පයාගලට:SEARCH_TRAIN.entity.to
බේරුවලට:SEARCH_TRAIN.entity.to
මග්ගොනට:SEARCH_TRAIN.entity.to
අලුත්ගමට:SEARCH_TRAIN.entity.to
බෙන්තරට:SEARCH_TRAIN.entity.to
හික්කඩුවට:SEARCH_TRAIN.entity.to
අහුන්ගල්ලට:SEARCH_TRAIN.entity.to
ගාල්ලට:SEARCH_TRAIN.entity.to
කඹුරුගමුවට:SEARCH_TRAIN.entity.to
මාතරට:SEARCH_TRAIN.entity.to
මහලේකම්:SEARCH_TRAIN.entity.to
පයාගලින්:SEARCH_TRAIN.entity.from
කොටුවට:SEARCH_TRAIN.entity.to
පයාගලින්:SEARCH_TRAIN.entity.from
```

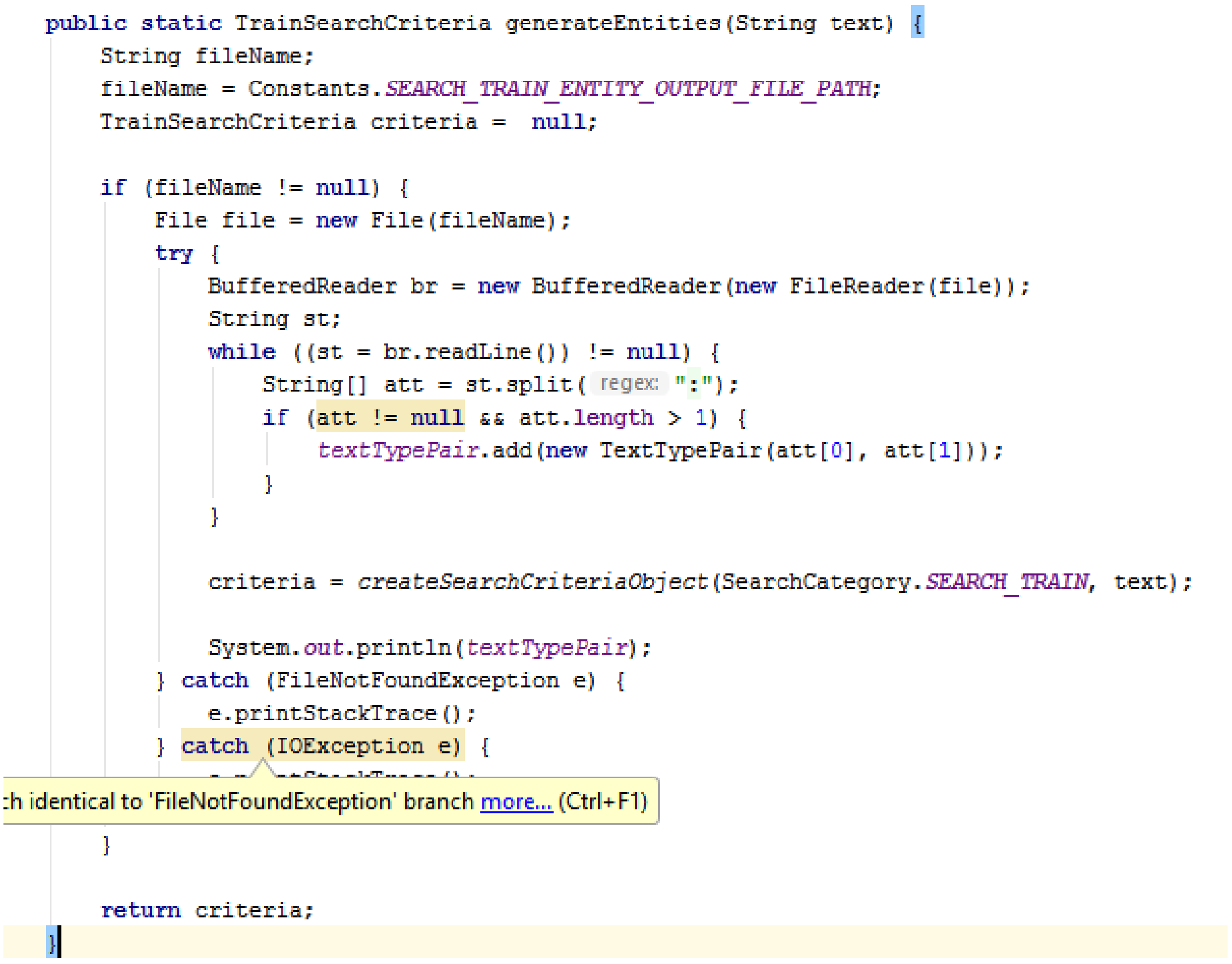
Figure 25: Sample data of Entity model

Entity Identification



