DEVELOPING A RETRIEVAL-BASED TAMIL LANGUAGE CHATBOT FOR CLOSED DOMAIN

Kumaran Kugathasan

198097X

Thesis/Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Computer Science and Engineering

Department of Computer Science & Engineering

University of Moratuwa

Sri Lanka

August 2023

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 14/08/2023

The above candidate has carried out research for the Masters thesis/Dissertation

under my supervision.

Signature of the Supervisor: Date: 15/08/2023

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Dr. Uthayasanker Thayasivam, my supervisor, for his invaluable advice and guidance throughout this project. I am grateful for his willingness to make himself available whenever I needed his assistance. Without his support and encouragement, I could not have completed this project successfully.

Additionally, I would like to thank members of my progress review committee Prof. Sanath Jayasena and Dr. Charith Chitraranjan, for their insightful feedback and guidance, which were extremely helpful to me.

I am also grateful to the entire lecturers and research students at the DataSEARCH Research Centre for their assistance, feedback and the resources they provided to carry out the project.

Lastly, I would like to say thank you to my family and friends for their unfailing support.

ABSTRACT

Chatbots are conversational systems that interact with humans via natural language. Frequently, it is used to respond to user queries and provide them with the information they need. To build a highly functional chatbot, a good corpus and a variety of language-related resources are required. Since Tamil is a low-resource language those resources are not available for Tamil. Additionally, since Tamil is also a morphologically rich language, high inflexion and free word order pose key challenges to Tamil chatbots. Due to all the above reasons, it is evident that developing an effective End-to-End chat system is challenging even for a closed domain.

This study introduces a novel method for building a chatbot in Tamil by leveraging a dataset extracted from Tamil banking website's FAQ sections and extending it to encompass the language's morphological complexity and rich inflectional structure. Intent is assigned to each query, and a multiclass intent classifier is developed to classify user intent. The CNN-based classifier demonstrated the highest performance, achieving an accuracy of 98.72%.

While previous works on short-text classification in Tamil focused only on a few classes and used a very large dataset, our method produced a superior accuracy of over 98% using a small number of per-class examples even when there are 56 classes and additional challenges like class imbalance problem in the data. This shows our approach is better than any other approach for short text classification in Tamil.

The major contribution of this research is the generation of the first-ever chat dataset for Tamil. Our research is the first of its kind in Tamil to show how an efficient context-less chatbot can be built using short text classification. Although this project is done for the Tamil language and for the Banking domain, this approach can be applied to other low-resourced languages and domains as well.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

FAQ	Frequently Asked Questions
NLU	Natural Language Understanding
NLG	Natural Language Generation
AIML	Artificial Intelligence Markup Language
POS	Part-of-speech
NER	Named-Entity Recognizer
BOW	Bag of Words
TF-IDF	Term frequency–inverse document frequency
SMOTE	Synthetic Minority Oversampling Technique

TABLE OF CONTENTS

1. INTRODUCTION

Chatbots are conversational systems that interact with humans via natural language [1]. Frequently, it is used to respond to user queries and provide them with the information they need. Currently, most businesses answer customers' questions about their products and services through their FAQ page or by hiring customer service agents. There are, however, several downsides to both approaches. The FAQ pages are lengthy and non-interactive, hence substantial effort is required to find the information the customer is looking for. As a result, customers have difficulty accessing the information they need within a short period of time.

If you were to use human customer service agents, it would be an expensive option to choose from. In order to provide 24x7 assistance, it is necessary to have these individuals available at all times of the day and night. There are also instances in which there may be a delay in responding if a significant number of people are not hired, as highlighted in Figure 1. It may be difficult for human agents to respond to customer chats within a reasonable amount of time if they are handling multiple chat requests at the same time. There is nothing more frustrating than being put on hold and having to wait for a long time. It is common for customers to switch from one business to another as a result of poor customer service experiences. There is therefore a need to come up with a more effective solution in order to overcome some of these issues.

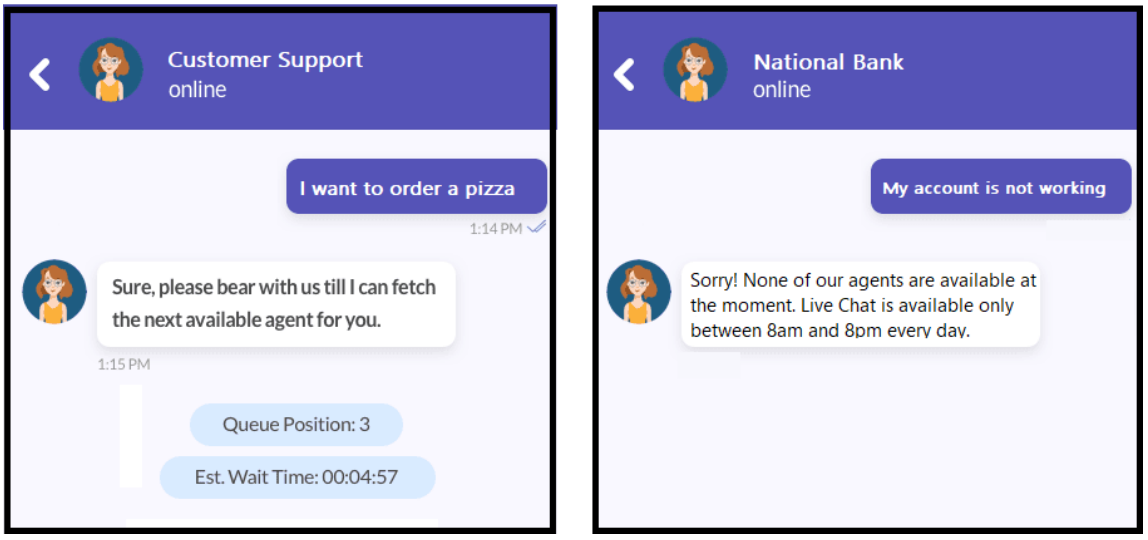

Figure 1: Problems with using human agents.

In order to address the problem described above, it would be most beneficial to use a text-based conversational agent as a solution. Using a chatbot has a number of benefits that can be attributed to it. A few of these features include quick response time, 24x7 availability, and the fact that it costs a fraction of what it would cost to hire a customer service agent. Hence, it's understandable that there has been a surge in global interest in chatbots in recent years, as depicted in Figure 2. The y-axis values in Figure 2 show peak interest levels, not query quantities. A score of 100 represents the peak interest time, with 25 indicating a quarter of that peak interest, and so forth.

Figure 2: Interest in chatbot over time

1.1 Type of chatbot

Depending on the nature of the conversation, it is possible to categorize chat systems into two categories: open-domain chat systems and closed-domain chat systems [1]. An open-domain chat system does not have a particular focus or focus on a particular area, and the topic of the conversation may change as the discussion progresses; a closed-domain chat system, on the other hand, is restricted to a specific topic, industry or sector, and can only serve a specific purpose [2]. For example, in Figure 3, if a user interacts with an open domain chatbot by asking about today's weather and then inquiring about the winners of the 2011 cricket World Cup, the system can address both queries, even though they are unrelated. In contrast, Figure 4 illustrates a closed domain chatbot focused solely on weather. If a user asks about the cricket World Cup winner, this chatbot would likely respond with 'I don't understand,' as its knowledge is confined to the weather domain.

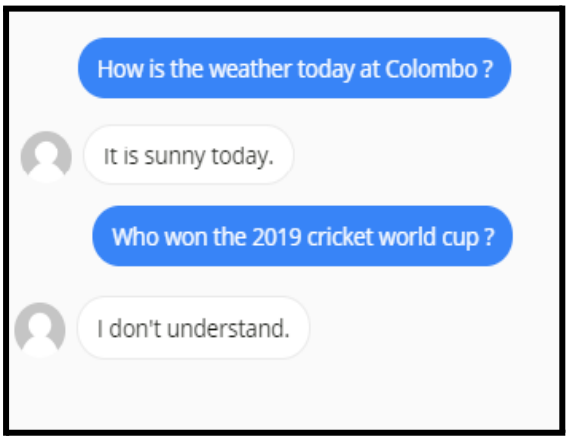

Figure 3: Open Domain Chatbot

Figure 4: Closed Domain Chatbot

Based on the approach used for development, the chatbot system can be categorised as a retrieval system and generative system [2]. The retrieval-based system picks a response from a fixed set of responses based on the input and context using a rule-based approach or by using machine learning classifiers[3]. In generative systems responses are produced word-by-word from scratch[4].

Based on the application of the chatbot, the chatbot system can be categorised as a goal-oriented system and a non-goal-oriented system. The Goal-oriented chatbots are expected to help users to complete the task. These tasks can be anything from restaurant reservations, movie ticket booking, bank transactions, etc.

These kinds of systems engage in multiturn conversation and are expected to understand the user input within the context of the goal and actively engage in conversation to assist the user to complete the task. In the case of non-goal-oriented chatbots, the chatbot basically functions as the question-answering system, when a user asks a question, it will just provide an appropriate answer for the question.

The difference between the two systems becomes very much evident if we refer to Figures 5 and Figure 6. In Figure 5, it could be observed that the chatbot is actively asking questions to guide the user to complete the restaurant reservation. But in Figure 6, the chatbot just provides answers to the question asked without initiating any follow-up questions.

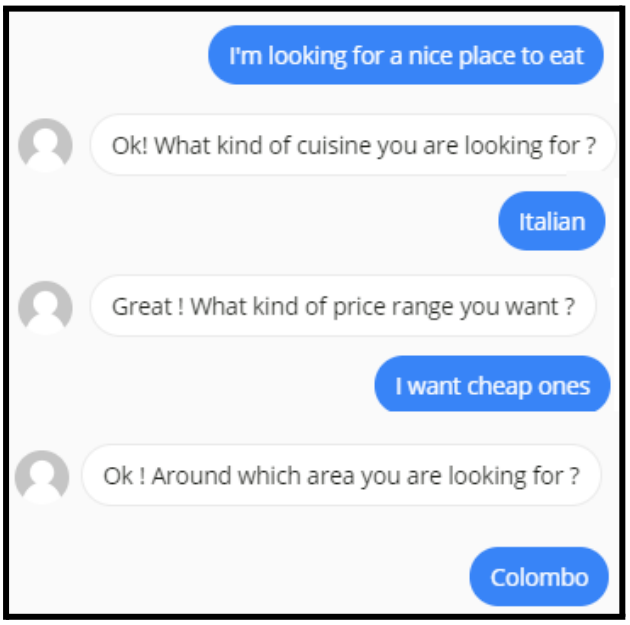

Figure 5: Goal oriented chatbot

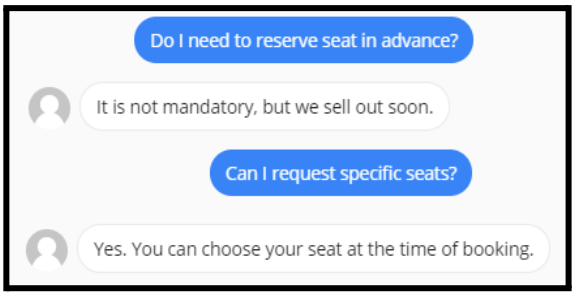

Figure 6: Non-Goal oriented Chatbot

Based on the functionality of the chatbot, the chatbot can be divided into different levels.

