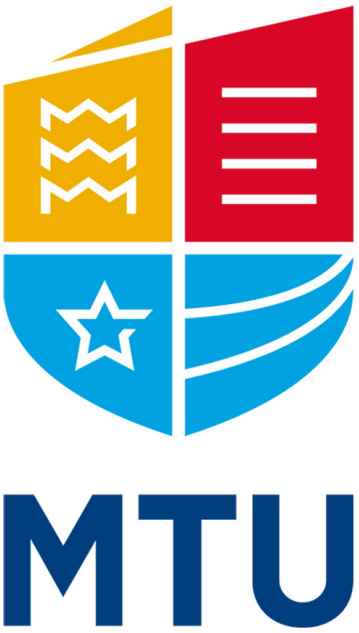


FáilteBot - A multilingual AI for refugees in Ireland

by

Ronan O'Keeffe

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science (Hons) in Software Development

in the
Faculty of Engineering and Science
Department of Computer Science

May 2022

Declaration of Authorship

I, Ronan O'Keeffe , declare that this thesis titled, 'FáilteBot - A multilingual AI for refugees in Ireland' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an undergraduate degree at Munster Technological University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Ronan O'Keeffe

Date: 02/05/2022

Munster Technological University

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Ronan O'Keeffe

The intention of this project is to answer the question; Can conversational AI assist in the integration of refugees in Ireland?

A commitment has been made by the current government in Ireland to support the integration of refugees and replace the system of Direct Provision with a new approach modeled on the successful initiatives to integrate Syrian refugees in Ireland. Currently these people are provided information on living in Ireland via a 100 page PDF in their chosen language. It is an ideal time to examine the delivery of this material and ask is there a better way?

Conversational AI is a growth industry, set to go from a market size of $4.8 billion in 2020, to $13.9 billion in 2025. Conversational AIs combine Natural Language Processing, a form of machine learning, with more common software such as chat-bots, voice assistants, or interactive voice recognition systems. Replacing the need for human support staff or static support pages, the main aim of conversational AIs in the business world is concise and relevant delivery of information. Conversational AIs also increase customer engagement by simulating conversations with quick answers and helpful prompts.

FáilteBot is a multidisciplinary project that aims to combine a conversational AI framework with information provided by the Department of Justice to examine if a technology driven approach can improve the delivery of information to newly settled refugees, and thus improve their integration in a new society.

There are several key objectives of this project:

1. To compare the delivery, and ease of use, of this system to the 'Guide to Living Independently' documents published by the Department of Justice
2. To create an open-source multilingual conversational AI, as a minimum in English and French, and one language using a non-Roman character set.
3. To provide an easy to use, modular system for developers to integrate new languages into the chat-bot.
4. To provide documentation for developers wishing to build upon this application or its principles
5. To evaluate the overall utility of the created system and provide steps for further research or development.

Acknowledgements

I would like to thank my supervisor Dr. Alex Vakaloudis for his assistance in researching and developing this concept. I am grateful to him and Dr. Indika Dhanapala for the professional guidance and education they provided at the Nimbus Centre during my work placement. I would also like to thank Séamus Lankford MSc. for encouraging me to develop this concept when I first proposed it to him in early 2021.

Gratitude is extended to the Department of Justice for use of their materials in creating this project.

Finally, I would like to thank those who supported and encouraged me during the last six months. I am immensely grateful and could not have completed this year without you.

Contents

Cur non.

Chapter 1

Introduction

1.1 Motivation

The motivation for this project began with the publishing of the White Paper on Ending Direct Provision in early 2021. Though the issues with Ireland's system of assisting refugees had always been known to me, reading the report was extremely eye opening. Despite the disappointment felt in the existing system, I was inspired by the commitment to providing a more humane system. As a result of this, I paid greater attention to the issues faced by refugees in Ireland. From there I discovered the 'Guide to Living Independently'.

This guide is a 94 page PDF, in several languages, is provided to refugees in Ireland to assist with their settlement. Though I felt the content of these guides was useful, their delivery seemed outdated in a system that intends to be more forward thinking. From my work placement I had gained experience with conversational agents, which I felt were a useful technology if applied to the correct issue. I initially proposed a chat-bot for refugees to my Work Placement supervisor in April 2021 and over the following months I paid close attention to the area of non-commercial chat-bots and refugees in Ireland.

Ensuring that the chat-bot is multilingual has also been an important aspect of this project since its inception. In my opinion if the 'Guide to Living Independently' is to be adequately replaced, the solution should not be useless to a large proportion of its intended audience.

1.2 Contribution

My intention with this project is to provide a solution to a niche, but important problem. I intend to cover all aspects of this multi-faceted system, from research, to planning, to implementation, and evaluation. Research and elicitation of requirements will occur during the research phase. From there the system will be designed and packaged using an Agile approach.

This project should also demonstrate the utility of cutting edge technologies in unique use cases and how modifications to existing technologies can provide digital solutions to 21st Century problems. It will also provide a set of best practises for developers seeking to follow the FáilteBot concept in designing their own multilingual chatbot.

The contributions developed by this work are as follows:

- Academic research on key topics.
- Identification of technological solutions for existing problems.
- Translating an informational document in machine-readable code for a Rasa-based chat-bot.
- Modifying an existing framework to allow for multilingualism.
- Programming Python scripts for a Rasa-based chat-bot.
- Agile project management skills.
- Maintaining an open-source project through a Git-based source-control system.
- Evaluation of an existing project
- Academic writing

1.3 Structure of this Document

The structure of this document is as follows:

- Chapter 1 is an introduction to this project, it's motivations, and the structure of the document.
- Chapter 2 is a view of the "state of the art" of conversational agent technology.
- Chapter 3 is a more detailed view of the problem this project aims to solve, and the objectives that have been derived therein.
- Chapter 4 provides a high level overview of the system architecture, methodology, and implementation plan.
- Chapter 5 provides a rundown on the work undertaken during the Implementation Phase, examining successes and failures.
- Chapter 6 examines the project based on the objectives established during the Research phase and examines their outcomes. Detailing of system testing is also included.
- Chapter 7 provides a discussion on the project's overall outcome, lessons learned, and future work relating to the FáilteBot concept.

Chapter 2

Background

2.1 The Conversational Agent Market

A conversational agent is a computer system through which a human can have a human-like conversation with the system. Interactions can be made through speech, haptics, text input, or gestures. Voice agents like Amazon's Alexa offer an example of a consumer grade conversational agent. Consumer grade conversational agents are usually used to replace the role of a customer service agent as a source of information, saving on time and resources. Conversational agents are a growth industry, valued at $17.17 billion in 2020, by 2026 the industry is projected to reach over $100 billion[1]. Advances in AI technology and the proliferation of more efficient machine learning processes has triggered a massive growth in the abilities and availability of chatbot technologies. Outside of private companies, government bodies have seen a drastic increase in the need for, and the use of, chatbots. The 2020 Center for Digital Government Surveys, which surveyed public officials in state, city, and county bodies in the United States, pointed to chatbots as an increasingly popular solution to their problems [2].

2.2 Non-Commercial use of Conversational Agents

The closure of government buildings due to the COVID-19 pandemic led to more requests for information from government hot-lines or websites. chatbots provide an easy method of filtering requests while also reducing the human costs of maintaining such a system. This approach can be described as "digital triage", where simple requests for information are handled by digital solutions, with human agents providing answers to more complicated requests. Mass General Brigham, a hospital network of 11 hospitals,

with a patient body of 1.5 million people successfully implemented a digital triage system during the COVID-19 pandemic. In an effort to reduce the number of abandoned calls for information, a chatbot was built for the purpose of dealing with callers who may have milder cases, or the "worried well".[3]

In King County, Washington, a chatbot helped identify coronavirus-like symptoms in users and directed them to a nurse if they fit the criteria. The county estimated that the chatbot reduced the time spent by nurses on these requests by around 35%. The Texas Workforce Commission has also utilised chatbots to deal with frequently asked questions. This has allowed their workers to focus on more pressing or complex issues. A COVID-19 chatbot for the state of Connecticut logged approximately 40,000 interactions in a four month time frame, on a 24/7 basis. The state estimated that this work would have required four full time employees to complete this task normally. Of particular relevance to this paper is the case of San Joaquin County's chatbot. The county built a multilingual chatbot to answer user queries on a variety of topics. The chatbot, entitled 'AskSJC', can be accessed at sjgov.org.[2]

Finally, in a 2021 study on a COVID-19 chatbot Mabrouk [4] et al. found that upon the outset of the COVID-19 pandemic, government institutions in Tunisia were faced with an overwhelming request for information that could not be delivered by conventional means such as by phone hot-lines. This is relevant to this particular use case as refugee intakes are not a steady stream. Instead, large numbers of people are taken in on short notice, most recently in Sprint of 2022 when the Irish government committed to accepting 200,000 Ukrainian after the invasion of Ukraine. Any system to address this use case should also be modular enough to allow for the quick and easy addition of other languages. The technical overhead should be low as well, reducing the costs of maintaining such a project.