```
public static void main(String[] args) {
    String text = "ඊළඟට මරදානට කෝච්චියක් තියෙන්නෙ කීයටද?";
    NaiveBayesClassifier classifier = new NaiveBayesClassifier(NaiveBayesModels.Bernoulli);
    String category = classifier.predict(text);
    String fileName = null;
    if (category.equals("SEARCH_TRAIN")) {
        TrainSearchCriteria criteria = generateEntities(text);
    }
}
```

Figure 26: Identify intent and send values to generate Entities

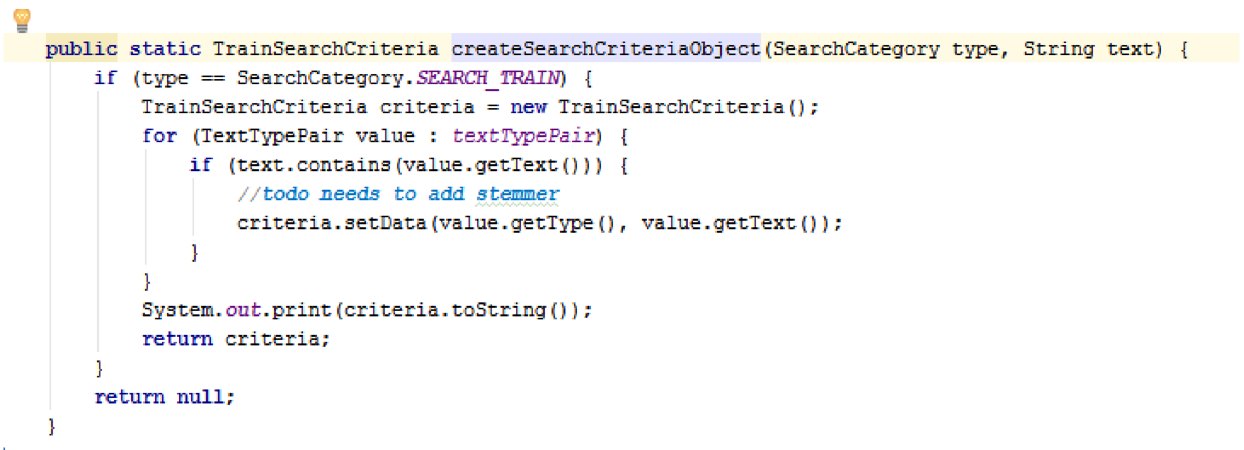


```
public static TrainSearchCriteria generateEntities(String text) {
    String fileName;
    fileName = Constants.SEARCH_TRAIN_ENTITY_OUTPUT_FILE_PATH;
    TrainSearchCriteria criteria =  null;

    if (fileName != null) {
        File file = new File(fileName);
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String st;
            while ((st = br.readLine()) != null) {
                String[] att = st.split( regex: ":");
                if (att != null && att.length > 1) {
                    textTypePair.add(new TextTypePair(att[0], att[1]));
                }
            }

            criteria = createSearchCriteriaObject(SearchCategory.SEARCH_TRAIN, text);