Level 1 - Notification Assistants: The notification assistants just serve to deliver notifications to the users, if the user asks any follow-up question about the notification, it will just transfer the chat back to the human agent. Figure 7 illustrates a Level 1 chatbot.

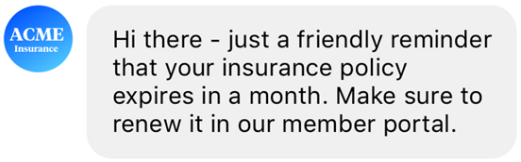

Figure 7 - Level 1 Chatbot. Adapted from [19]

Level 2 - FAQ Assistants: The FAQ assistants just provide answers to the questions asked by the user about the area it has extensive knowledge on. Figure 8 illustrates a Level 2 chatbot.

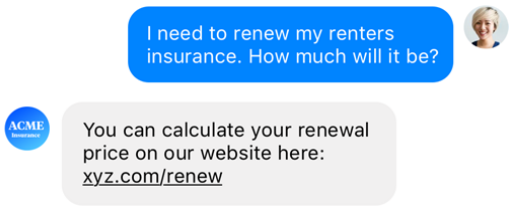

Figure 8: Level 2 Chatbot. Adapted from [19]

Level 3 - Contextual Assistants: The system understands the user query and answers based on the context of the conversation flow. Figure 9 illustrates a Level 3 chatbot.

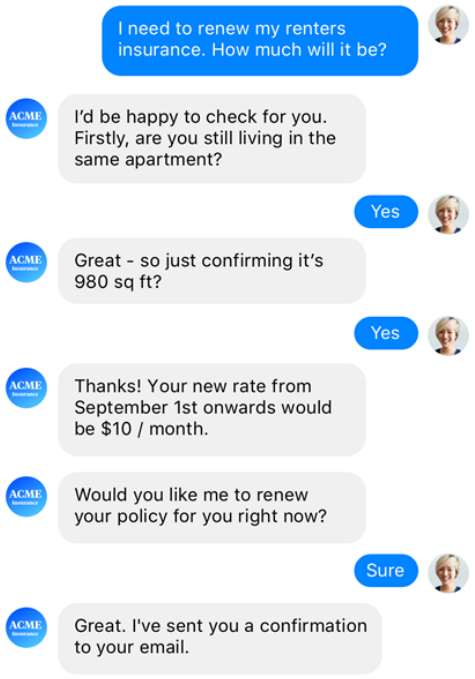

Figure 9 - Level 3 Chatbot. Adapted from [19]

Level 4 - Personalised Assistants: Chat system is personalised for each person and it will remember your preferences and give you the ultimate, personalised interface.

Level 5 - Autonomous Organisation of Assistants: The end-to-end system is run by AI without constant human checks and balances.

1.2 Problem in Low-Resource Chatbots

Low-resource languages present several challenges when it comes to building chatbots. There is a lack of chat corpus, to begin with. You are very unlikely to find a good chat corpus or other language-related resources for low-resource languages. A new corpus would have to be created somehow for any research to take place. For low-resource languages, there are usually high budget constraints compared to high-resource languages. Since creating a large chat corpus is expensive, generated dataset sizes are also usually smaller. Low-resource languages also struggle with code-mixing because of the influence of the English language.

So a modern chatbot for low-resource languages should be able to work with smaller datasets with few language-related resources. It also should be able to handle code-mixing and the overall solution should be cost-effective.

1.3 Problems with Tamil Chatbot

Tamils also suffer from the same challenges as the rest of the low-resource chatbots like lack of language-related resources, no publicly available corpus and codemixing. But in addition, it has specific challenges related to the language. First of all Spoken Tamil and written Tamil is different. In spoken Tamil also there are numerous dialects. There are people especially the older population who write Tamil the way they speak. So chat corpus needs to capture these variations to work effectively.

Tamil is a highly inflexional language. Therefore, high inflexions pose an additional challenge. Moreover, Tamil is a free word order language, which means its subject, object, and verb order can be freely changed. This is also a challenge. Even commercially available Tamil chatbots today are not very effective because of all these challenges. In some chatbots, the user is prompted to select input from predefined options instead of entering it directly. In some chatbots, users can ask questions in Tamil but only receive responses in English. There are only two previous research work for Tamil. The first system uses a rule-based approach using some own language-related resources but their performance wasn't objectively evaluated. The next approach uses the Dialog flow chatbot framework and google cloud and the performance was evaluated using human evaluators. The problem with this approach is it is a paid framework and the NLU part is a completely black box, we don't know what happens inside.

1.4 Problem Statement

To build a highly functional chatbot, a good corpus and a variety of language-related resources are required. Since Tamil is a low-resource language, neither any domain-specific corpus nor any standardised resources like dictionaries, dependency tree corpora, semantic databases, part-of-speech(POS) tagger, named-entity recognizer(NER) and Morphological Analyser are available for Tamil. Additionally, since Tamil is also a morphologically rich language, high inflexion and free word order pose key challenges[7] to Tamil chatbots.

Coding-mixing of Tamil and English by some users will be a practical challenge for building any Tamil chatbot. Currently, available Tamil chatbots dominantly suffer from these challenges even for a closed domain. Due to all the above reasons, it is evident that developing an effective End-to-End chat system is challenging even for a closed domain.

1.5 Motivation

Tamil is one of the national languages of Sri Lanka, the official language of Singapore and the State of Tamil Nadu, India. It is spoken by around 70 million people in India alone. But there is no effective chatbot system in Tamil to cater to this population. None of the available chatbot frameworks such as Rasa, Dialogflow, Microsoft bot framework, or Facebook Bot Engine supports the Tamil language. So by developing an effective End-to-End chatbot for Tamil, a huge range of businesses and organisations can reach out to the Tamil population in their native language. Any approach for developing such a chat system needs to be generalised so it can be useful not only for a particular domain or a particular language but to a wide range of domains and other similar low-resource languages.

1.6 Objective

This research focuses on identifying an effective approach to developing an End-to-End closed domain Tamil Chatbot.

- Evaluate approaches used for building a closed-domain chatbot.
- Recommend a suitable approach for low-resource languages like Tamil.
- Create a corpus related to the banking domain as a sample use case for training and evaluating the system
- Develop and evaluate the chatbot system which can answer questions related to the banking domain.

1.7 Research Contributions

This research has several contributions as listed below.

- A novel machine learning-based approach for building an end-to-end chat system for Tamil.
- First-ever intent annotated dataset for Tamil for the banking domain which will be made publicly available.
- State-of-the-art Tamil intent classifier model for short text inputs in the banking domain.
- Fully functional End-to-End Tamil Chatbot for Banking Domain.
- Two Research Publications
    - A peer-reviewed research abstract published in Technological Advances in Science, Medicine and Engineering Conference (TASME 2021), USA (Virtual).
        - Kugathasan, K., & Thayasivam, U. (2021, June). Suitable Approach for Building End-to-End Tamil Language Chatbot for Closed Domain. In Technological Advances in Science, Medicine and Engineering Conference 2021.
    - Full paper in peer reviewed Scopus indexed Conference - 5th International Conference on Natural Language Processing and Information Retrieval (NLPIR 2021), China (Virtual).
        - Kugathasan, K., & Thayasivam, U. (2021, December). Retrieval-based End-to-End Tamil language Conversational Agent for Closed Domain using Machine Learning. In 2021 5th International Conference on Natural Language Processing and Information Retrieval (NLPIR) (pp. 141-145).

1.8 Organisation

- Chapter 1: This deals with the introduction to chatbots, different types of chatbots, problems in developing chatbots in low-resource languages, what are language-specific problems in Tamil, problem definition, motivation, objective and research contributions.
- Chapter 2: Explains the different technologies used for building chat systems and how chatbots are evolved over time.
- Chapter 3: Discusses the latest approaches presented in the literature about building chat systems. This section provides an overall survey of the latest techniques used in English, low-resourced languages and Tamil.
- Chapter 4: Discusses the novel methodology used for building end-to-end systems in this research.
- Chapter 5: Discussing the dataset, the experiments carried out with different machine learning models and the chat web application developed.
- Chapter 6: This section talks about the results of machine learning models and chatbot test results.
- Chapter 7: Results of both the machine learning models and the performance of the chatbot are analysed in detail to find various insights.
- Chapter 8: This section talks about the conclusions of the research and the future works.

1.9 Chapter Summary

In this chapter, the discussion started by scrutinizing the limitations associated with employing FAQ pages and manual human agents as user support mechanisms. We then proceeded to delve into an examination of various chatbot categories. Furthermore, we addressed the intricacies linked to developing chatbots for languages with limited resources, highlighting the specific challenges encountered in the context of the Tamil language.

Collectively, these discussions culminated in the identification and delineation of the underlying problem. Subsequently, we transitioned into the realm of motivation, elucidating the driving force behind addressing this issue. Within this context, we elaborated on the project's objectives, shedding light on the distinctive research contributions that this endeavor brings to the fore. Lastly, we outlined the organizational structure of the subsequent chapters, providing an overview of the logical progression that readers can anticipate.

2. EVOLUTION OF CONVERSATIONAL SYSTEMS

The efforts to come up with a system that is capable of having a conversation started way back in 1960. As highlighted in Figure 10, over the years, several techniques, such as pattern matching, Markov chains, ontologies, AIML, chat script, and machine learning, have been adopted to enhance the sophistication of such systems.

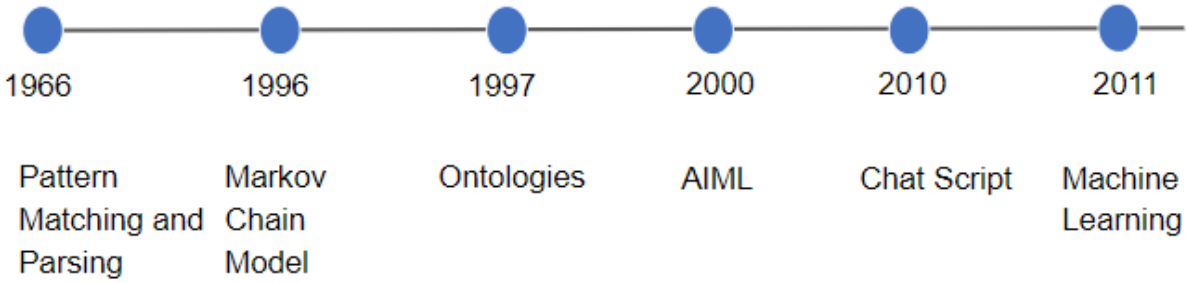

Figure 10: Evolution of Chat technology

2.1 Pattern matching

The pattern matching approach was the first and most common approach to building chatbots in the early days of chat systems when they were used primarily for just answering questions. Almost all chatbot systems that exist today use some variant of some pattern-matching algorithm in order to detect distinct patterns. In spite of the fact that there are a variety of approaches to pattern matching, the basic idea remains the same regardless of how complex those approaches may be. Pattern-matching algorithm is widely used in chatbots that operate within a close domain. Figure 11 depicts the description of a simple pattern-matching system.