2.3 Classification of Conversational Agents

For the purpose of maintaining a simple implementation of a multilingual conversational agent, we will focus on a text based agent, sometimes referred to as "chatbots". Chatbots are usually embedded into web pages or integrated with chat services such as WhatsApp, Facebook Messenger, or Discord.

There are many different approaches to classify conversational agents based on their context of use, technical capabilities, modality etc. A prevailing standard for the classification of conversational agents does not yet exist. In the context of this work we can consider there to be two main categories of chatbot. The first category can be described

as 'finite' chatbots. These bots have a limited amount of responses they can provide the user. Interactions with these bots are usually done through buttons. Basing interactions around buttons guarantees a higher level of understanding of the intent of the user. Bots like this are best used for basic business interactions and frequently asked questions. When a user has reached the limit of the amount of information a finite chatbot can provide, or asks a question outside the scope of the bot, they are usually pointed towards further resources.

The other category of chatbots are best described as 'infinite' bots. These bots are intended to provide an endless amount of conversation for the user. Infinite chatbots are considerably more difficult to implement, requiring a lot of training data, and cannot guarantee correct responses. This is a concerning point for this paper as reliable and accurate information is required to build trust between the system and prospective users. Because of this, the intention of this project will be to build a finite chatbot, based mostly on button based interactions.

2.4 Conversational Agents - Concepts and Architecture

The architecture of a conversational agent, in broad terms, follows a similar principle regardless of type. A user inputs a message, the message is classified, compared against a context and training data, a response is selected, and returned to the user. Through this a conversation is generated, between a computer and a user. To understand the deeper architecture of conversational agents some terms must be introduced.

Intents - An intent is the goal, or intention, behind an input from a user. For example, if a user says "Hello" to their voice assistant, the conversational agent classifies the intent of this input to be a greeting. If the user said "Good morning", "Hey there", or "Hi", these would also be considered greeting intents.

Entities - When a user inputs a message the conversational agent will search their message for entities. Entities are important pieces of information contained within a message that may modify the context of an interaction. They are mainly used to catch data input from the user. For example, a chatbot could ask the user for their email address, to which the user replies "My email address is example@mtu.ie". The useful piece of information here is just the email address, a properly trained entity extractor will recognise that "My email address is..." is unimportant information, and screen that out. Entities are also sometimes referred to as "slots".

Much like the classification of conversational agents, the classification of entities is platform dependent. However, generally they follow a similar vernacular.

Contextual/Simple Entities - These are the most basic form of entity. A simple entity is a singular piece of data. For example, an EirCode, a phone number, or an email, are all simple entities.

Composite Entities - Composite entities are entities composed of multiple pieces of data that alter the context together. For example, a user asking a book recommendation bot for a "cheap horror book not written by Stephen King" has provided a composite entity for the program.

Stories and Context - Stories are conversational scenarios between a user and the conversational agent. Conversational agents are provided with stories by the developer, from which they can train for possible interactions. For example, a conversational agent may be given a story to tell them when a user says "Hi" they should reply with "Hi, how may I help you?". The position of the conversational agent within this story is referred to as the "context". As the bot traverses the story the context changes, refining the possible responses it can select.

```
- story: Recommend a book          # Name of the story
  steps:
  - intent: greet                  # User inputs a greeting - e.g: "Hello"
  - action: utter_ask_howcanhelp   # Bot replies asking how it can help - e.g: "Hi there, what kind of book are you looking for?"
  - intent: inform                 # User inputs the type of book, an entity is picked up - e.g: "I'm looking for a horror book"
    entities:
    - type: "horror"               # The entity extractor has recognised that the user is looking for a horror recommendation
  - action: utter_on_it            # Bot utters a placeholder message - e.g:"I loved horror! Give me one second to have a think"
  - action: utter_horror           # Bot replies with a recommendation - e.g: "How about 'The Shining' by Stephen King?"
```

FIGURE 2.1: An example of a story for a Rasa 2.0 based chatbot

The above figure is an example of a story for a chatbot based on Rasa 2.0. In this example the bot recommends books based on a query from the user. In this story, the user has asked for a horror recommendation. The particular entity extracted at this point is "horror". However, the chatbot could be configured to search for other entities in the same input, such as price and author. A user who inputs "I'm looking for a horror novel not written by Stephen King" would not find the above example to be an acceptable response.

Natural Language Processing and Understanding - Selecting an appropriate response based on the above concepts is a process referred to as "Natural Language Processing", or NLP. NLP is an extremely common feature of modern conversational agents based on Artificial Intelligence. The extraction of intents and entities from inputs is also referred to as "Natural Language Understanding", or NLU. NLU utilises machine learning to properly extract the meaning of user inputs. A properly configured context will allow for a firmly defined conversation to play out between the user and the conversational agent. It prevents the bot from just focusing on the immediate input from the user, and maintains the overall context of the wider conversation being had. The

accuracy of NLU in a conversational agent is decided by the quality of the training data provided to the model. Poor quality training data will result in poor quality responses. However, no conversational agent is 100% accurate and a useful fallback response should always be available to a conversational agent when it is unable to process the input of a user.

2.5 Frameworks for Text-Based Conversational Agents

There are many available chatbot frameworks. Companies like Microsoft, Amazon, Google, and Facebook all offer proprietary platforms from which to run a chatbot. An advantage of these technologies is that many of them can be easily integrated with devices like Amazon's Alexa, Siri, or Google Assistant. Smaller companies such as Rasa and Wit.AI offer more customisable, and open source, chatbot solutions. Before selecting a platform several factors must be considered for this particular use case. The programming language used will affect the cost and availability of additional resources. Preferably the framework will utilise a well known language with a wide range of packages. Packages like TextBlob for Python offer features like sentiment analysis, which can be useful for chatbot technology. The technical skills required should also be considered. Multilingual support, and custom actions are critically important features for this use case. Finally, the availability of an Open Source version, or a Free plan, are important considerations, to keep costs to a minimum and to offer further extension of the ideas developed by this project.

Framework	Language	Technical Skills Required	NLP	Multilingual Support	Customisation	Open Source	Cost
Rasa	Python	Yes	Yes	Yes	Yes	Yes	Free
IBM Watson	Java, C++	Yes	Yes	Yes	Yes	No	Free up to 1,000 users
Dialogflow	NodeJS	No	Yes	Yes	Yes	No	$0.002 per request
Wit.AI	Ruby	Yes	Yes	No	Yes	Yes	Free
Lex	Java	Yes	Yes	Yes	Yes	No	$0.00075 per request
Chatfuel	n/a	No	No	Yes	No	No	Free for up to 50 users
Botpress	JavaScript	Yes	Yes	Yes	Yes	Yes	Free
BotKit	NodeJS	Yes	No	No	Yes	Yes	Free

It is clear that many of the technologies could be used for this particular use case. However, it is important that a framework that offers as many of the desired features as possible is chosen. From the above table Rasa, Lex, and Botpress are all technologies that offer the closest features to our use case. Each will be examined in further detail to decide the most suitable framework for this project.

2.5.1 Rasa Open Source

Concepts and Architecture

Rasa is an open source machine learning based conversational agent framework. Rasa allows for text or voice based conversations between users and the agent. The architecture of Rasa Open Source centres around the conversational agent. Intent classification, extraction of entities, and generation of responses is handled in the same manner as Fig 2.2. The Rasa team refers to the NLU process as the "NLU Pipeline". Custom functionality, referred to as "actions", are written in Python and stored in the "Action Server". The Action Server must be running for the chatbot to access this custom functionality. A Rasa application can be deployed through a cloud service such as AWS, or on an on-premise machine.