            System.out.println(textTypePair);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

ch identical to 'FileNotFoundException' branch more... (Ctrl+F1)

```
    }

    return criteria;
}
```

Figure 27: Generate Entities for Search Train Intent

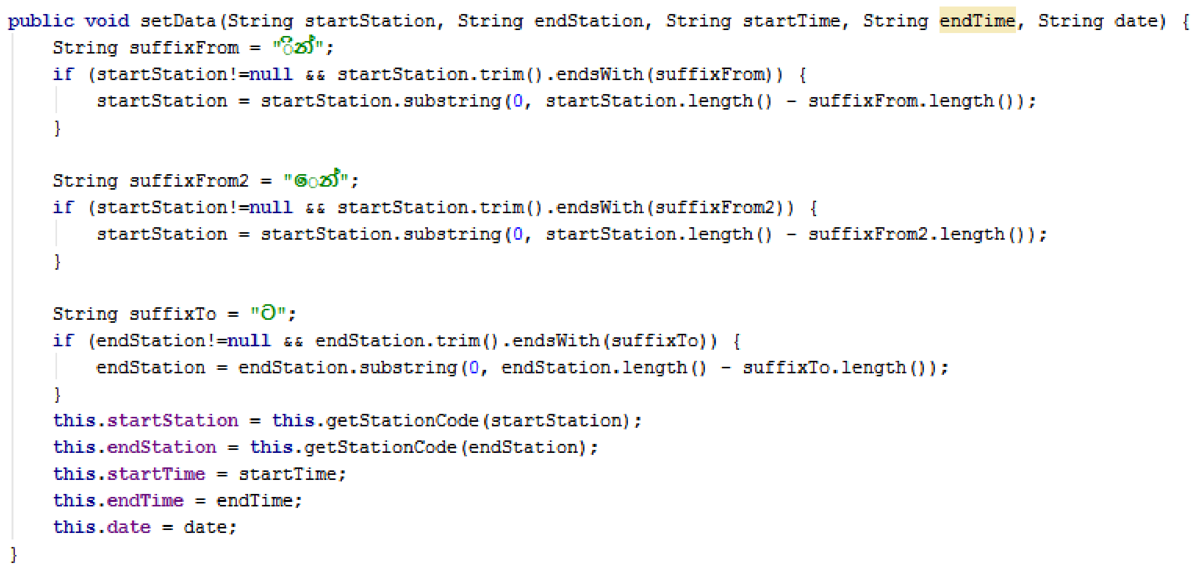


```java
public static TrainSearchCriteria createSearchCriteriaObject(SearchCategory type, String text) {
    if (type == SearchCategory.SEARCH_TRAIN) {
        TrainSearchCriteria criteria = new TrainSearchCriteria();
        for (TextTypePair value : textTypePair) {
            if (text.contains(value.getText())) {
                //todo needs to add stemmer
                criteria.setData(value.getType(), value.getText());
            }
        }
        System.out.print(criteria.toString());
        return criteria;
    }
    return null;
}
```