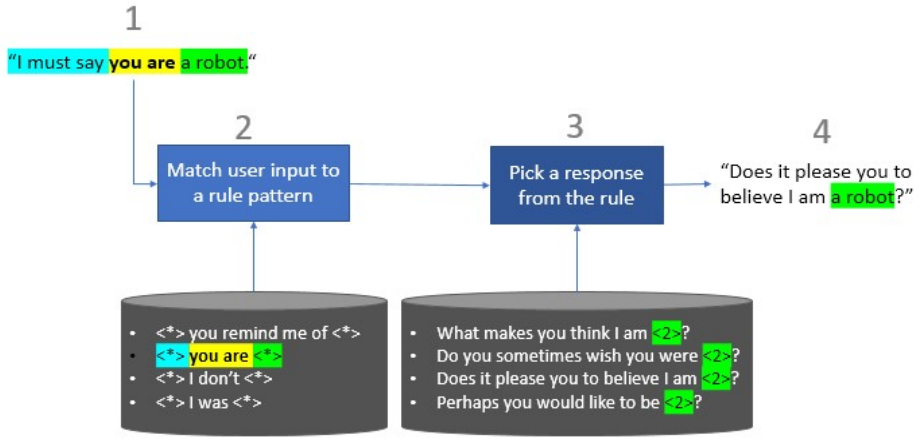

Figure 11: Pattern matching system

As shown in this example, despite the simplicity of the approach, it still produces results that are good enough to cause confusion as to whether this is a computer or a human performing the operation. One of the earliest chatbots to be developed was ELIZA in 1966 by the MIT Artificial Intelligence Laboratory. It was designed in an attempt to mimic the role of a psychologist by using pattern matching and responding with messages that are converted into questions based on the user's input. Figure 12 illustrates a conversation with the Eliza chatbot.

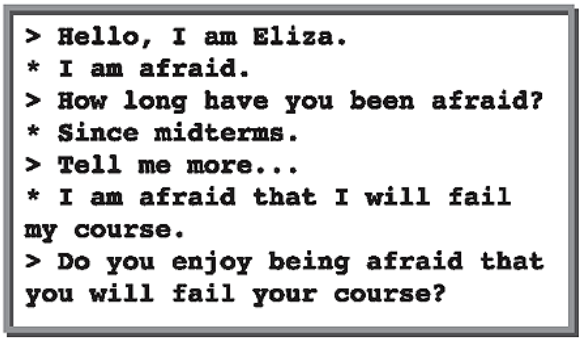

Figure 12: Eliza chatbot

It examines each word in the user input and return the word that has highest keyword ranking. Then check each rule for the word in ranked order and choose the very first pattern that matches and applies transformation. If there is no keyword match, it just replies “Please go on”, That’s very interesting, I see”.

2.2 Parsing

Parsing textual data involves converting input text into tokens (lexical parsing) with features, mostly to identify the underlying grammatical structure. In addition to that, the lexical structure can also be checked to determine whether they form an allowable expression (syntactical parsing). Figure 13 illustrates how parsing works.

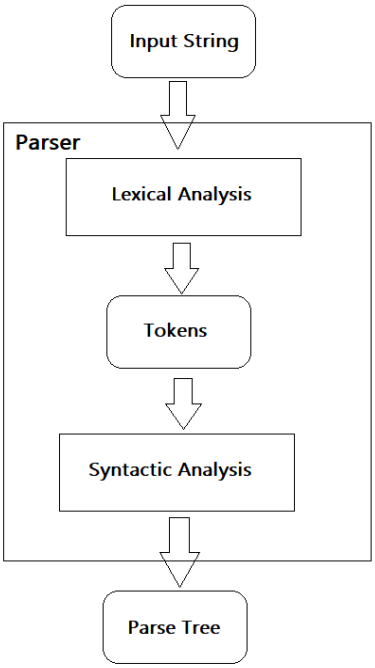

Figure 13 : Parsing

Parsers of earlier times were very simple, searching for recognisable keywords in the order in which they should occur. For example, if we were parsing the sentences "please take the umbrella" and "can you get the umbrella?", both would be parsed as "take umbrella". This approach allows a chatbot with a limited set of patterns to cover multiple input sentences. It is worth noting that even ELIZA, the chatbot we discussed earlier, used parsing to identify keywords in the user's input. As chatbots evolved, more complicated parsers performed complete grammatical parsing of natural language. Figure 14 shows a parse tree acquired through the parsing of the input string.

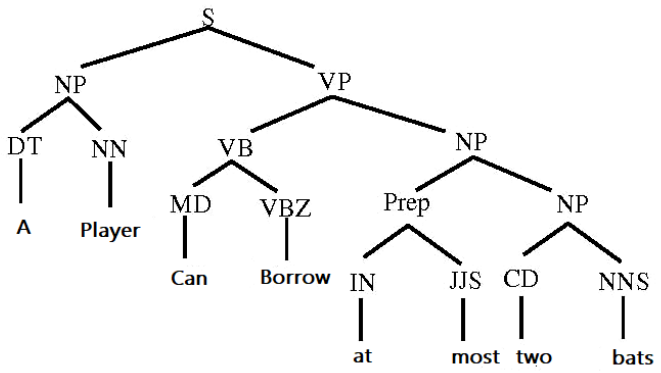

Figure 14 : Parse Tree

2.3 Markov Chain Models

The Markov Chain Model is used in the case of chatbots in order to construct responses that would be more probabilistically viable and therefore more accurate than those constructed by conventional algorithms. The idea behind Markov Chain Models is that they are based on the assumption that each letter or word occurrence in some textual dataset occurs with a fixed probability.

Model order refers to how many consecutive occurrences the model takes into consideration. According to the Markov model of order 0, for example, if the input text is ‘helloworld’, it is predicted that the letter "o‟ occurs with a probability of 0.2 (2/10). In the order 1 model, each letter would still have a fixed probability of occurrence, but that probability would depend on the letter before it. In the case of the order 2 model, the probability of a letter would depend on the two letters before it. The HeX[20] chatbot is an excellent example of how Markov Chain Models can be used to generate responses for automated applications. Figure 15 shows how a Markov Chain Model works.

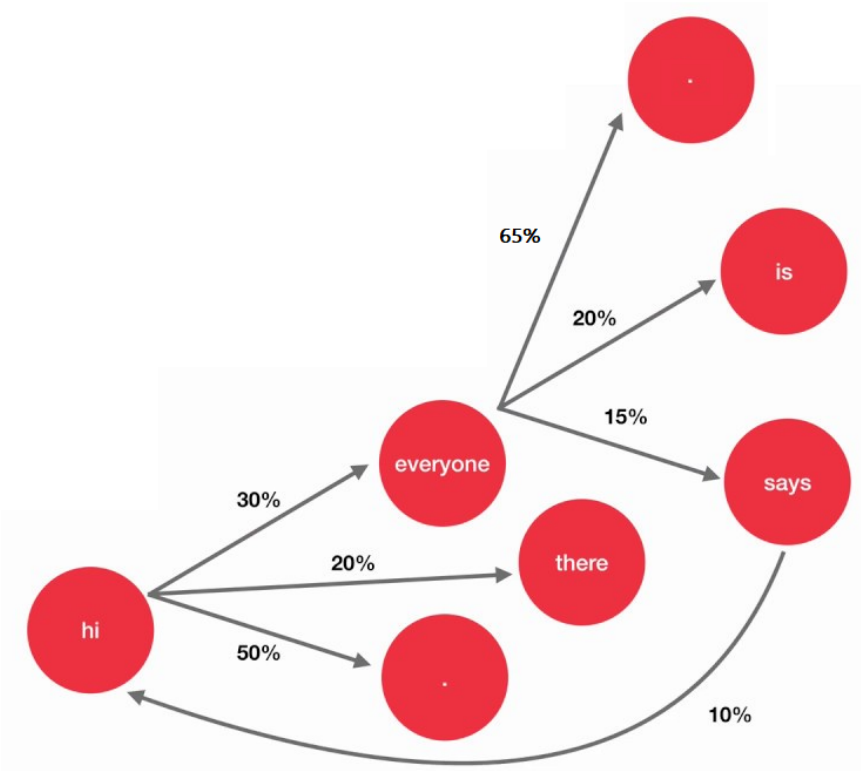

Figure 15 : Markov Chain model

2.4 Ontologies

An ontology, or a semantic network, is a set of concepts that are interconnected in a hierarchical and relational way. According to [21] the main reasons for developing an ontology are as follows:

- People or software agents need to be able to understand the structure of information in a common way.
- Making it possible for domain knowledge to be reused.
- Explicitly identifying the domain assumptions that need to be made.
- In order to separate domain knowledge from operational knowledge
- Analyzing domain-specific knowledge

In the context of chat applications, ontologies offer a platform to conduct semantic analysis and, very importantly, the ability to define some measure of semantic closeness. This is done based on a graph with concepts as nodes, and the relations between them as arcs. As a result of semantic distance measurements, we can identify groups of concepts that have similar meanings, and therefore, we can identify lexical chains of related words. A chatbot can use these concepts directly to figure out hyponyms, synonyms, and other relationships between them.

An ontology can be categorized into two types: general ontologies and ontologies specific to a particular domain. As part of a general ontology, we develop an extended vocabulary that provides extra-linguistic information, such as related words or case grammars. WordNet and FrameNet are examples of general ontologies. There are more than 200 languages represented in the WordNET lexical database, which contains semantic relationships (i.e. synonyms, hyponyms, and meronyms) between words. Essentially, WordNet is a combination and extension of a dictionary and a thesaurus [21].

FrameNet is a lexical database of English that can be read by both humans and machines, based on annotating examples of word usage in actual texts. It contains a dictionary that contains more than 13,000-word senses, many of which include annotated examples, which

demonstrate the meanings and usages of the words. In addition, it has 200,000 manually annotated sentences, which are linked to more than 1,200 semantic frames that describe the sentences[21].

When modelling a specific domain according to its own rules, an ontology that is domain-specific will be used. It indicates how terms in that domain are used in particular contexts, according to their particular meanings. For example, let's take the word "windows". The ontology about the domain of computers would model the word with the meaning "operating system" and "graphical control element." A couple of examples of early chatbots which used ontologies include OpenCyc and CONVERSE[20].

2.5 Artificial Intelligence Markup Language (AIML)

The syntax of Artificial Intelligence Markup Language (AIML) is based on XML, and it is largely composed of input rules and output rules corresponding to the input rules. In chatbots, it is used to define the flow of conversation between the user and the agent. A chatbot named A.L.I.C.E (Artificial Linguistic Internet Computer Entity) [20] was developed as the first chatbot to use AIML. In AIML, categories are the key units of knowledge since they form the building blocks of knowledge. It is important to note that categories comprise at least two additional elements: patterns and templates. As the name implies, a pattern is a string of characters that is expected to match one or more inputs from users. It is important that the pattern includes everything in the input and it is not case-sensitive. If you wish to exclude a particular word from the search simply use a wildcard (*) which binds to one or more words. When a pattern is matched, a template specifies what the response should be. Figure 16 illustrates the fundamental building block of AIML.

```
<category>

<pattern>I want help</pattern>

<template>Sure. How can I help you?</template>

</category>
```

Figure 16 : AIML code block 1

The reason why AIML is so successfully used for building chatbots is because of its ability to call itself recursively. It can submit input to itself using the <srai> tag. Element “stri” is an abbreviation for Symbolic Reduction in Artificial Intelligence. To reply to synonymous input, this would be extremely useful.