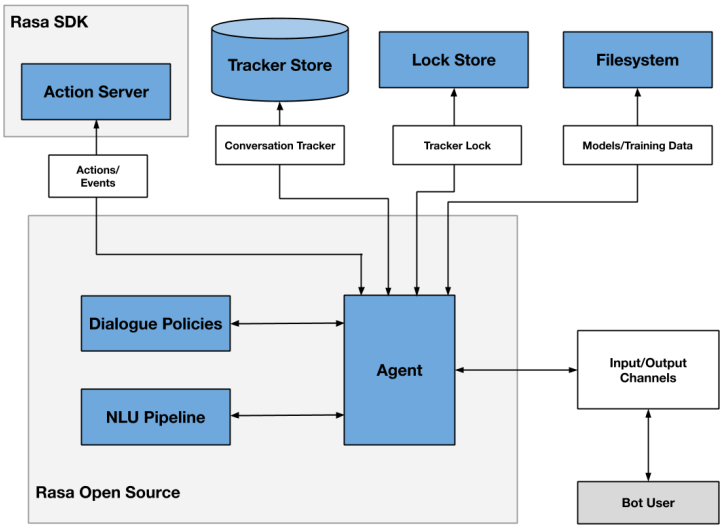


FIGURE 2.2: Architecture of a Rasa chatbot [5]

Technical Skills Required

A minimal amount of technical skills are required to create and run a basic Rasa bot. Responses and stories are configured through Markdown or YAML files, which use plain English utterances attached to a variable name. Knowledge of Python and basic command line usage is required to configure Rasa to a greater degree.

NLP Support

Rasa's NLP functionality is based on machine-learning Python libraries like Tensorflow and spaCy. The open source and modular nature of Rasa allows for developers to configure, train, and implement their own NLU models. Existing or bespoke modules for features such as sentiment analysis or spelling correction can also be implemented.

Multilingualism

Rasa offers a high enough level of customisation to implement multilingualism. However, training data and translations must be provided by the user, either by connecting to a translation API such as Google Translate, or translating and transcribing a script with human translators.

Customisation

Rasa is one of the more customisable frameworks examined. All aspects of the NLU pipeline can be configured by the developer. The Python Actions server also allows for the developer to heavily configure the responses and entity extraction.

Cost

Rasa Open Source is completely free, but an enterprise version is available, with additional features. However, Open Source has the minimum features required for this project.

2.5.2 Amazon Lex

Concepts and Architecture

Lex is an Amazon web service that allows for users to create conversational agents, for voice or text conversations. Though Lex can be used as a voice assistant it is not to be confused with Amazon's smart speaker device Alexa. Each was created independently by different teams at Amazon.

Lex is a cloud based platform that can easily integrate with other Amazon services such as translation with Translate, text-to-speech with Polly, or intent fulfilment with Amazon Lambda (Amazon Lex Documentation, 2021). Below is an example of a Lex application integrated with associated Amazon microservices.

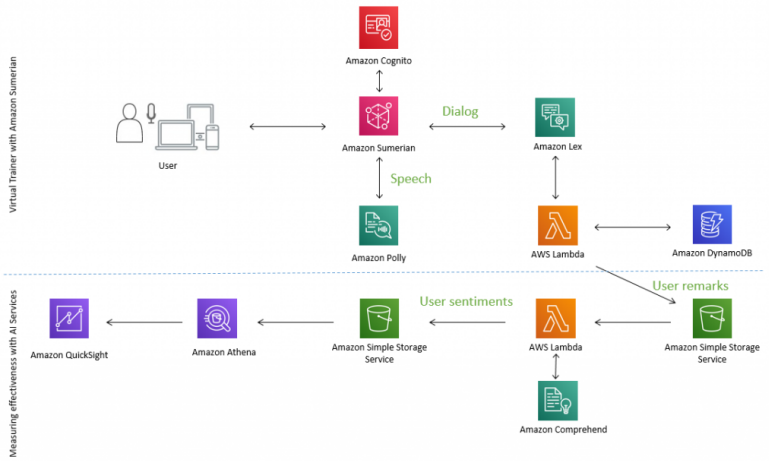


FIGURE 2.3: Example architecture of a Lex-connected application [6]

Technical Skills Required

Unlike Rasa, the technical skills for Lex are far more minimal. A Lex chatbot can be created entirely through the Lex console, a graphical user interface that allows users to build stories and responses without needing to program. To deploy the bot knowledge of cloud services is required, however Lex is built to be easily deployed through Amazon's cloud services. Further programming with Lex requires some knowledge of Java.

NLP Support

Lex combines Natural Language Understanding functionality with Automatic Speech Recognition to create what Amazon refers to as "Speech Language Understanding". The principle of intent seeking and entity extraction is similar to other conversational agents but the exact mechanisms are closed source. The models used by Lex to run this SLU system are also encapsulated, and cannot be configured like with a Rasa one.

Multilingualism

Lex offers multilingualism through Amazon's Translate service. However, the dialogue has to be translated from a parent document. This means that the translated dialogue may not be as accurate as desired. This issue is of particular concern for this use case as the bot needs to provide information as accurately as the translated "Guide to Living Independently".

Customisation

In comparison to Rasa, Lex is far less customisable. As a closed source project, features like the NLU models cannot be adjusted by the developer. In addition to this, Lex is best integrated with other Amazon services, which may not be to the desired level of functionality.

Cost

Of the three frameworks being examined, Lex is the only service that has an associated cost. Lex is priced on a per request basis, with each text request costing $0.00075. The monthly cost for a bot with 50 daily users, averaging 12 messages each, would be $13.95. While this is a relatively low cost, use of additional services such as Amazon's Simple Storage Service (S3) will add to the cost of the bot. There is also no option to host a Lex application outside of the Amazon ecosystem.

2.5.3 Botpress

Concepts and Architecture

Botpress is an open source conversational agent framework. Like Rasa it is in use in industry. Botpress is described as a "cloud-agnostic" system, like Rasa, it can be deployed on an on premises machine or a cloud service. A key feature of Botpress is extensive multilingualism, with approximately 150 languages available through fastText models.

Technical Skills Required

Botpress can be described as a low code platform, requiring few technical skills, and offering a graphical user interface for building conversations. Additional functionality can be implemented through NodeJS functions. A Botpress chatbot can be deployed as an embedded web asset or through messaging services like Facebook Messenger or Slack.

NLP Support

Botpress offers a built in Natural Language Processing system. Interestingly Botpress offers features such as spellchecking out of the box, rather than as optional modules. In comparison to Rasa the NLU models are far less configurable but more is offered out of the box.

Multilingualism

Botpress excels in terms of multilingualism, with native support for over 150 languages available through the use of open-source fastText models. However, this does not cover the full spectrum of human languages. Frequently it is ethnic minorities with unique languages that are persecuted in conflicts, such as the Rohingya, who's own language does not have an available FastText model. If a Botpress approach is chosen, this will mean we will have to generate our own FastText models, distracting from the purpose of this project and increasing the technical skills required to update this solution should another language need to be added.

Customisation

Botpress sits between Lex and Rasa in terms of customisation. Though a Botpress application is more customisable than a Lex one, it will fall short of the control offered by Rasa. As previously mentioned however, Botpress has far more features built into the base version.

Cost

Like Rasa, Botpress is completely free but offers an enterprise edition for extended functionality. However, this functionality is not required for this particular use case.

2.5.4 Final Evaluation

Any of these frameworks would be a suitable selection for this use case, each offering a lot of functionality and unique features. However, for the purpose of a text based chatbot, Rasa is the best option. Firstly, the technology is open-source, reducing costs and indicating a more modifiable architecture. Rasa is also considerably more flexible than the next closest competitor, Botpress. These factors make it the most suitable option for this particular use case.