Figure 28: Create criteria object with identified knowledge

Response Generator

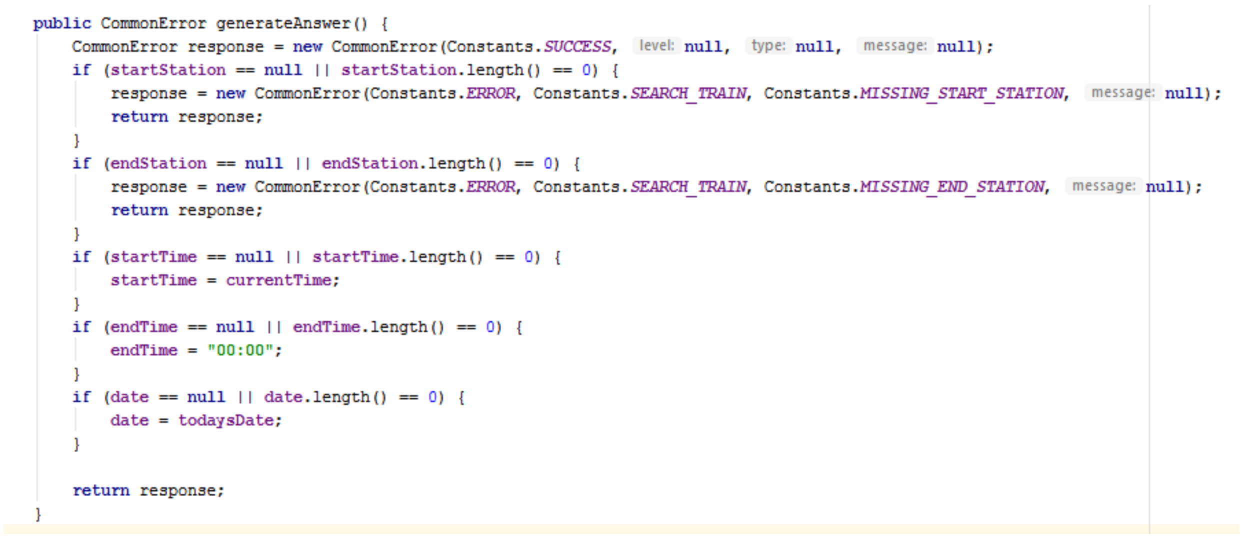
Format Criteria data



```java
public void setData(String startStation, String endStation, String startTime, String endTime, String date) {
    String suffixFrom = "ින්";
    if (startStation!=null && startStation.trim().endsWith(suffixFrom)) {
        startStation = startStation.substring(0, startStation.length() - suffixFrom.length());
    }

    String suffixFrom2 = "ෙන්";
    if (startStation!=null && startStation.trim().endsWith(suffixFrom2)) {
        startStation = startStation.substring(0, startStation.length() - suffixFrom2.length());
    }

    String suffixTo = "ට";
    if (endStation!=null && endStation.trim().endsWith(suffixTo)) {
        endStation = endStation.substring(0, endStation.length() - suffixTo.length());
    }
    this.startStation = this.getStationCode(startStation);
    this.endStation = this.getStationCode(endStation);
    this.startTime = startTime;
    this.endTime = endTime;
    this.date = date;
}
```