Another key element in AIML is the star element. The star element is used to echo portions of the user’s input that were captured by wildcards (*) through the <star/> tag. When multiple wildcards are present in the pattern, the index is used to specify which wildcard should be echoed. As <star index="1"/> is most commonly used, it is usually written as <star/> to save time. Figure 17 illustrates how the <srai> tag can be employed to redirect to another category.

```
<category>
<pattern>I want help</pattern>
<template>Sure. How can I help you?</template>
</category>
<category>
<pattern>Can you do me a favour?</pattern>
<template> <srai>I want help</srai> </template>
</category>
```

Figure 17 : AIML code block 2

As it can be seen in the Figure 17, the input “Can you do me a favour?” has the same meaning as “I want help”, so the <srai> tag is used to specify that it should be redirected to the category that matches the input "I want help". In order to avoid redundancy, this approach is helpful. Figure 18 demonstrates how wildcard handling is performed in AIML.

```
<category>
<pattern>My name is * and I'm * years old </pattern>
<template>Hello! <star />. I'm also <star index="2" /> years old!</template>
</category>
```

Figure 18 : AIML code block 3

There are two wildcards in the input in Figure 18, one for the name, and one for the age. In the response we are using <star/> to echo the first wildcard and <star index="2" /> to echo the second wildcard. Therefore, if the user enters "My name is John, and I'm 22 years old", he will receive the response from the system "Hello! John. I'm also 22 years old!".

2.6 Chat Script

ChatScript is regarded as the successor to AIML in the field of chatbots. With ChatScript, you are able to maintain your application with ease as well as add various features like concepts, continuations, logical and/or, variables, fact triples, and functions. These added functionalities attempt to address the need for ontologies within the script itself [20]. A ChatScript file is stored in plain text format, and it comes with more than 2000 predefined concepts and more concepts can be created by scripters.

The ability of ChatScript to maintain user state across multiple volleys is one of its most notable features. There are a number of steps involved in each volley, including accepting input from a user, loading data about the user and user state, computing a response, writing out a new state and sending the response to the user. As a result, this allows for an interactive conversation between the system and the user.

A topic is a collection of rules that constitute the basic component of ChatScript's code. A rule consists of a pattern and code. The rules within a topic are considered in turn by comparing their pattern components. In addition to being able to access global data and the user's input, patterns are also capable of comparing input data and memorizing portions of data. Upon failure of the pattern, consideration is given to the next rule in the topic. In the event that a pattern is successful, the rule's code section is executed. Figure 19 illustrates the fundamental building block of ChatScript.

```
topic: ~fastfood

t: Hello! This is our online fast food. Please make your order.

u: BURGER (I want a chicken burger) Okay, your order is hamburger.
```

Figure 19 : Chat script

As you can see from Figure 19 ChatScript code, the topic has been declared using the keyword (topic:). The topic name should be defined with “~” at the start. "fastfood" is the name of the topic in this example. In the next line, we see the rule type "t:", which is called a "gambit". Gambits allow the bot to steer a conversation within a topic. In order to control the flow of conversation, the chatbot relies on this mechanism.

In this case, the bot is designed to take orders for fast food, so it starts the conversation immediately by saying "Please make your order". As a result of this manoeuvre, the user is limited to responding to a particular topic that interests the bot. After this, the user is supposed to respond with the item he wants to order so the bot needs to acknowledge the order and respond back. The next step is to use the "responder" rule type. There are four main components to a rule: the rule type, the label, the pattern, and the output.

In this case, the rule type "u:" specifies that the rule is a response to either a question or a statement from the user. The rule is labelled "BURGER". Although a rule label is optional, it's always helpful to have one when documenting your code in case you need to debug it. A pattern is something you expect the user to say. The rule will not be executed if the input does not match the pattern. It is always necessary to place the expected pattern inside brackets. According to this example, the rule needs to be executed when the user says "I want a chicken burger.". Once the pattern is specified, we specify what the output should be when the pattern is matched. Based on the example above, if the pattern matches, the bot will say "Okay, your order is hamburger".

2.7 Machine Learning

Machine learning is the current state-of-the-art technique used for developing chat systems. It is used either alone or alongside, some of the techniques mentioned above to improve the quality of chatbots. As chatbots learn and adapt through experience, machine-learning techniques improve their performance. With each interaction, the chatbot will likely be able to better understand the content and context of the user's input, resulting in more accurate, relevant responses[22]. The current trend in chatbot development indicates that chatbots will continue to improve with advanced technologies driven by machine learning. Machine learning based system will be explore more in the next chapter.

2.8 Chapter Summary

This chapter provides an overview of the evolving landscape of chatbot technologies over time. It begins with an exploration of pattern-matching algorithms, highlighting their continued effectiveness in present-day closed-domain chatbot applications. The discussion then shifts to parsing algorithms and their role in generating parse trees. Additionally, the chapter delves into the utilization of Markov Chain Models to enhance response construction through probabilistic viability.

Further, the exploration extends to ontologies, where a graph structure facilitates measuring semantic distances and identifying clusters of related concepts, leading to the recognition of interconnected lexical chains. The chapter proceeds by examining AIML (Artificial Intelligence Markup Language) and its simplified approach to chatbot development, followed by an introduction to ChatScript as a successor to AIML.

Concluding the analysis, the chapter emphasizes the pivotal role of machine learning in contemporary chat system construction. It emphasized that a thorough and comprehensive study of the literature is essential to fully comprehend the intricacies of machine learning-based chat systems.

3. LITERATURE REVIEW

First approaches related to developing chatbot in English were studied. Then prominent approaches used in similar low resource languages were studied. Then chatbot research related to Tamil language was studied. Techniques used for dataset creation were then explored, short text classification. Finally, evaluation metrics used in chat systems were studied.

3.1 English Language Chat Systems

There are several approaches to building end-to-end chat systems. A generative chatbot is designed in two ways. One way is a components-based chatbot where Natural Language Understanding(NLU) Module, Dialog Manager and Natural Language Generation(NLG) Module are trained separately and integrated together[10]. This has been the prominent approach for high-resource languages until recently. But this approach has several problems such as the requirement of the expensive labelled dataset, a large amount of handcrafting multiple components and the effects caused by individual components due to error propagation[11].

So the current research trend moves more towards an End-to-End trainable system. One prominent strategy that is been currently explored is to convert chat problems as a sequence to sequence mapping problems which is modelled by sequence to sequence architecture. This kind of early research is carried out only in goal-oriented systems. This kind of approach are not very suitable for our research since dataset collection will be expensive. [23] had spent 400 USD for data collection alone. Additionally, our research focuses on only non-task-oriented systems.

3.2 Low Resource Chat Systems

Low-resourced language effort of end-to-end text-based conversational agents use a mostly retrieval-based approach. The majority of the latest efforts convert the chat problem as a short text intent classification problem and try to retrieve the answers based on the intent classified. The accuracy is the system depends on the accuracy of the intent classification.

A conversational agent developed for the Thai language [8] has collected 2,636 question-and-answer pairs related to the banking domain and categorized them into 80 classes. Question classification accuracy on the test set was 83.9% using RNN-LSTM.

Chat system developed in the Bengali language [12] for a consumer electronics site used 91 classes, and 1731 queries in total with 5 – 10 queries per class. It recorded the highest recall of 89.08% using CNN. These two efforts show it is possible to achieve high accuracy/recall even with a small dataset and by classifying the user intent correctly using machine learning, the answer retrieval can be improved.

3.3 Tamil Language Chat Systems

Among available Tamil chatbots, the "Poongkuzhali" chatbot uses a rule-based retrieval approach [9]. But the system performance wasn't objectively evaluated. The authors just claimed that the users who tested the system found the system to be satisfactory. The dataset used for this chatbot was not made publicly available. Apart from this, there is no research done for Tamil language chatbots. Figure 20 illustrates how Poongkuzhali works.

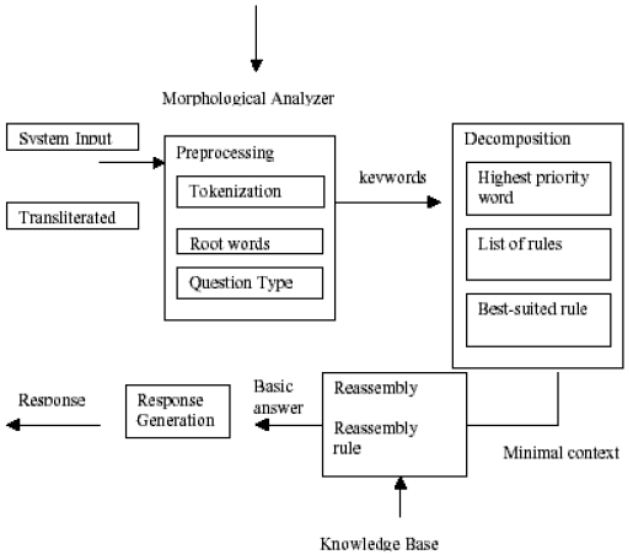

Figure 20 : Poongkuzhali Chatbot

“Cody” [13] and “Machan” [14] are two other corporate chatbots available in Tamil. Figure 21 and Figure 22 illustrate the problems in the Cody and Machan chatbots, respectively.