Chapter 3

FáilteBot - A multilingual AI for refugees in Ireland

3.1 Problem Definition

According to the United Nations High Commissioner for Refugees, there are 82.4 million displaced people worldwide. 34.4 million of these people are externally displaced, meaning they have been moved from their country of origin to another. Refugees are defined as "People fleeing conflict or persecution. They are defined and protected in international law, and must not be expelled or returned to situations where their life and freedom are at risk." This same report indicates that, due to climate change, these figures are set to rise. Refugees come from across the globe, covering a vast spectrum of the human race and its cultures. In 2020, there were 11.2 million newly displaced people worldwide. These refugees stemmed from the Sahel region, Mozambique, Ethiopia, Yemen, Somalia, Syria, Afghanistan, Armenia, and Azerbaijan [7].

Arising from this displacement of refugees, be they external or internal, are problems of integration. Integration, simply put, is the intermixing of previously segregated populations. Poor integration of refugees can have negative effects on the refugee population, as well as that of the host country. Refugees without support will suffer economic disadvantages and the associated negative outcomes that stem from this. Problems of housing, education, and health are common. A knock on effect of this is that the host country must then undo the effects of these problems.(Challenges and successes in refugee integration, 2021). The UNHCR describes integration of refugees as a "two-way process". Though a refugee must be prepared to adapt to their host society, without foregoing their own identity, a similar effort must be made on behalf of the host community. Within an Irish context, the issue of language barriers has been a consistent one for newly settled

refugees[8]. In 2007 a conference of GPs in Galway highlighted cases of doctors having to communicate with "gestures, miming, dictionaries, and the help of [English-speaking] children" to provide medical care to refugee patients[9].

In Ireland refugees are processed via the system of Direct Provision. Originally established in 1999 as an emergency measure, the system has been the source of widespread criticism, across all parties. In 2021 the Irish government signalled its intention to end the system completely by 2024. The system will, instead, be based on the successful program used to integrate Syrian refugees in Ireland. In the White Paper on Ending Direct Provision Taoiseach Michael Martin said this model will be:

"Centred on a human rights and equality based approach, this new model will support applicants for International Protection from day one. It will allow us to ensure their needs are met and that they can integrate with independence into the community. It recognises the diverse and differing needs of applicants, depending on their life situations"[10].

With this new forward-thinking approach it seems to be a good time to ask, can technology be used to improve the integration of refugees in Ireland? Though issues like adequate shelter or economic empowerment are outside of the scope of a software project, there are aspects of the system that can be improved by a technological approach. Currently one of the mechanisms for assisting refugees settle into Irish society is the Department of Justice's "Guides to living independently". This is a 94 page PDF translated into 6 different languages and distributed to refugees or people granted protection status in Ireland. This guide provides basic information on topics such as accommodation, healthcare, employment, and education in Ireland. The guide also provides a directory of local government offices and links to additional resources that may be useful to refugees [11].

Though this guide is well written, there is room for improvement on this model. Technology offers many solutions to this. There are many possible applications of technology to improve a system like this. For the purpose of delivering informational content there are two broad options. These can be described as static solutions and interactive ones. Static resources require no interaction from the user and will only host information. For example, a website that transcribes all the information contained in the Guide to Living Independently would be described as static. Interactive resources do require more interaction from the user. In return they will be able to access the information they want in a timely manner. A telephone hot-line that refugees could call to answer their questions would be an interactive resource. However, a hot-line comes with a large overhead cost from maintaining a call centre or out-sourcing the task to a private company. It may also be difficult to find native speakers of the refugee's languages to man these phone

lines. Another solution might be a social media account, with many state bodies like An Post and Transport for Ireland maintaining similar accounts. However, this would be on a public forum and any private interactions between users and the account would be stored by a private social media company. There are ethical concerns associated with this and it may reduce the trust users have in the system. This would, ultimately, be counteractive to the intention of this solution.

An alternative to these solutions is a conversational agent. This offers an interactive, and private, medium for a refugee to learn from. It would also offer a way of delivering content in the native language of the user, something a hot-line or social media account would struggle to offer. Conversational agents have also been demonstrated to be better than static resources. A 2021 study by Altay, S et al. indicated that chat-bots were more effective at imparting vaccine information to participants versus a static page of text [12].

3.2 Objectives

There are several key aims for this project based on my analysis of the problem.

- **The system should be multilingual**
  Based on my research, the language barrier is one of the greatest blocks to the integration of refugees within Irish society. An excellent English language chat-bot is of no use to Pushtu speaking Afghans. Ideally, this system should also use a vernacular that the refugees are comfortable with. However, this kind of work is beyond the scope of this particular project.

- **The system should be able to easily integrate additional languages as quickly as possible**
  It cannot be assumed that a system like this will be maintained by a developer solely dedicated to the system. Reducing the required workload will reduce the costs of maintenance. Additionally, should another language suddenly be required of the system, integration of the new language should not be a time consuming task. Ideally the translation should be the only task that takes any appreciable amount of time.

- **The system should have the functionality to take feedback from users**
  It is important that the system should not be seen as a static resource. The information will change, it is important that any inaccuracies should be easy to report for users. In addition to this, users may spot inconsistencies in translations or transcriptions. It is important that they are allowed the ability to report these to maintain the quality of the application.

- **Amendments to existing content must also be as easy to implement as possible**
  It should be as easy as possible for the administrators of the system to amend the content within the system. This allows for the content to be continuously improved. Without an easy method of implementing changes, the best feedback system in the world will be useless.

- **The system should be able to identify when a question is out of its scope**
  It is important to maintain the scope of the system and acknowledge that this chat-bot cannot conceivably answer every question put to it. A so-called "infinite" chat-bot is a level of work beyond the level of this particular project. With this in mind, the system should be able to identify questions out of its scope and point the user to relevant resources.

- **The system should be accessible across as many platforms as possible**
  In order to have the highest level of coverage the system should be available on any kind of smartphone device and desktop. This includes devices on older versions of their operating system.
- **The system should be trustworthy**
  To maintain a high level of use in such a system, trust between the user and the application is paramount. It should be clear that the application will not collect information from the users and that the questions asked by the users are not accessible by outside parties.

It is important to note that the actual content of this system, the information provided and its vernacular, is outside the scope of this project. This information will be drawn from the documents already made available by the Department of Justice. This ensures that there is an easy comparison between the current approach and this proposed one. However, given more resources, this would be an extremely important aspect to examine.

3.3 Domain Objectives

The following are the domain-level objectives for this project, based on the objectives outlined above. Domain-level objectives can are the objectives considered to be related to the project, but unrelated to its technological implementation.

1. To improve upon the current model of settlement for refugees in Ireland, by providing previously available information in a more interactive digital format.
2. To provide content based on key topics of concern to refugees in Ireland.
3. To compare the delivery, and ease of use, of this system to the 'Guide to Living Independently' documents published by the Department of Justice.
4. To create a conversational agent that can provide a relevant answer, or helpful fallback response, to user inputs.
5. To allow improvements to the application by receiving and evaluating user feedback.
6. To provide a conversational agent that provides information in multiple languages
7. To create a system that is flexible and adaptable to content change

3.4 Technical Objectives

The following are the technical objectives for this project, based on the objectives outlined above. Technical objectives can be considered to be the objectives relating to the technological implementation of the project.

8. To develop and deploy an open-source multilingual conversational agent
9. To build this agent based on prior research and work done
10. To provide finite conversations in English and French, with room for integration of a sample language using a non-Roman character set (e.g: Ge'ez, Cyrillic, Urdu).
11. To demonstrate easy to use and modular architecture for developers.
12. To demonstrate a linguistically modular system.
13. To provide documentation for developers wishing to build upon this concept or its principles
14. To establish a system that is transparent, trustworthy, and does not store personal information.