Figure 29: Format Entity parameters for response generation

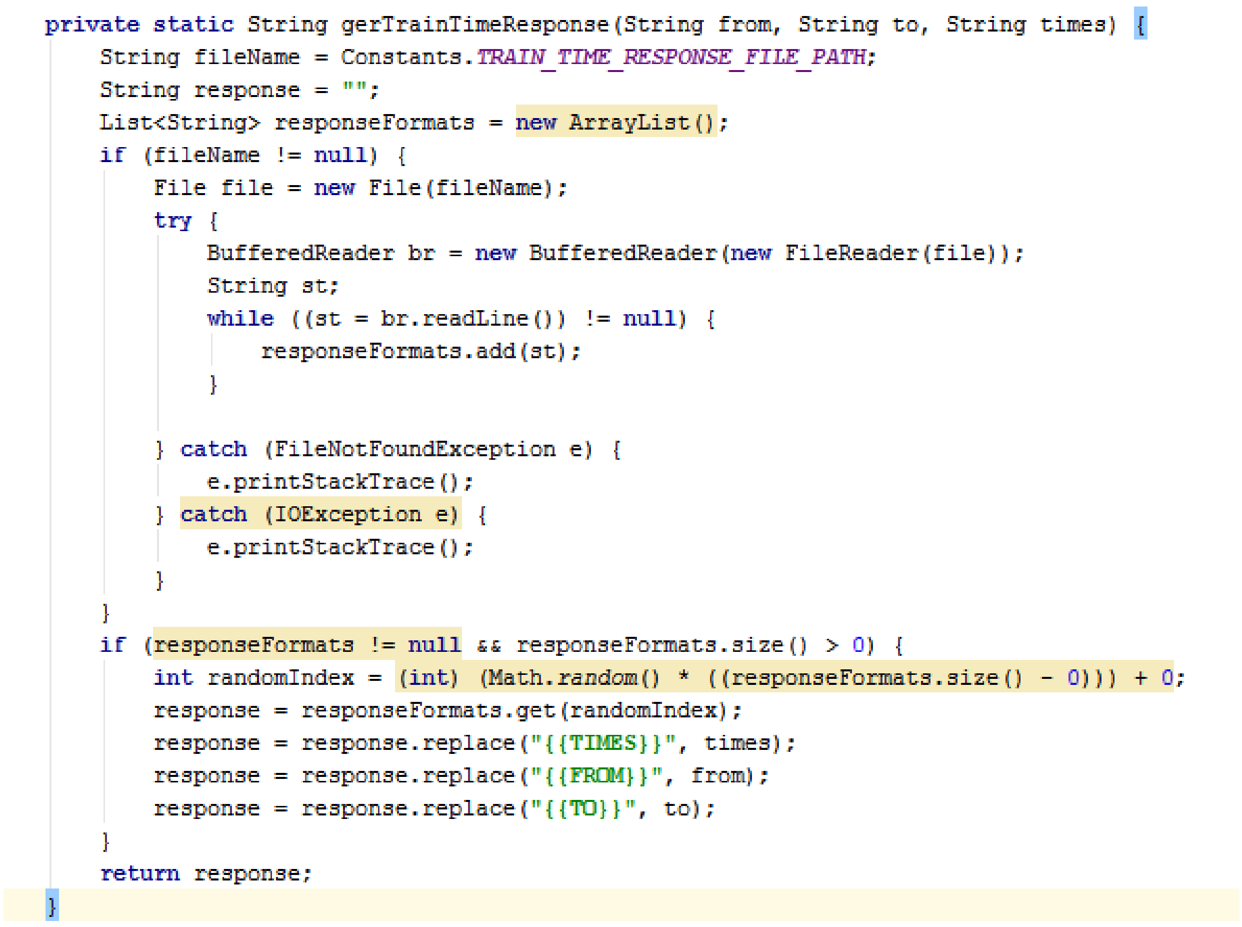
Generate Answer in Rule based approach



```
public CommonError generateAnswer() {
    CommonError response = new CommonError(Constants.SUCCESS,  level: null,  type: null,  message: null);
    if (startStation == null || startStation.length() == 0) {
        response = new CommonError(Constants.ERROR, Constants.SEARCH_TRAIN, Constants.MISSING_START_STATION,  message: null);
        return response;
    }
    if (endStation == null || endStation.length() == 0) {
        response = new CommonError(Constants.ERROR, Constants.SEARCH_TRAIN, Constants.MISSING_END_STATION,  message: null);
        return response;
    }
    if (startTime == null || startTime.length() == 0) {
        startTime = currentTime;
    }
    if (endTime == null || endTime.length() == 0) {
        endTime = "00:00";
    }
    if (date == null || date.length() == 0) {
        date = todaysDate;
    }

    return response;
}
```