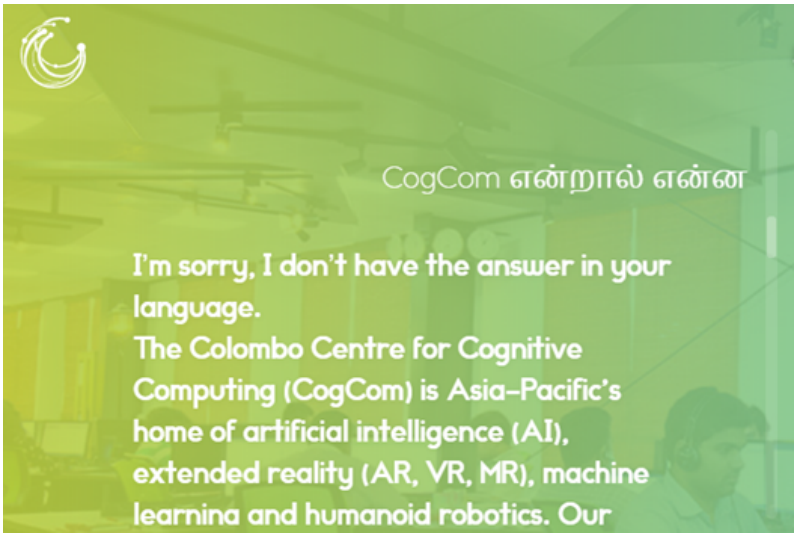

Figure 21 : Cody Chatbot

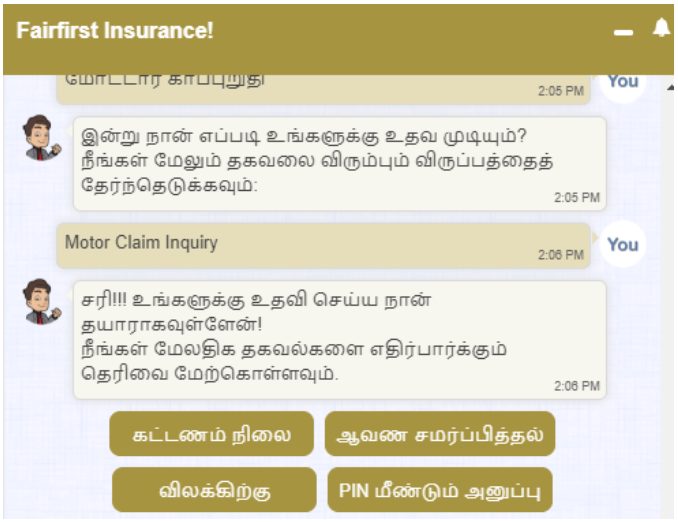

Figure 22 : Machan Chatbot

“Cody” can take input in Tamil but can only answer in English. This means the NLU component is working well whereas NLG part is yet to be perfected. “Machan” chatbot doesn’t allow the user to enter input directly instead it prompts the user to select it from predefined options. This approach is very much detrimental as far as user experience is concerned.

3.4 Techniques used for dataset creation

When it comes to NLP research on a low-resourced language such as Tamil, one of the key challenges is identifying a suitable dataset. Given the scarcity of pre-existing datasets, researchers often need to create their own dataset.Currently, there are no existing datasets for Tamil that can be utilized for developing a chatbot.As a result, in order to carry out research on Tamil chatbots, a dataset needs to be created.

There are three possible methodologies that can be used to generate the required dataset.

- Oblique translation
- Wizard-of-Oz paradigm
- Utterance generation using the native speakers

Oblique translation is a commonly employed technique in translation when the fundamental or abstract aspects of the source language cannot be conveyed accurately to the target language without altering the meaning or disrupting the grammatical structure[15]. An example of oblique translation from English to Tamil is provided in Table 1.

The city “Los Angeles” is translated to “**யாழ்ப்பாணம்**” and the “baseball game” is translated as “**துடுப்பாட்ட போட்டி**” since those were better suited in the Tamil context.

Table 1: Sample English to Tamil Obligue Translation

English Sentence	**Tamil translation**
Tell me the weather report for Los Angeles.	யாழ்ப்பாணத்தின் தற்போதைய வானிலை அறிக்கையை எனக்கு அறியத் தாருங்கள். Yāḻppāṇattiṉ taṟpōtaiya vāṉilai aṟikkaiyai eṉakku aṟiyat tāruṅkaḷ.
What time does today's baseball game start?	இன்றைய துடுப்பாட்ட போட்டி என்ன நேரம் ஆரம்பிக்கிறது ? Iṉṟaiya tuṭuppāṭṭa pōṭṭi eṉṉa nēram ārampikkiṟatu

When it comes to Tamil, carrying out word-for-word or direct translations from a dataset in a highly-resourced language can prove challenging due to cultural nuances and language subtleties. However, utilizing oblique translation is a viable option for creating a dataset in Tamil, particularly if a high-quality English dataset is available. This approach can leverage datasets that are more readily available in a high-resource language to develop a dataset for Tamil.

To construct a task-oriented chatbot for the restaurant domain, a unique data collection methodology was introduced based on the Wizard-of-Oz paradigm [16] and crowd-sourcing. This approach segments users into two groups, with one group serving as customers who must generate suitable sentences to accomplish a task based on provided task descriptions and dialogue history. The second group of users assumes the role of Wizards, responsible for slot filling in response to user queries and typing a reply based on dialogue history and database findings. Repeating this process with various users yields a significant dataset. However, a key disadvantage of this approach is the requirement for software system to facilitate the data generation process, making it an expensive and time-consuming option.

When developing a Telugu language chatbot[17], the authors first identified 12 distinct intents and then created 68 potential patterns that users might use to express these intents. This methodology is useful to the Tamil language as well, given its close relationship with Telugu, as both belong to the Dravidian language family.

3.5 Short Text Classification

The intent is defined as a purposeful action[1]. Intent Classification can also be considered an essential first step in various other tasks like building conversational agents. Much prior work in intent classification addresses the challenge of understanding queries obtained from search engine logs and large text documents. Short text classification has been widely studied since the recent explosive growth of online social networking applications[1]. Short text classification is particularly challenging compared to large text documents since large text documents provide more explicit information whereas, in short, text classification, the information is implicit.

Major attempts to tackle the problem are to expand short texts with knowledge extracted from the textual corpus, machine-readable dictionaries, and thesauruses[2]. Despite the versatility of dictionaries and thesauruses, they are often domain-independent. As a result, the data distribution of external knowledge frequently differs from the test data gathered from a particular domain, leading to a decrease in the overall performance of categorization.

Deep learning techniques like CNN have been considered state-of-the-art for text classification. Many authors have attempted to apply CNN[4], attention-based CNN[5], bag-of-words-based CNN[6], and the combination of CNN and recurrent neural networks [7] to text classification. Most of the prior research on short text classification mainly focused on binary classification or fewer classes. Multi-class classification with a higher number of classes is difficult. Optimising a multi-class classification is data and domain-dependent. Especially for low-resourced languages, this is an open and challenging problem due to a lack of data and other related language resources.

One approach to tackle multi-class classification with a higher number of classes is using Hierarchical multi-label classification. This approach classifies multiple labels for a text, where each label is part of an underlying hierarchy of categories. Hierarchical multi-label classification can be mainly classified into local and global approaches[3]. Local approaches involve creating a distinct classifier for every node, parent node, or level in the taxonomy, whereas global approaches entail constructing a single classifier for the whole taxonomy. The local approach will result in parameter explosion and it is computationally costly. Whereas in a global approach, The classifier constructed is not flexible enough to cater for changes to the category structure.

But adopting hierarchical multi-label classification also poses additional challenges as we move down the hierarchy since lower-level hierarchies are fine-grained compared to the upper-level of categories. It is common for the quantity of training data in a lower level to be considerably less than that in an upper level, leading to a reduction in the overall performance of classification. Short text Multi-label Text Classification is extremely challenging and almost all current approaches have equal or worse accuracy compared to using flat classifiers. Only a few notable works for Short text Multi-label Text Classification exist even for highly resourced languages and there have been no previous low-resource efforts.

3.6 Evaluation Metrics

There are both qualitative and quantitative methods to analyse a chat system. Early chatbot research [24] mostly relied on qualitative approaches like user reviews, user acceptance and etc. Later quantitative methods become prominent. The suitable quantitative methods will be different based on the requirement and the taxonomy of the chat system. For example, goal-oriented systems evaluate the success of the system using task completion rate, the time taken for task completion[25]. But this approaches are not suitable for non task-oriented systems. In non task-oriented retrieval based systems, the most used approaches are accuracy and recall@k[24]. Here the accuracy of the system is taken as the accuracy of the classification model since the answer retrieval is solely dependant on intent classification. In Recall@k, the system is asked to select k most likely responses for the user input and if the true response is among the k candidates, the response is considered as correct. When it comes to low resource language results both recall@1 [12] and accuracy [8] are used.

3.7 Chapter Summary

In this chapter, two primary strategies for constructing English language chatbots were first investigated. However, it was established that these approaches prove unsuitable for Tamil due to the requirement of costly labeled datasets and language-related resources. Subsequently, a deeper investigation was undertaken into low-resource languages, revealing that efforts in creating end-to-end text-based conversational agents for such languages primarily adopt a retrieval-based approach. Recent endeavors predominantly transform the chat problem into a short text intent classification task, subsequently retrieving answers based on the classified intent. These efforts have demonstrated the potential to attain very good accuracy and recall even with limited datasets.

The subsequent investigation focused on prevalent challenges in current Tamil language chat systems. We then explored three key strategies for dataset generation, and within the context of Tamil short text classification research, we revealed the ongoing complexity in creating a multi-class classification system. At the end, this chapter culminated in an extensive review of diverse evaluation metrics employed to evaluate chatbots. Notably, accuracy and recall emerged as the mostly used metrics for evaluating their performance.

4. METHODOLOGY

The research aims to devise a system for creating a chatbot based on FAQ data available on the banking domain's web pages in Tamil. Our approach consists of three main steps. The first part is the dataset generation, the second part is the intent classification, and the third part is building an end-to-end chatbot which takes queries from users in Tamil and responds to them. In this research, we try to demonstrate that it is possible to build an effective closed-domain chatbot for a low-resource language by transforming the chat problem into an intent classification problem and retrieving the answer based on intent classification. We can build functional chatbots with this method even without resources such as dictionaries, dependency tree corpora, semantic databases, part-of-speech taggers, named entity recognizers, and morphological analyzers. Although this project is done for the Tamil language and for the Banking domain, this approach can be applied to other low-resourced languages and domains as well. Figure 23 outlines the comprehensive methodology of this research.

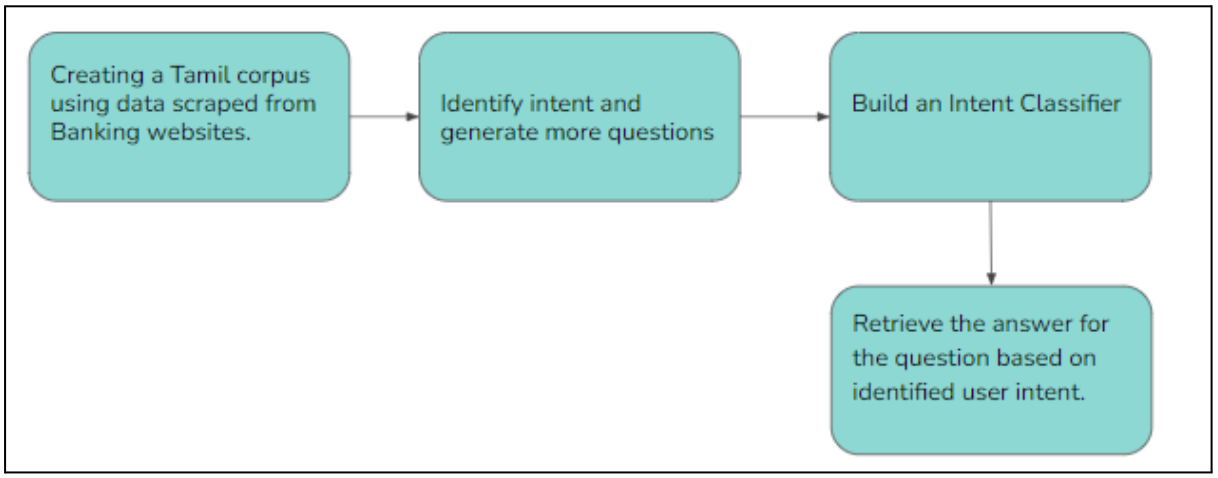

Figure 23 : Methodology

4.1 Dataset creation

The dataset for this study was created by improvising on the method used by the Telugu chatbot [17]. In order to generate the Tamil FAQ corpus, the FAQs were scraped from the websites of two Tamil banks.