Chapter 4

Implementation Approach

4.1 Architecture

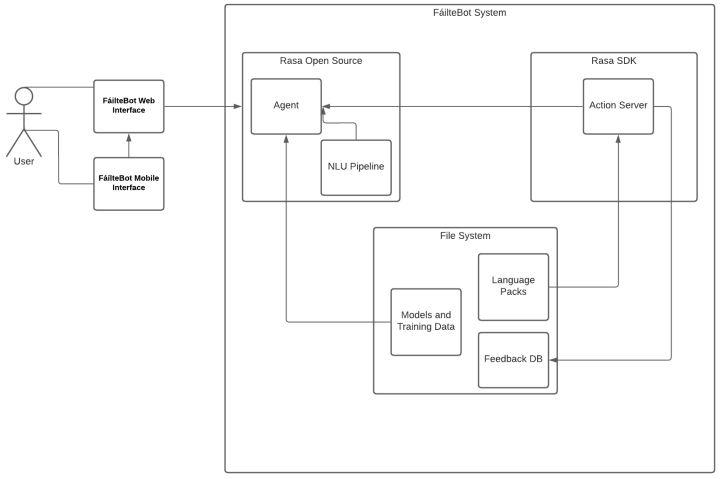


Figure 4.1: A UML diagram of the proposed FáilteBot system

It is possible for the FáilteBot system to be deployed entirely on a local machine or a cloud server assuming it meets the requirements of a Rasa system. The machine will require a modern Linux or Windows distribution, 2-6 vCPUs, 4-8GB of RAM, and approximately 100GB of disk space.

The system will be built with a modified version of the Rasa Open Source chat-bot framework. Unlike a typical Rasa project, hard-coded system utterances will not be used during story scripts. Instead a custom utterance action will be run by the Action Server. From here Python actions will fetch the user language and desired utterance key. These will be matched to an appropriate language pack to find the correct dialogue, which will then be returned to the user. Each language pack will consist of Comma

Separated Values (CSV) file with two columns; utterance key and a content value in that pack's particular language.

For example, the stories file indicates that FáilteBot should greet the user at a certain point by running the action "say_hello". The Agent will ask the Action Server to run this action. The Action Server will first check the user's language slot. The Action Server then searches the language pack of that particular language for the utterance key "say_hello", it finds the row and outputs the content value to the user.

Users will interact with the FáilteBot agent in two ways, through an Android or iOS mobile application or a web interface. For the purposes of simplicity, and to maintain an achievable scope for this project, these applications will be simple web-view apps. These apps will display the web interface in a mobile format for the user.

4.2 Use-Case Description

Below the use-cases for the FáilteBot System, presented as user stories. Each user story will be used as a work package during the Implementation Phase. Some user stories will include a sequence diagram to better communicate their implementation. Each story will also include the objectives that they fulfill, they can be referenced in the Objectives section of page 14 of the report. It is important to note that some use cases encapsulate multiple stories. In this section these will be presented as one story but will be separated for implementation.

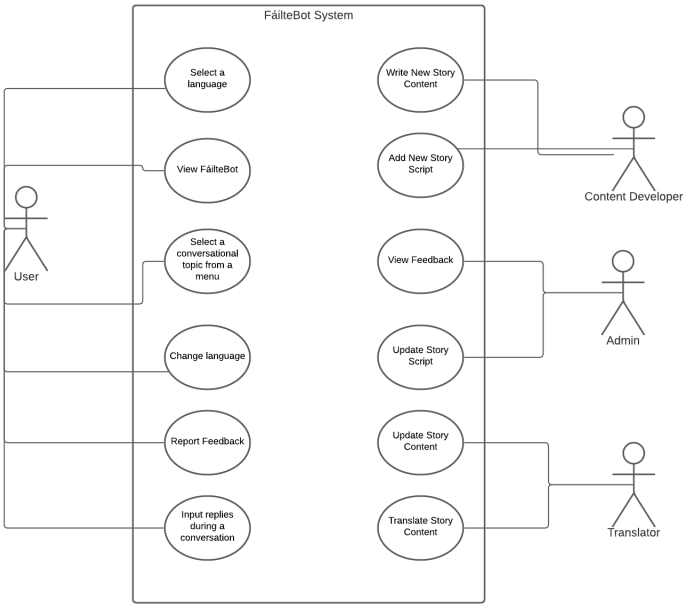


FIGURE 4.2: Use-Case diagram of selected use-cases

There are several roles envisaged in these use cases. The first is that of a User, the person who interacts with FáilteBot. They want to learn about topics of importance relating to living in Ireland. The next role is the Content Developer. This role would ideally be occupied by a software developer comfortable with scripting in Markdown. Their role is to develop new stories for the chatbot. The content for these stories is then translated by the Translator. Finally, the Admin maintains the project. It is envisaged that the Admin is someone of low technical ability but is able to handle the FáilteBot system's most basic mistakes.

1. **View FáilteBot -** *(Obj. 7) As a user I want to interact with FáilteBot via a graphical user interface*
   Acceptance Criteria - The user is using a computer running the FáilteBot application page or has accessed a website where it is hosted. In both cases the user can see a chat interface and a text from the bot that displays a menu with buttons to access various topics

2. **Select a language -** *(Obj. 3) As a user I want to select a language to converse with FáilteBot through*
   Acceptance Criteria - The user is using FáilteBot for the first time. FáilteBot greets them and asks them which language they would like to use. They are given a list of buttons of available languages. The user selects one and FáilteBot begins using the language.

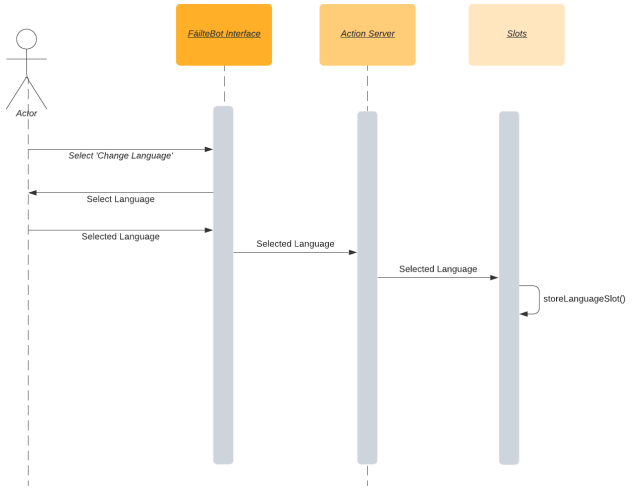


FIGURE 4.3: Use-Case diagram of selected use-cases

3. **Change language -** *(Obj. 3) As a user I want to be able to change the language that I converse with FáilteBot through*
   Acceptance Criteria - The user is using FáilteBot and has decided to change to a different language. The user returns to the main menu and selects the "Reset language" option. FáilteBot offers the selection of languages again. The user selects one and FáilteBot begins using the language.

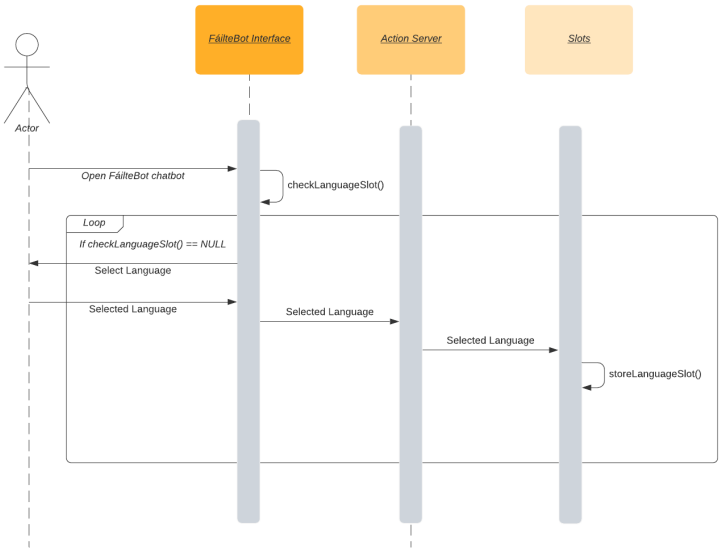


FIGURE 4.4: Use-Case diagram of selected use-cases

4. **Select a conversational topic from a menu -** *(Obj 1, 2, 8) As a user I want to be able to select a conversational topic so that I can find out more information about it.*
   Acceptance Criteria - The user is using the FáilteBot system and has passed any on-boarding process. They are greeted by a menu of possible conversation topics. The user chooses one by pressing a button or typing an input that corresponds to the desired option they are interested in. After this a conversation is triggered.

5. **Input replies during a conversation -** *(Obj. 4, 6, 8) As a user, when I'm conversing with FáilteBot I want to be able to input replies via text or button presses to continue or branch the conversation*
   Acceptance Criteria - The user is in a conversation with FáilteBot, they have reached a point where FáilteBot expects a reply. The user is given options via buttons. They can select one of the buttons or input text that corresponds to

their desired option. FáilteBot then replies with a relevant answer or fallback response.