Figure 30: Generate answer

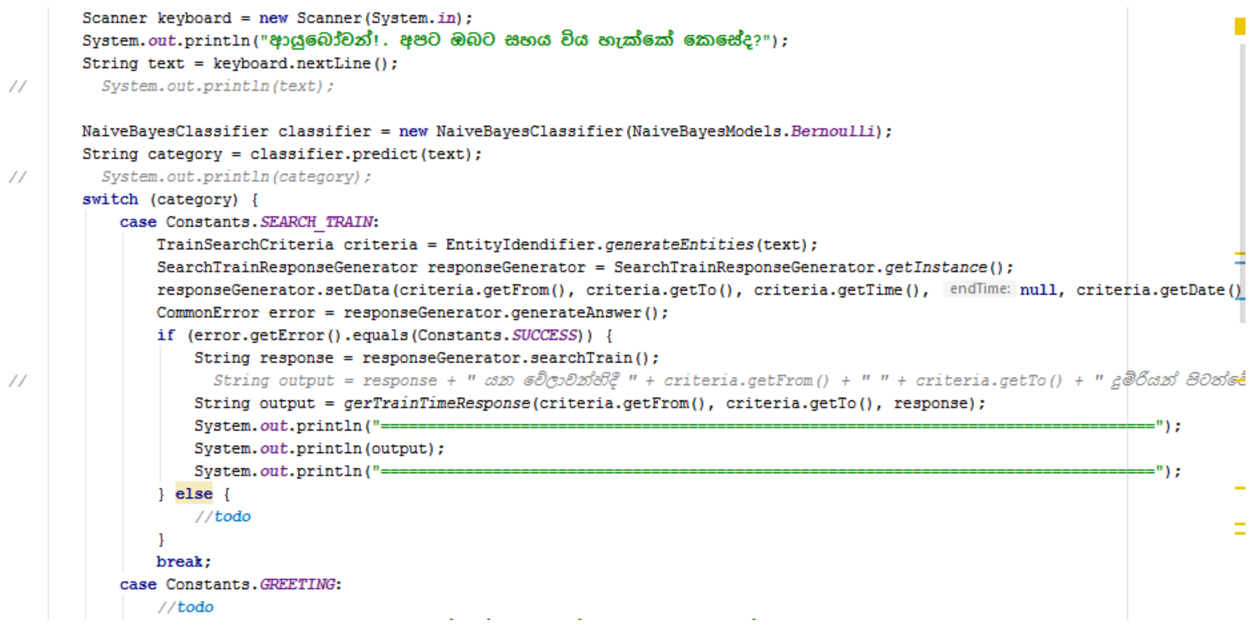
Random Response selector



```java
private static String gerTrainTimeResponse(String from, String to, String times) {
    String fileName = Constants.TRAIN_TIME_RESPONSE_FILE_PATH;
    String response = "";
    List<String> responseFormats = new ArrayList();
    if (fileName != null) {
        File file = new File(fileName);
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String st;
            while ((st = br.readLine()) != null) {
                responseFormats.add(st);
            }

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (responseFormats != null && responseFormats.size() > 0) {
        int randomIndex = (int) (Math.random() * ((responseFormats.size() - 0))) + 0;
        response = responseFormats.get(randomIndex);
        response = response.replace("{{TIMES}}", times);
        response = response.replace("{{FROM}}", from);
        response = response.replace("{{TO}}", to);
    }
    return response;
}
```