An intent was assigned to each question after the scraped dataset has been analyzed. Intents and questions in the FAQ corpus are mapped one-to-one. For example, when ten FAQ questions were scraped, those ten questions would be mapped to ten intents. This is because there will be no redundant questions on the FAQ pages of the bank. Then, several native speakers are used to generate more examples for each intent. There are two ways in which a user can interact with the chat system.

Either by making a statement or asking a question. In the case where a user wants to know the interest rate on a fixed deposit, for instance, he may either ask a question “Can you please tell me the interest rates for fixed deposits?” or he may pose a statement such as “I would like to know the interest rates for fixed deposits”. Native speakers were therefore asked to generate as many questions and statements as possible related to each intent using different inflexions. Due to Tamil's free word order, more variations of the same sentence are also required considering the free word order. Due to the fact that code-mixing in Tamil and English is a common issue among Tamil people, it is also very crucial that the dataset represents this fact as well. There is a difference between Tamil in its spoken form and Tamil in its written form, although some people write Tamil as they speak it. It is also important to ensure that the dataset also captures this variation.

After the data set generation task, the generated dataset was checked for spelling errors and any errors that were found were promptly corrected. Since different native speakers were used to generate more variations for each intent, there were several duplicate questions/statements found for each intent when compiling them together. Those duplicate questions were removed from the data and the dataset was finalised.

4.2 Intent classifier

First several preprocessing steps were carried out such as tokenising and removing the stop words. Then several vectorising techniques were tried such as Bag-of-words, TF-IDF and Fasttext word embedding. Several techniques for tackling class imbalance undersampling, oversampling, SMOTE and adding class weights were also attempted.

Then the database was split into test and train and different machine learning models were tested and the accuracy of such models was noted. The best-performing classifier was selected using the highest accuracy value.

4.3 End-to-End chatbot development

Then a web-based chat application was built using Python to interact with the user. The user can enter his query and the chatbot will send that query to the backend where the query is preprocessed and intent is identified by passing it to the previously trained model. then the answer for that particular intent will be sent to the front end and the bot reply with the answer.

4.4 Chapter Summary

This chapter provided an in-depth explanation of the research methodology adopted. The approach encompasses three primary stages. The initial step involves dataset generation, followed by intent classification, and culminating in the development of a comprehensive end-to-end chatbot designed to receive and respond to user queries in the Tamil language.To generate the Tamil FAQ corpus, a two-fold process was employed. FAQs were initially scraped from the websites of two Tamil banks, which served as the foundational dataset. Subsequently, native Tamil speakers were engaged to further enrich the dataset by generating additional utterances. An intent classifier was then constructed, with the selection of the optimal classifier being based on the highest recorded accuracy.

The subsequent phase involved the creation of a web-based chat application utilizing Python. Users input their queries into the interface, and the chatbot transmits the query to the backend for preprocessing. The previously trained model is employed to identify the intent of the query. Once the intent is determined, the corresponding response is sent to the front end, facilitating a seamless interaction between the user and the chatbot.

5. EXPERIMENTS

In our experiment and evaluation, we aim to determine if a functional chatbot can be built using intent classification on a small dataset.

5.1 Dataset

Participants in the data generation task were 12 native language speakers between the ages of 18 and 19. Participants in the data generation task were given 10 intents along with a few sample statements/questions for each intent. One intent was assigned to at least two participants. This enabled us to capture more inflectional variation for the same intent. But this also means a lot of time was spent while compiling the data since duplicate questions/statements need to be removed to avoid redundancy in data. After cleaning the data, the final dataset consists of 1567 examples for 56 intents. The minimum number of examples per class was 10. The distribution of examples per class in Figure 24 reveals the presence of a class imbalance problem in the dataset.

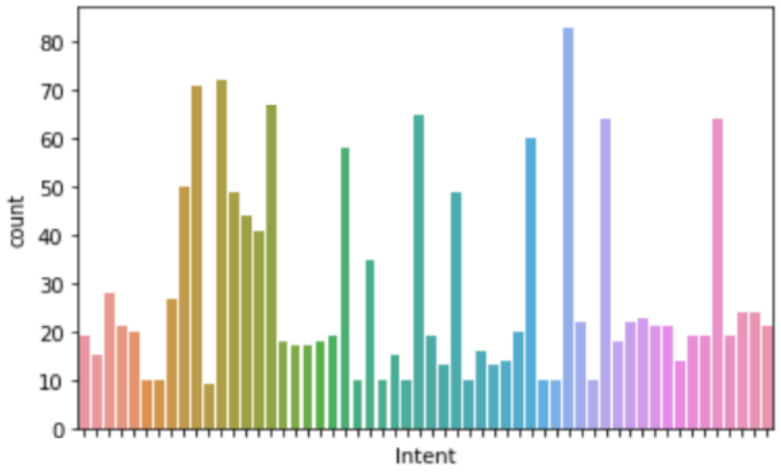

Figure 24 : Class imbalance

The dataset set was highly imbalanced because for some intent only fewer inflexions were possible. Therefore, participants could only generate a few variations, while for simpler intents, more inflexions were generated. For many intents the participants fail to balance between spoken, code-mixed and written Tamil, so written Tamil was most frequent in the dataset followed by code-mixed examples and finally only a few spoken Tamil examples.

In Table 2, you can observe a selection of sample sentences obtained from the data generation task.

Table 2 : Sample sentences from dataset

Tamil Sentence	English Translation	Observation
எனது கடனட்டையை நான் எப்படி இரத்து செய்வது? Eṉatu kaṭaṉaṭṭaiyai nāṉ eppaṭi irattu ceyvatu?	How can I cancel my credit card?	This sentence is in the written form of Tamil.
வதிவுள்ள வெளிநாட்டு நாணய கணக்கொன்றை எப்படி ஆரம்பிப்பது எண்டு சொல்ல ஏலுமோ? Vativuḷḷa veḷināṭṭu nāṇaya kaṇakkoṉṟai eppaṭi ārampippatu eṇṭu colla ēlumō?	Can you tell me how to open a resident foreign currency account?	This sentence is in the Spoken form of Tamil.
எனது BOC debit card **எங்கு பாவிக்கலாம் என்ற விபரங்கள் வேண்டும்.** Eṉatu BOC debit card eṅku pāvikkalām eṉṟa viparaṅkaḷ vēṇṭum.	I need details on where to use my BOC debit card.	Example of Tamil-English code-mixed sentence.
ரெசிடெண்ட் போறின் அக்கௌன்ட் ஸ்டார்ட் பண்ணுவது எப்படி? Reciṭeṇṭ pōṟiṉ akkauṇṭ sṭārṭ paṇṇuvatu eppaṭi?	How to open a resident foreign account?	Example of Tamil-English code-mixed sentence written using only Tamil alphabets.

5.2 Machine Learning Models

For vectorizing the words, Bag of Words (BOW), TF-IDF, word2vec and fastText word embeddings were tried. Several approaches were experimented with to tackle class imbalance problems like random oversampling, random undersampling, SMOTE, combining random oversampling, undersampling and adding class weights. The Hold Out method was employed to assess the model's performance. The dataset was divided into 80% and 20% for training and testing. Machine learning models were trained with 80% of the data and tested with 20%. Several machine learning models such as SVM, Bi-LSTM, and CNN were tried with different settings and the best settings were found empirically.

5.3 Chat Application

The chat application was manually tested for robustness using several queries ranging from both queries that are part of the dataset and new queries outside the dataset. The new queries were used to check whether the chatbot was able to answer queries that are not part of the dataset with the same effectiveness. We also tested code-mixed queries, queries with spelling mistakes and queries with spoken Tamil. Figure 25 shows the developed chat application.

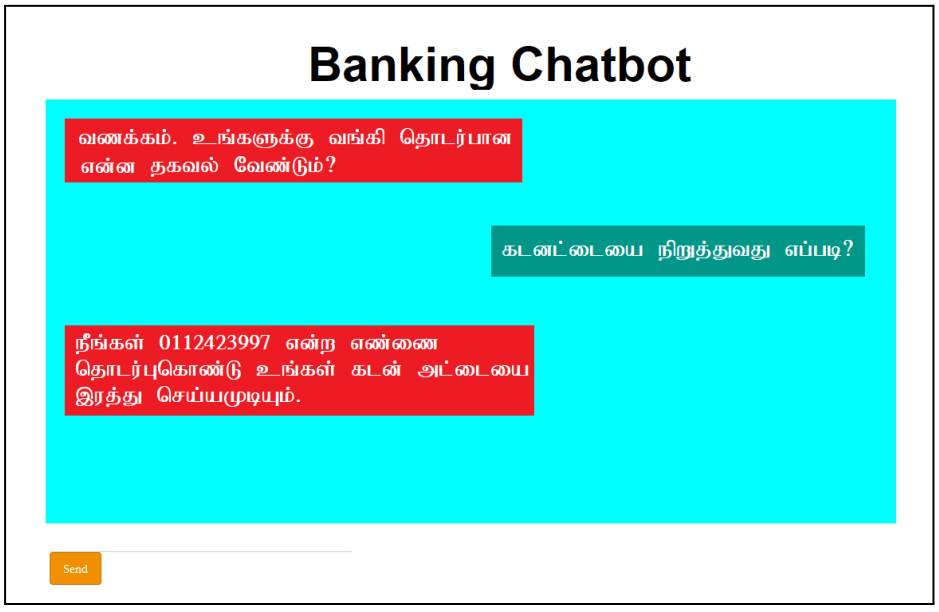

Figure 25 : Developed Chat Web Application

5.4 Chapter Summary

In this chapter's initial phase, an analysis was conducted on the generated dataset, comprising 1567 instances spanning 56 intents, originating from the collaborative efforts of 12 native speakers. A subsequent observation pertained to the existence of a class imbalance issue within the dataset. Subsequent sections delved into the development of machine learning models, encompassing a comprehensive exploration of varied vectorization techniques, as well as the implementation of diverse strategies to mitigate class imbalance. Multiple machine learning algorithms were subjected to experimentation. Finally, we presented the developed end-to-end chatbot and outlined the testing procedure.

6. RESULTS

Both the model performance and how well the chatbot answers user input were both evaluated.

6.1 Model Performance

Table 3 provides a comprehensive overview of the accuracy and recall values achieved by the various considered models.

Table 3 : Model Performance

Model	Thai chatbot (RNN-LSTM) [8]	Bi-LSTM (Embedding layer - 300, units 128) with BOW	SVM with TF-IDF	Bi-LSTM fastText (Embedding Size-300, units 128)	CNN + fastText with class weight
Accuracy	83.90%	94.26%	96.40%	97.48%	98.72%
Recall	N/A	91.37%	94.08%	96.73%	98.45%

The Bi-directional LSTM model, equipped with an embedding layer of size 300 and 128 units, along with Bag of Words for vectorization, demonstrated an accuracy of 94.26%. This remarkable performance signifies a substantial accuracy improvement of over 10% compared to the Thai chatbot's accuracy.