6. **Report Feedback -** *(Obj. 5) As a user, if I have any feedback for the FáilteBot system I should be able to report this through the FáilteBot system.*
   Acceptance Criteria - The user is in the main menu of FáilteBot. They would like to report that one of the conversations does not work properly. They select the "Submit feedback" option. FáilteBot directs them to a website where they can submit bug reports or feedback in their language of choice.

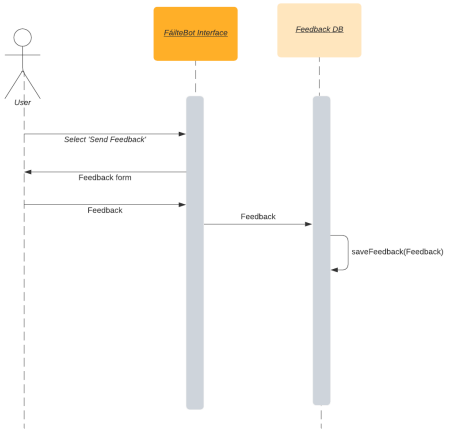


FIGURE 4.5: Sequence diagram for report feedback use case

7. **Write New Story Content -** *(Obj. 2, 9) As a content developer I want to be able to write a new story for FáilteBot to use in a flexible and easy manner as to keep the content up to date.*
   Acceptance Criteria - The content developer has decided to integrate a new conversation into the FáilteBot system. The content developer draws out a conversation, breaking it into a conversational format. They then give this script to the administrator. They should be able to do this without any coding.

8. **Add New Story Script -** *(Obj. 2, 4, 9) As an administrator I want to be able to add a new script to the system, based off the story developed by the Content Developer.*
   Acceptance Criteria - The administrator has been asked to include a new conversation in FáilteBot. The admin has been given a script by the content developer. The administrator then transcribes this script into a format that FáilteBot can

read. They include this in FáilteBot's stories directory, train a new model, and update the system. Users can now access this new conversation.

9. **Update Story Script -** *(Obj. 2, 9) As an administrator I want to be able to update existing story's scripts so as to keep the content up to date*
   Acceptance Criteria - The administrator has been asked to amend an existing conversation's script. The administrator should be able to make these changes to the existing script.

10. **View Feedback -** *(Obj. 5) As an administrator I want to be able to see feedback submitted by users so as to improve the system*
    Acceptance Criteria - The administrator is emailed when feedback has been submitted by a user. The administrator can also view all the feedback submitted so far through an Excel or Google Sheets document.

11. **Translate Story Content -** *(Obj. 4, 6, 8, 10) As a translator I want to have clear instructions on what content is required and how to edit it so as to to provide translation for a new language*
    Acceptance Criteria - The translator should be able to take the content for one language and translate it to another language without having to edit code.

12. **Update Story Content -** *(Obj. 4, 6, 8, 10) As a translator I want to be able to easily amend existing content so as to improve or correct the conversation*
    Acceptance Criteria - A translator sees a mistranslation or thinks of a better translation for a certain piece of content. They should be able to amend this content without having to edit code.

4.3 Risk Assessment

A risk assessment on the project was performed using the follow matrices based on a guide provided by the University of Manchester[13]. Risk scores are calculated based on the product of the risk's Likelihood and Impact. The higher the risk score the more pressing the need to resolve or mitigate the risk.

Likelihood Description	Score
Rare, 0-5%	1
Possible, 6-20%	2
Likely, 21-50%	3
Very Likely, 51-79%	4
Almost Certain, +80%	5

Impact Description	Score
No significant Impact	1
Minor Impact	2
Adverse Impact	3
Major Impact	4
Complete Shutdown	5

Risk Score	Risk Assessment
0-5	Very Low Severity
6-10	Low Severity
11-15	Moderate Severity
16-20	High Severity
21-25	Very High Severity

The following are the risks identified with the project, the likelihood of the risk, the impact of the risk if it occurs, the overall risk rating, and the mitigation actions that will be taken to reduce their severity.

Risk	Likelihood	Impact	Risk	Actions Taken
Code base is corrupted/lost	2	4	Low	An internet based version control platform will be used (e.g: GitHub or GitLab).
Available content is removed	2	4	Low	The available content will be backed up in several locations.
Incorrect transcriptions	4	2	Low	Robust testing of stories will be conducted continuously, to maintain the integrity of the stories.

No available content for FáilteBot	3	4	Medium	Base content in all target languages has been identified.
Underestimate story points for work packages	4	4	Medium	Work package costs will be updated throughout the Implementation phase. Story points assumptions will assume a worst case scenario.
Translation required for transcription of FáilteBot content	5	3	Medium	Google Translate or a human translation service will be used.
Rasa provides incorrect content	4	3	Medium	Leave conversation option and high quality training data for NLU.
Feature creep causes project to be delayed	4	5	Medium	Work Plan has been broken into sprint format.

4.4 Methodology

The approach used to research this project is best described as topical. Topics and concepts were identified and researched individually. To obtain this holistic view of the conversational agent field wide range of papers were reviewed. These were obtained through keyword searches with search engines such as Google Scholar. Dr Alex Vakaloudis, the supervisor of this project and industry expert, was also consulted for literature recommendations.

A range of research was carried out on the topic of refugees and their integration as well. The 'Guide to Living Independently' was quickly identified as a significant gap that conversational agent technology would be able to resolve.

A wide range of computer science skills are required to bring a project of this nature to life. The intention of this project is to provide a complete solution, not just a conversational agent. A significant amount of time will be devoted to aspects of the system outside of the conversational agent and the same diligence in research on the agent will be applied to the other components.

Conducting Academic Research - The first skill is that of academic research. A significant amount of time was dedicated to reading about the topic of conversational agents. Dr Vakaloudis was also consistently consulted for his insight. Finally, a significant amount of time was spent researching refugees, a topic outside the realm of computer science.

System Architecture - System Architecture is an important high level skill for the development of a complete system such as FáilteBot. Time was dedicated, during the Research Phase, to developing this skill. A period of experimentation, constant testing and incremental development of the entire system will also assist in developing this skill during the Implementation Phase.

Conversational Agent Technologies - To implement this solution several areas of practical computer science knowledge must be advanced. The most important is conversational agent tools. A period of experimentation with the chosen conversational agent tool will be required before any full implementation begins. An important aspect of this system will be modifying it to provide the multilingual elements of the chat-bot. Once this functionality is available it will become a question of translating the documents into a conversational format, and then transferring these into the conversational agent framework. It will not be realistic to completely transcribe the 'Guide to Living Independently' document in the short time frame available. Fully functioning conversations covering several different topics in more than two languages should be achievable.

4.5 Implementation Plan Schedule

According to the module description of the Project - Implementation Phase (INTR8015) there are 13.5 hours per week available to the student. Including the Easter break, but excluding the two weeks of exams, there are approximately 15 weeks of work available. This leaves just over 200 hours in Semester 2 to develop this system. To mitigate time loss, a worst case scenario will be assumed for each task. In addition to this, the project workload total workload will not be set to 200 hours. Once again, this is to mitigate for extenuating circumstances and ensure the full delivery of a functioning project.

While testing will be continuous, a sprint will be dedicated solely to evaluation and refinement of the system. This is to ensure the content is as accurate as possible and adheres to the objectives established in Chapter 4

4.5.1 Planned Sprint Schedule

The Implementation Plan Schedule will broadly follow four phases of implementation. Each phase will be composed of one or more sprints, of approximately 1.5 weeks.

- **Base System Development (Sprint 1):** Establish the base multilingual modification to the Rasa system. Ideally establish a basic conversation.
- **Interface Development (Sprint 2):** Establish the web interface that communicates with a Rasa server. Investigate the possibility of hosting said interface on a cloud service.
- **Content Development (Sprints 3-5):** Developing content and documentation for the FáilteBot system.
- **Testing and Evaluation (Sprints 6-7):** Testing, evaluating, and refining the system as a whole in preparation for submission.