Figure 31: Select Random response template and assign values to the template

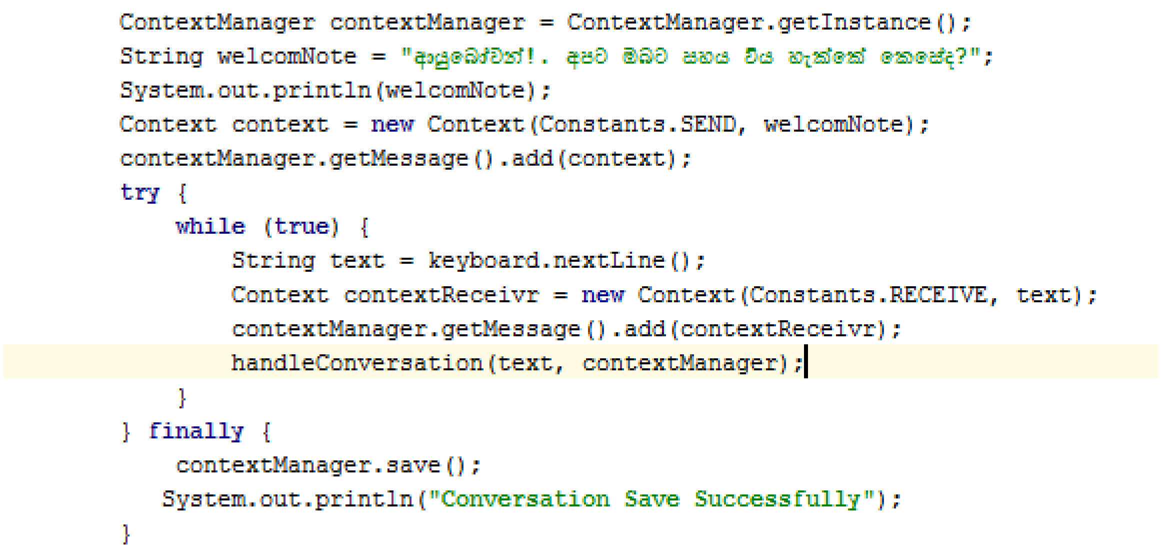
Conversation Manager



```
    Scanner keyboard = new Scanner(System.in);
    System.out.println("ආයුබෝවන්!. අපට ඔබට සහය විය හැක්කේ කෙසේද?");
    String text = keyboard.nextLine();
//    System.out.println(text);

    NaiveBayesClassifier classifier = new NaiveBayesClassifier(NaiveBayesModels.Bernoulli);
    String category = classifier.predict(text);
//    System.out.println(category);
    switch (category) {
        case Constants.SEARCH_TRAIN:
            TrainSearchCriteria criteria = EntityIdendifier.generateEntities(text);
            SearchTrainResponseGenerator responseGenerator = SearchTrainResponseGenerator.getInstance();
            responseGenerator.setData(criteria.getFrom(), criteria.getTo(), criteria.getTime(), endTime: null, criteria.getDate()
            CommonError error = responseGenerator.generateAnswer();
            if (error.getError().equals(Constants.SUCCESS)) {
                String response = responseGenerator.searchTrain();
//                String output = response + " යන වේලාවන්හිදී " + criteria.getFrom() + " " + criteria.getTo() + " දුම්රියන් පිටත්
                String output = gerTrainTimeResponse(criteria.getFrom(), criteria.getTo(), response);
                System.out.println("=================================================================================");
                System.out.println(output);
                System.out.println("=================================================================================");
            } else {
                //todo
            }
            break;
        case Constants.GREETING:
            //todo
```