Utilizing the same settings but employing fastText for vectorization, the Bi-directional LSTM model achieved a significantly higher accuracy of 97.48%, showcasing an impressive accuracy gain of more than 3%.

The SVM model, implemented with TF-IDF for vectorization and a linear kernel, yielded a notable accuracy rate of 96.40%, reflecting its strong performance on par with neural network models.

The CNN model, which utilized fastText for vectorization and incorporated class weights to address class imbalance, showcased highest accuracy, reaching an impressive value of 98.72%.

A similar pattern is evident when examining the Recall values across the models. The Bi-directional LSTM model with an embedding layer of size 300 and 128 units, coupled with Bag of Words (BOW) for vectorization, demonstrates a recall of 91.37%. The SVM model, leveraging TF-IDF for vectorization and a linear kernel, achieves a higher recall of 94.08%.

The Bi-directional LSTM model utilizing fastText for vectorization, maintaining an embedding size of 300 and 128 units, outperforms with a recall of 96.73%. Remarkably, once again the CNN model integrated with fastText for vectorization and class weight adjustments attains an impressive recall of 98.45%. Since the Thai chatbot [8] did not provide any recall values, a comparison of our models' recall values with theirs is not possible.

In direct comparison to the performance of the Thai chatbot [8], which employed an RNN-LSTM-based model, using accuracy values, all of our developed models have consistently delivered remarkably high results. A detailed discussion about the model's performance is carried out in Chapter 7.1.

6.2 Chatbot Performance

The robustness of the chat application was thoroughly evaluated through manual testing, encompassing a diverse range of queries. These queries spanned not only those included within the dataset but also entirely new inquiries that were outside the dataset. This comprehensive approach aimed to determine the chatbot's ability to effectively respond to questions beyond its pre-existing knowledge base. Furthermore, the testing process involved examining the chatbot's performance when faced with code-mixed queries, those containing spelling errors, and even those articulated in spoken Tamil. By subjecting the chatbot to these varied scenarios, we sought to ascertain its capability to handle real-world conversational complexities and challenges, thereby gauging its robustness and versatility. Table 4 shows some of the input queries tested on the developed chatbot.

Table 4 : Chatbot testing results

Example Number	User Input	English Translation	Comment about Input	Correctly Answered
1	வீடமைப்பு கடனை பெற்றுக் கொள்ள தேவையான ஆவணங்கள் என்ன ? Vīṭamaippu kaṭaṉai perṟuk koḷḷa tēvaiyāṉa āvaṇaṅkaḷ eṉṉa?	What are the documents required to get a housing loan?	Written Tamil No spelling mistakes	YES
2	வீடமைப்பு கடனை பெற்றுக் கொள்ள தேவையான ஆணங்கள் என்ன ? Vīṭamaippu kaṭaṉai perṟuk koḷḷa tēvaiyāṉa āṇaṅkaḷ eṉṉa?	What are the documents required to get a housing loan?	Written Tamil with spelling mistakes “ஆணங்கள் “	YES
3	எனது Debit அட்டையை நான் ரத்து செய்வதுக்கான செயன்முறை எவை ? Eṉatu Debit aṭṭaiyai nāṉ rattu ceyvatukkāṉa ceyaṉmuṟai evai?	What is the procedure for me to cancel my Debit Card?	Written Tamil with english code mix	YES
4	எனது டெபிட் அட்டையை நான் ரத்து செய்வதுக்கான செயன்முறை எவை ? Eṉatu ṭepiṭ aṭṭaiyai nāṉ rattu ceyvatukkāṉa ceyaṉmuṟai evai?	What is the procedure for me to cancel my Debit Card?	Written Tamil with English code mix written fully in Tamil alphabets	YES
5	நான் என்ர கடன் அட்டையை எப்படி நிப்பாட்ட வேணும் ? Nāṉ eṉra kaṭaṉ aṭṭaiyai eppaṭi irattu ceyyōṇum?	What is the procedure for me to cancel my Debit Card?	Spoken Tamil	YES
6	நான் என்ர debit அட்டையை எப்படி நிப்பாட்ட வேணும் ?	What is the procedure for me to cancel my Debit	Spoken Tamil with English Codemix	No. Returned the wrong

	Nāṉ eṉra debit aṭṭaiyai eppaṭi nippāṭṭa vēṇum?	Card?		answer.
7	நான் என்ர டெபிட் அட்டையை எப்படி நிப்பாட்ட வேணும் ? Nāṉ eṉra ṭepiṭ aṭṭaiyai eppaṭi nippāṭṭa vēṇum?	What is the procedure for me to cancel my Debit Card?	Spoken Tamil with English code mix written fully in Tamil alphabets	No. Returned the wrong answer.

When considering the chatbot's performance results, you can observe that the chatbot accurately responds to written Tamil queries, whether they contain spelling mistakes or involve code-mixed content in written Tamil, irrespective of whether the queries employ English or Tamil alphabets.It was also capable of providing accurate responses in spoken Tamil. However, this chatbot encounters difficulties in cases where the input becomes more complex, involving a combination of spoken Tamil and code mixing.

6.3 Chapter Summary

In this chapter, we initially presented the results of the machine learning models. It was observed that all of our developed models consistently yielded notably high results in comparison to the Thai chatbot. Particularly, the CNN model, employing fastText for vectorization and incorporating class weights to counteract class imbalance, demonstrated the highest level of accuracy. Additionally, SVM exhibited performance comparable to other neural network models.

Subsequently, we provided an account of the chatbot's performance results. The evaluation encompassed a diverse array of queries, assessing the chatbot's accuracy in furnishing accurate responses. The findings illustrated the chatbot's adeptness in precisely addressing written Tamil queries, even when encompassing spelling errors or featuring code-mixed content in written Tamil. Remarkably, this proficiency extended to queries utilizing both English and Tamil alphabets, while also effectively responding in spoken Tamil. However, it is worth noting that the chatbot exhibited certain limitations in other conditions.

7. DISCUSSION

Results of both the machine learning models and the performance of the chatbot are analysed in detail to find varies insights.

7.1 Model Performance

Based on the results it could be seen that the best-performing model was the CNN-based model which used fastText word embedding for word vectorization and class weights to tackle class imbalance. I think the main reason for this is translation invariance which is a key feature of CNN. Translation invariance allows CNN to detect patterns irrespective of their position in the sentence. The local order of words is not that important, this is a very useful feature in a free word-order language like Tamil. So CNNs are well-suited to carry out this task with efficiency.

All our models have yielded very high results compared to the Thai chatbot which used the RNN-LSTM-based model. But since the nuances of the languages and the datasets are different, it is not truly comparable. Though CNN and fastText embedding trained on the dataset with class weight outperformed the rest of the models the performance increase is only small compared to SVM. This is not very much surprising because it is well-known that non-neural network-based models can perform on par or sometimes better than neural network-based models when the dataset size is small.

When fastText is used for vectorizing we see a jump in accuracy, it could be because fastText can handle out-of-vocabulary words well. Our model is also performing well compared to previous works on short text classification in Tamil.

7.2 Chabot Performance

Upon evaluating the chatbot's performance, it becomes evident that the system demonstrates a remarkable level of robustness when it comes to handling written Tamil queries. This robustness extends to queries regardless of whether they contain spelling errors or are composed of code-mixed content in written Tamil. Notably, the chatbot adeptly handles queries that utilize a combination of English and Tamil alphabets, showcasing its ability to

seamlessly navigate the intricacies of code-mixed language. This proficiency across various query types underscores the chatbot's capacity to effectively engage with users in written Tamil, irrespective of linguistic intricacies or potential errors, thus contributing to an enhanced and versatile user experience. Example 3 to 7 used for testing is special because none of them are in the dataset.

That particular intent doesn't have any sample for both spoken Tamil or written Tamil with Code mix. Only written Tamil sample are there for that intent. So it is difficult for chabot to arrive at the correct answer yet the chatbot was able to answer that question correctly. This shows that the chatbot is is robust. Example 6 and 7 is very difficult examples because there are code mix and the spoken Tamil together, in this case the chatbot fail to arrive at the correct response. This is a shortcoming of this chatbot.

7.3 Limitations

There are three types of limitations in this research. The first limitation is caused by the generated dataset that was used for this research and the second one is the limitations of the approach that is used for building the chatbot. The final one is the limitation caused by our testing strategy.

7.3.1 Limitations in Generated Dataset

The dataset we used for this research was gathered using native speakers by using young people. Hence the vocabulary to phrase questions or statements could be limited to the words used by young people. So some of the utterances which are prominent among the older population would have been missed out. Another factor is that all the native speakers are from Colombo, hence the colloquial variation of spoken Tamil such as Jaffna Tamil or Batticaloa Tamil will also not be captured as much as we wanted.

7.3.2 Limitations in the Approach

The main limitation of our approach is, the proposed approach will not be useful to build a chatbot for a very large FAQ corpus like the central bank. Our data generation technique uses the inflectional variations of Tamil to expand the existing corpus, but when the corpus is

large, the amount of time it would take for data generation will be huge and expensive. For example, let's take the central bank FAQ pages which would have more than 1000 questions, so here we need to generate more examples for 1000 classes. Again when modelling, the presence of a large number of classes will affect the choice of algorithms useful for modelling.

In addition to that, if the FAQ corpus has a very rigid structure with complex and compound sentences, our approach would still fail because generating different variations for such questions would be difficult hence the number of examples per class will be less which would generally result in low accuracy.

One of the main limitations of the study is the lack of any previous benchmark for Tamil to compare our model against. Comparing our model with the best-performing models of other languages is not a good way of comparison since the accuracy of other language models is influenced by the nuances of those languages and the resources available for those languages. For example, if we take the Thai chatbot we compared against, in the Thai language, there are no spaces in-between words so their approach would be drastically different from our approach hence accuracy is not truly comparable.

7.3.3 Limitations in the Testing Strategy

The Hold Out method was employed to assess the model's performance in this research. Although we employed stratified sampling, we did not assess per-class accuracies and instead focused solely on the overall accuracy of the test set, utilizing an 80% training and 20% testing dataset split. Looking at accuracy per class would have given more insights into which classes are classified poorly. This would have enabled us to optimise the system more. Another limitation is that the test set was not generated considering the variation such as written Tamil, Spoken Tamil and codemixed Tamil and we didn’t check how the accuracy of classification on the test set varies for different types of Tamil input such as spoken Tamil, code-mixed Tamil and spoken Tamil. This would have given us additional insights into the system.

7.4 Chapter Summary

In this chapter, an analysis of the results from both the machine learning models and the chatbot's performance was conducted to unveil various insights. It was discerned that the superior performance of the CNN model could be attributed to translation invariance, enabling it to identify patterns without dependence on their position within a sentence. This attribute proves particularly advantageous in a free word-order language like Tamil. Moreover, a comprehensive discussion addressed the limitations of comparing the Thai chatbot, emphasizing the dissimilar nuances and datasets of the languages.