4.5.2 Product Backlog

The following is the product backlog for the Implementation Phase. For the purpose of brevity these will not be presented in full User Story format with Acceptance Criteria etc. Instead they will be grouped by Epics with a title and an estimated time cost.

4.5.2.1 Environmental Setup Epic

Story Description	Story Points
Setup development machine for Rasa	2
Implement multilingual modification for Rasa	8
Implement non-Roman character set for Rasa	2
Setup development machine for mobile framework	2
Research and Select Cloud Platform	6

4.5.2.2 FáilteBot Epic

Story Description	Story Points
As a User, I want to navigate through conversational menus in FáilteBot to access different stories	1
As a User, I want to get an appropriate fallback response when FáilteBot cannot answer my query	2
As a User, I want to ask FáilteBot, in English, about where to get advice in Ireland	4
As a User, I want to ask FáilteBot, in French, about where to get advice in Ireland	4
As a User, I want to ask FáilteBot, in Arabic, about where to get advice in Ireland	4
As a User, I want to ask FáilteBot, in English, about Accommodation in Ireland	4
As a User, I want to ask FáilteBot, in French, about Accommodation in Ireland	4
As a User, I want to ask FáilteBot, in Arabic, about Accommodation in Ireland	4
As a User, I want to ask FáilteBot, in English, about Education in Ireland	4
As a User, I want to ask FáilteBot, in French, about Education in Ireland	4
As a User, I want to ask FáilteBot, in Arabic, about Education in Ireland	4

4.5.2.3 Web Interface Epic

Story Description	Story Points
Develop a web chat interface for FáilteBot	10
Add web chat interface to hosted platform	3

4.5.2.4 Documentation Epic

Story Description	Story Points
System Testing	20
Update Documentation	20

4.6 Evaluation

The objectives from chapter 3 will be used as a heuristic to evaluate the success of this project. In cases where an objective is not achieved, an adequate explanation for its replacement or non-completion should be provided.

1. To improve upon the current model of settlement for refugees in Ireland, by providing previously available information in a more interactive digital format.
2. To provide content based on key topics of concern to refugees in Ireland.
3. To compare the delivery, and ease of use, of this system to the 'Guide to Living Independently' documents published by the Department of Justice.
4. To create a conversational agent that can provide a relevant answer, or helpful fallback response, to user inputs.
5. To allow improvements to the application by receiving and evaluating user feedback.
6. To provide a conversational agent that provides information in multiple languages
7. To create a system that is flexible and adaptable to content change
8. To develop and deploy an open-source multilingual conversational agent
9. To build this agent based on prior research and work done

10. To provide finite conversations in English and French, with room for integration of a sample language using a non-Roman character set (e.g: Ge'ez, Cyrillic, Urdu).

11. To demonstrate easy to use and modular architecture for developers.

12. To demonstrate a linguistically modular system.

13. To provide documentation for developers wishing to build upon this concept or its principles

14. To establish a system that is transparent, trustworthy, and does not store personal information.

4.7 Prototype

FIGURE 4.6: Wireframe of FáilteBot chat interface

The interface for FáilteBot will be a typical chat interface. Currently their is no intention to provide ancillary elements such as an account system or notifications.

The layout for the dialogue menu within FáilteBot will be simple and easy to navigate for the user. A user will go from the onboarding process, to language selection, to a menu of the topics covered by FáilteBot. At the end of every conversation they will have the opportunity to end the conversation and come back another time, or loop back to the main menu and choose a new topic. At any time a user will be able to alter the language of the conversation.

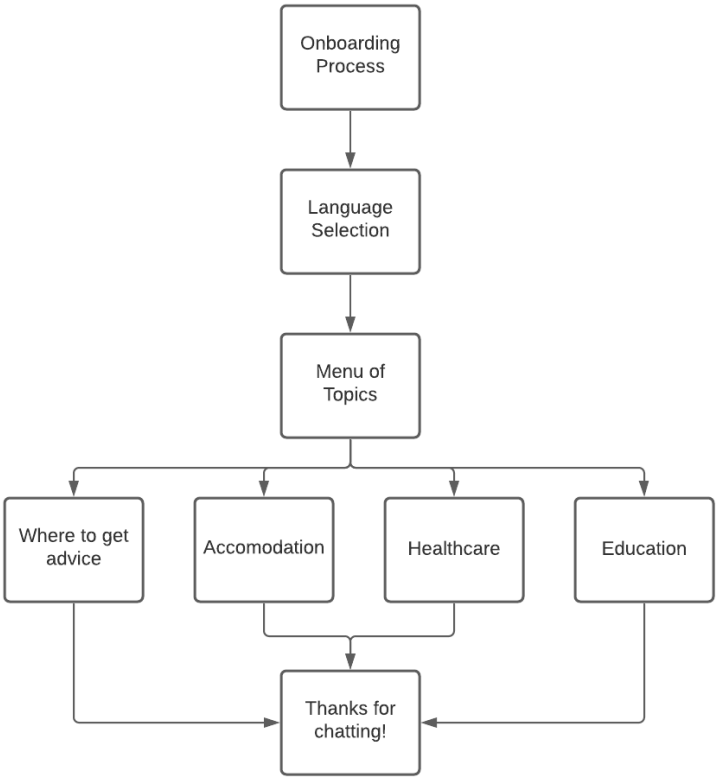


FIGURE 4.7: Wireframe of FáilteBot's menu structure

Chapter 5

Implementation

The following chapter will examine some of the issues encountered over the course of the implementation of this project. Each issue will be divided into three categories, Low-level problems, mid-level problems, and high-level problems. Low-level problems are problems that were easily solved, with little difficulty. Mid-level problems are problems that required a more involved process to resolve. Finally, High-level problems are problems that required an extremely involved process to resolve, or problems that could not be resolved.

5.1 Difficulties Encountered

- **Low-level Problems**:

    1. *Transcription of non-English texts* - Initially, there was difficulty in transcribing text from the non-English documents to the language packs. Ensuring the correct utterances were mapped over in all three languages was made difficult by a lack of Arabic or French experience. This issue would not be present if implemented by someone with foreign language skills in a real world application. However, it presented a roadblock for this project by creating the possibility of incorrectly transcribed content. To remedy this issue each transcription was checked with Google Translate before being entered into the language pack. This negatively affected time spent developing conversations. However, it was not time consuming enough to completely disrupt the workflow of the project. In addition to this, it improved the accuracy of the conversations.
    2. *Non-Latin Text* - The functions that extracted the correct utterances for returning to the user initially did not recognise non-Latin text. This was an

issue in the case of the Arabic utterances. Without fixing this issue users attempting to use the FáilteBot system for non-Latin alphabets, such as Arabic, Greek, or Cyrillic would not be able to correctly render utterances. The functions that extract utterances were altered to allow for the extraction of all UTF-8 characters instead. This fix was easy to implement and required little time to introduce. This change had the added benefit of allowing emojis to be displayed to the user, breaking up the visuals of the conversations and hopefully improving the user experience.

3. *Formatting Arabic text* - Transcribing text from the documents to the CSV files in Arabic was more difficult than anticipated. The document contained line and paragraph breaks that could not be easily edited out, due to the right to left formatting of Arabic text. The Integrated Development Environment (IDE) used to edit these texts for the FáilteBot system was built for left to right formatting, causing confusion whenever Arabic text needed to be edited inline. The easiest solution was to strip out any format characters before entering an utterance into a language pack. An online formatting extraction tool, was used to filter text before being input to the language pack. This solution did increase development time, but not greatly.

- **Mid-level Problems**:

1. *Reducing errors when transcribing conversations to code* - Due to the high volume of utterances, and the variety of conversations, large amounts of time was lost correcting human errors from this process. Correcting these errors was easy to do, but was consuming too much of the development process. To reduce the complexity of the system, a design document developing a multilingual chatbot was created. This document was used as a reference for designing all conversations. Following a set procedure for designing these conversations greatly reduced the number of errors produced during development. The document can be viewed in Appendix A.

- **High-level Problems**:

1. *System performance and training time* - The initial build of the system was noticeably slow to respond to user inputs. Training time for the machine learning model used to control conversation was also too high. A system that is too slow to respond to user inputs is one that could be incorrectly flagged as broken, and would reduce overall user retention. An investigation indicated there may be too high a volume of python functions being called. To improve performance, utterances were clustered into singular functions, only being broken up when user input was required or the conversation ended.