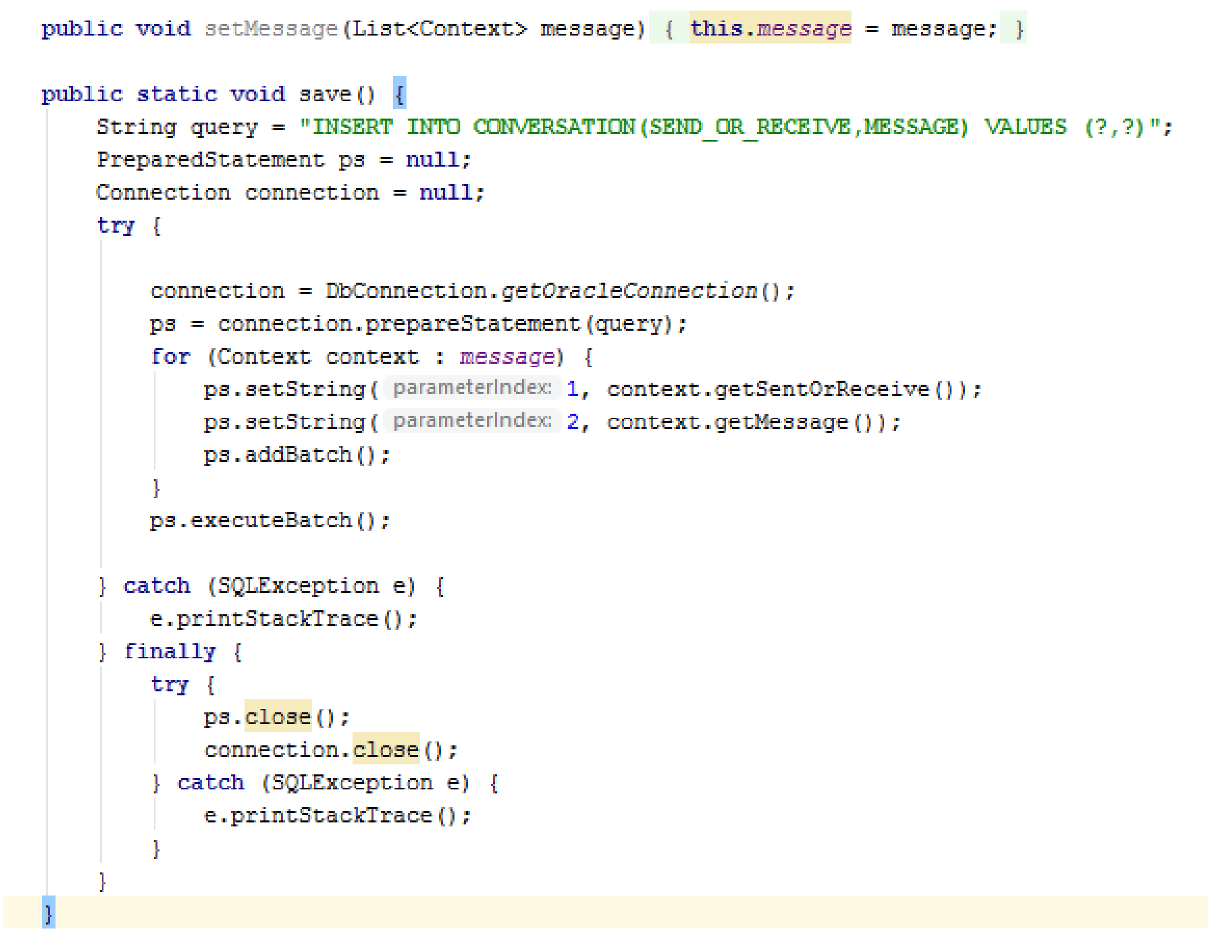
Figure 32: Flow of a single conversation

Context Management



```
ContextManager contextManager = ContextManager.getInstance();
String welcomNote = "ආයුබෝවන්!. අපට ඔබට සහය විය හැක්කේ කෙසේද?";
System.out.println(welcomNote);
Context context = new Context(Constants.SEND, welcomNote);
contextManager.getMessage().add(context);
try {
    while (true) {
        String text = keyboard.nextLine();
        Context contextReceivr = new Context(Constants.RECEIVE, text);
        contextManager.getMessage().add(contextReceivr);
        handleConversation(text, contextManager);
    }
} finally {
    contextManager.save();
   System.out.println("Conversation Save Successfully");
}
```

Figure 33: Context Management



```java
public void setMessage(List<Context> message) { this.message = message; }

public static void save() {
    String query = "INSERT INTO CONVERSATION(SEND_OR_RECEIVE,MESSAGE) VALUES (?,?)";
    PreparedStatement ps = null;
    Connection connection = null;
    try {

        connection = DbConnection.getOracleConnection();
        ps = connection.prepareStatement(query);
        for (Context context : message) {
            ps.setString( parameterIndex: 1, context.getSentOrReceive());
            ps.setString( parameterIndex: 2, context.getMessage());
            ps.addBatch();
        }
        ps.executeBatch();

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            ps.close();
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 34: Save data to Database