Furthermore, the observation was made that SVM's performance is comparable to neural network models, attributed to the relatively small dataset size. Notably, the utilization of fastText for vectorization showcased a noticeable accuracy boost, potentially attributed to its proficiency in handling out-of-vocabulary words. Additionally, our study concluded that our model excels in short text classification within the Tamil context, outperforming previous approaches.

During the discussion on chatbot performance outcomes, it became evident that the chatbot exhibits commendable performance even with examples not originally included in the dataset. However, it encounters challenges primarily in the intricate scenarios where code-mixing is combined with spoken Tamil. Concluding this segment, three distinct limitations of this project were pinpointed: limitations related to the generated dataset, limitations inherent in the approach adopted, and limitations pertaining to the testing strategy employed.

8. CONCLUSION

Through this project, we have demonstrated that it is possible to build a functional context-less chatbot (Level 2 chatbot) for Tamil even with a small dataset by converting the chat problem into a short text classification problem and retrieving the answer from the predefined response based on the intent identified by the short text classifier.

8.1 Future work

8.1.1 Develop a better dataset

One way to improve the dataset for the chatbot is to gather a diverse group of native Tamil speakers from different age groups. It's important to include all possible types of Tamil expressions such as written Tamil, spoken Tamil, and code-mixed Tamil for each intent to avoid any imbalance in the data. This will enhance the chatbot's effectiveness and usefulness.

8.1.2 Create Multiple Responses

Currently there is only one response to pick for each intent. Returning the same answers, again and again, is not a good approach in terms of user experience. Many of the modern chatbots for English have a predefined set of responses for each intent. That can be used for mixed things up in conversation. For example, if a user asks “How to open a bank account”, the response can be “You can visit the bank to open an account”, “The account can be opened by visiting the bank” or “By vising the bank in person you can open an account”. Using the same technique used for generating multiple utterances of the intent. Native speakers can be employed to generate more variations of the answer as well.

8.1.3 Multi-level classification

After a better dataset is developed, another potential research area is to use hierarchical multi-label classification instead of the flat classifier we used. This approach will allow us to classify multiple labels for a text, where each label is part of an underlying hierarchy of categories.

For example in our case all the question regarding saving products can be grouped together and labelled as “savings”, then each intent under the savings category have separate labels like “opening savings”, “savings rate” etc. So if the user enters “How to open savings account” the model would predict the first label as “savings” and the second label as “savings rate”. There is an added benefit of using this approach. For example, if the user asks a follow-up question as “What about the rate?” since we know the previous first label predicted is savings, we can deduce he is asking about the “savings rate”. This approach will enable us to bring context into the conversation. The current system doesn’t have context.

8.1.4 Confidence Score for Classification

The current chatbot that has been developed will be unable to handle queries that are not related to its designated scope. This limitation arises from the current method, which only classifies input queries into existing categories under the assumption that users will exclusively pose questions within the predefined scope. As a result, users may receive incorrect responses instead of the chatbot appropriately stating, "I'm sorry, we can't assist with that. Addressing this issue is crucial to ensure the practical viability of the chatbot, as the assumption that users will strictly adhere to the predefined scope is likely to be violated in real-world scenarios.

A potential strategy to solve this involves incorporating a confidence scoring mechanism for classification and establishing a minimum confidence threshold. When the confidence level surpasses this threshold, the chatbot would present responses; otherwise, it would reply with "I'm sorry, we can't assist with that”. A Confidence Score is a numerical value ranging from 0 to 1, indicating the probability that the class predicted by a machine learning model is accurate. Every prediction is accompanied by a confidence score, where a higher score indicates a greater level of confidence from the machine learning model. Determining the appropriate confidence threshold is a critical step and is dependent on the specific dataset. As a general guideline, if the confidence score of the predicted class exceeds 0.5, it can be considered suitable to permit that response.

In the context of the models we have developed, for non-neural network based models like SVM finding confidence score is pretty straightforward. Scikit-learn library provides methods like “predict_proba” to output the confidence score. When it comes to the confidence score for neural network based models such as Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks it is a bit tricky. One good approach in these cases is to use the probabilities produced by the softmax layer. By analyzing these probabilities derived from the softmax layer, we can determine the model's confidence in its predictions. This information becomes invaluable for setting a suitable confidence threshold to decide whether a particular response should be presented to the user or if the model should indicate its inability to provide assistance for a given input. For out chatbot this method should be good enough.

It's worth noting that a common challenge with many deep neural networks is their tendency to exhibit overconfidence in their predictions, despite performing well in terms of accuracy. This issue becomes particularly apparent when considering the estimated probabilities generated by the softmax layer's output. These predicted probabilities might not accurately reflect the true probabilities of the events they represent.

To address this concern, an effective solution for calibrating the predicted probabilities of neural networks is detailed in the paper titled "On Calibration of Modern Neural Networks"[26]. This calibration process aims to rectify the tendency of neural networks to be excessively confident in their predictions by aligning the predicted probabilities with the actual probabilities of events. Should the need arise for further calibration, techniques like those outlined in the mentioned paper can be explored to enhance the accuracy and reliability of the predicted probabilities from neural networks.

8.2 Chapter Summary

In this final chapter, a comprehensive overview was provided, encapsulating both the project's conclusions and a glimpse into potential future endeavours. Through this project, a significant achievement was showcased—successfully crafting a functional context-less chatbot (Level 2 chatbot) for Tamil, even when dealing with a limited dataset.

This accomplishment was realized by strategically transforming the chat problem into a short text classification problem and subsequently retrieving responses from pre-defined set based on the intent identified by the short text classifier.

Looking ahead, a multitude of avenues for improvement have been identified. The initial area of enhancement entails elevating the dataset's quality by engaging a diverse array of native Tamil speakers spanning various age groups. The creation of multiple responses for each intent, facilitated by native speakers, is the second avenue, aimed at enhancing the overall user experience. The third prospective avenue involves delving into hierarchical classification, a potential avenue to develop a chatbot that effectively comprehends contextual nuances. Finally, leveraging confidence scores to detect and appropriately respond to out-of-scope inputs emerges as a pivotal aspect of further advancement.

REFERENCES

[1] Shawar, B. A., & Atwell, E. S. (2005). Using corpora in machine-learning chatbot systems. International journal of corpus linguistics, 10(4), 489-516.

[2] Ilievski, V., Musat, C., Hossmann, A., & Baeriswyl, M. (2018). Goal-oriented chatbot dialog management bootstrapping with transfer learning. arXiv preprint arXiv:1802.00500.

[3] AlHumoud, S., Al Wazrah, A., & Aldamegh, W. (2018). Arabic Chatbots: A Survey. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 9(8), 535-541.

[4] Song, Y., Yan, R., Li, X., Zhao, D., & Zhang, M. (2016). Two are better than one: An ensemble of retrieval-and generation-based dialog systems. arXiv preprint arXiv:1610.07149.

[5]Gorbachov, V., & Cherednichenko, O. (2017). Improving communication in enterprise solutions: challenges and opportunities. In Computational linguistics and intelligent systems (COLINS 2017). National Technical University.

[6]Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. (2016, March). Building end-to-end dialogue systems using generative hierarchical neural network models. In Thirtieth AAAI Conference on Artificial Intelligence.

[7] Allam, Ali Mohamed Nabil, and Mohamed Hassan Haggag. "The question answering systems: A survey." International Journal of Research and Reviews in Information Sciences (IJRRIS) 2.3 (2012).

[8]Muangkammuen, P., Intiruk, N., & Saikaew, K. R. (2018, November). Automated Thai-FAQ Chatbot using RNN-LSTM. In 2018 22nd International Computer Science and Engineering Conference (ICSEC) (pp. 1-4). IEEE.

[9]Kalaiyarasi, T., Parthasarathi, R., & Geetha, T. V. (2003). Poongkuzhali-an intelligent tamil chatterbot. In SIXTH TAMIL INTERNET 2003 CONFERENCE (Vol. 1). Sn.

[10]Milhorat, P., Schlögl, S., Chollet, G., & Boudy, J. (2013, August). Multi-step Natural Language Understanding. In Proceedings of the SIGDIAL 2013 Conference (pp. 157-159).

[11]Liu, B., Tur, G., Hakkani-Tur, D., Shah, P., & Heck, L. (2017). End-to-end optimization of task-oriented dialogue model with deep reinforcement learning. arXiv preprint arXiv:1711.10712.

[12]Paul, Anirudha, et al. "Focused domain contextual AI chatbot framework for resource poor languages." Journal of Information and Telecommunication 3.2 (2019): 248-269.

[13]Cogcom.ai. (2019). [online] Available at: https://www.cogcom.ai/ [Accessed 4 Jun. 2019].

[14]fairfirst.lk. (2019). Insurance Company in Sri Lanka | Fairfirst Insurance. [online] Available at: https://www.fairfirst.lk/ [Accessed 4 Jun. 2019].

[15]Siregar, R. (2016). Translation quality assessment of "the 8th habit: from effectiveness to greatness by Stephen R. Covey" into Indonesian. International Journal of Language and Literature, 4(1), 228-239.

[16] Wen, T. H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P. H., ... & Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562.

[17] Danda, P., Jwalapuram, P., & Shrivastava, M. (2017, December). End to End Dialog System for Telugu. In Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017) (pp. 265-272).

[18] S, Ramraj; R, Arthi; Murugan, Solai; Julie, M.S. (2020). [IEEE 2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE) - Keonjhar, Odisha, India (2020.7.29-2020.7.31)] 2020 International Conference on Computational Intelligence for Smart Power System and Sustainable Energy (CISPSSE) - Topic categorization of Tamil News Articles using PreTrained Word2Vec Embeddings with Convolutional Neural Network. , 1–4. doi:10.1109/CISPSSE49931.2020.9212248

[19] Your guide to five levels of AI assistants in Enterprise. (2022, January 13). Retrieved April 30, 2023, from https://rasa.com/blog/conversational-ai-your-guide-to-five-levels-of-ai-assistants-in-enterprise/

[20] Bradeško, L., & Mladenić, D. (2012, October). A survey of chatbot systems through a loebner prize competition. In Proceedings of Slovenian language technologies society eighth conference of language technologies (pp. 34-37). Ljubljana, Slovenia: Institut Jožef Stefan.

[21] Noy, N. F., & McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology.

[22] Suta, P., Lan, X., Wu, B., Mongkolnam, P., & Chan, J. H. (2020). An overview of machine learning in chatbots. International Journal of Mechanical Engineering and Robotics Research, 9(4), 502-510.

[23] Wen, T. H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P. H., ... & Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562.

[24] Lowe, R., Pow, N., Serban, I., & Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. arXiv preprint arXiv:1506.08909.

[25] Schatzmann, J., Georgila, K., & Young, S. (2005). Quantitative evaluation of user simulation techniques for spoken dialogue systems. In 6th SIGdial Workshop on DISCOURSE and DIALOGUE.

[26] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On calibration of modern neural networks. In International conference on machine learning (pp. 1321-1330). PMLR.