The findings of this investigation also contributed to the FáilteBot design document.

2. *Hosting the FáilteBot application* - Fulfilling technical objective 8 required the FáilteBot system to be fully deployed to a hosted web page. An attempt was made during Sprint 2 to host the application. However, the time spent researching and learning the skills to fully deploy the application was negatively affecting both the workflow of the FáilteBot project and other modules. A decision was made to not implement this feature, sacrificing an objective but allowing for the project to proceed to a greater state of implementation.

5.2 Actual Solution Approach

5.2.1 Architecture of Solution

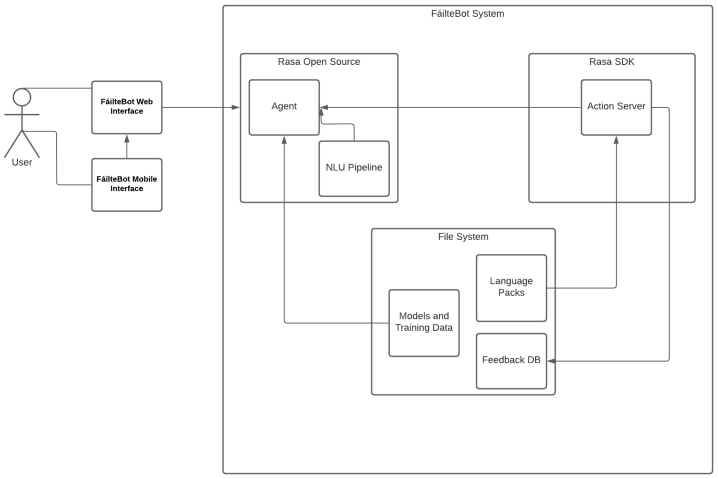


FIGURE 5.1: Initial FáilteBot system diagram

The initial plan for the FáilteBot system would be for a modified Rasa system, as seen on the right hand side of Fig. 5.1. In a typical Rasa system, there is no language pack. Instead, utterances are hard coded within the agent.

An alternative solution to the implemented one would have been to hard code three sets of utterances, one for each language. However, this would have exponentially increased the training time for the system. As stories require explicit references to these utterances, and utterances cannot share names, each addition of a language would increase the size of the overall code base by an unsustainable amount. In simpler terms, a bot that speaks one language may need 64 lines of code to be implemented. Two languages would require 128, and three languages would require 192. The training time would also be tripled,

straining resources. Finally, the cost of editing the agent's utterances would be higher, as it would require a user to be somewhat familiar with using an IDE, instead of handing them a basic spreadsheet.

In addition to this, it was originally envisaged that FáilteBot would have a hosted interface, accessible through a mobile app or website link. However, adequate respect was not given to the time and expertise required to move the system to an cloud platform, to securely exposing the ports, and obtaining a domain. After a week or so of efforts this functionality had to be removed as implementation was interfering with the workflow of the project.

The final created system is outlined in Fig. 5.2. Simply put, the project is the same but with the extra features removed. These could be added, if more time had been available, and they are an easy piece of implementation for any further work. A decision was also made to have any feedback or bug reports reported via a separate website, that would be linked through the chatbot. This was to reduce the complexity of the system and also integrate the kind of software a non-technical maintainer may be used to. In this case it is a link to a Google Form that sends the results into a Google Sheet that a maintainer could easily access.

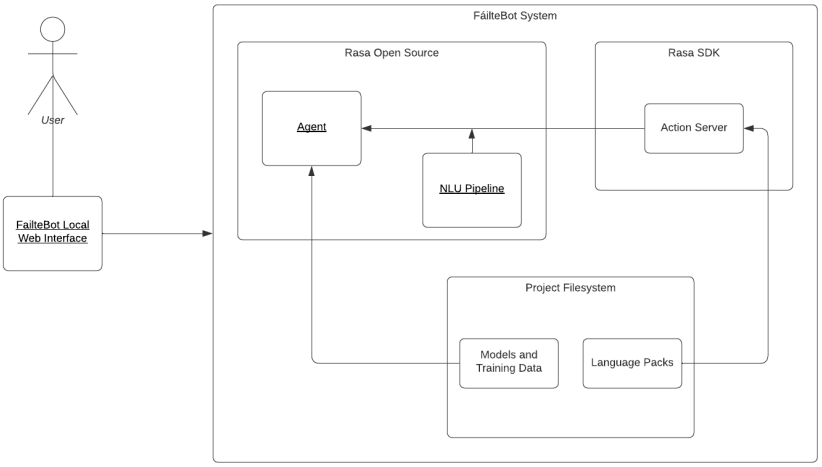


FIGURE 5.2: Final FáilteBot system diagram

5.2.2 Use Cases

The initial use cases outlined in Chapter 4 were, for the most part, followed almost exactly. For the purpose of brevity, these will not be reiterated within this section of the report.

The initial time assumptions for each use case were also quite accurate. The notable exception is the story for hosting the FáilteBot system on a cloud service. The initially assumed time cost of 3 hours was actually closer to 15 before a decision was made to shutter this feature. Unfortunately the amount of time needed to research and select an appropriate cloud platform, implement a sufficient security protocol, and find a domain were beyond the scope of the project. In hindsight aiming for a less ambitious project would have allowed for all objectives to be achieved fully. However, this was a valuable learning experience and the initial plans could not be described as regrettable.

A full view of the use cases implemented in the final system can be seen in Fig. , in the implementation schedule section.

5.2.3 Risk Assessment

The initial Risk Assessment performed in Chapter 4 outlined the several possible risks to the project. The following section will outline each risk, the actions taken to mitigate it, and examine if it still occurred. If so the impact of the risk will be examined. Finally, additional issues that may not hav been covered by the risk assessment will be examined.

- *Risk 1 - Code base is corrupted/lost* - Code base corruption/loss did not occur over the course of this project. The use of GitHub, however, ensured that if this occurred there would be no great loss of productivity. Additionally, GitHub allowed for features to be added via branches, ensuring a previously stable version of the system always existed.
- *Risk 2 - Available content is removed* - This risk did occur, towards the end of the project. After the outbreak of the Russo-Ukrainian war many materials available for refugees in Ireland were removed across government websites. The reason for this is unclear. Especially as none of the existing Guide to Living Independently documents were available in Ukrainian or Russian. The mitigation action of keeping a backup of this material prevented this from affecting the development of FáilteBot.
- *Risk 3 - Incorrect transcriptions* - This risk was much more pervasive than originally envisaged. On reflection the likelihood of this risk could have been update the highest possible category. However, the mitigation actions taken, as well as the additional measure of a heuristic design document prevented this as development went on.

- *Risk 4 - No available content for FáilteBot* - This risk was not an issue as the project began with the identification of suitable materials. It could be argued that this risk's inclusion was superfluous.

- *Risk 5 - Underestimate story points for work packages* - Though the estimation of time costs was for the most part accurate, a singular work package involving the hosting of the project was greatly over its projected time cost. This work package nearly disrupted the project enough to cause significant restructuring of the implementation plan. In hindsight the risk score could have been upgraded to the highest possible score.

- *Risk 6 - Translation required for transcription of FáilteBot content* - This risk did occur, which was accounted for during the initial implementation plan design. All needed translation was performed using Google Translate. Though not ideal, the costs and time required for a translation service was prohibitive.

- *Risk 7 - Rasa provides incorrect content* - The leave conversation option, proposed during the initial risk assessment, was not a feasible inclusion in the chatbot. Instead, the user can simply return to the main menu at any time by entering "main menu" or "return me to the main menu", in their chosen language. However, high quality training data for the NLU was included. The chatbot returning incorrect content was not an issue during testing either, possibly due to the button-based approach to delivering conversations.

- *Risk 8 - Feature creep causes project to be delayed* - Despite best efforts, this risk did occur. A difficult decision to reduce the scope of the project was taken, to ensure the integrity of the overall plan. In hindsight, further research on the requirements for hosting a system should have been taken during the Research Phase.

5.2.4 Methodology

The original indicated methodology for this project would be a topical one. To reiterate, topics and concepts would be identified and researched individually to obtain a holistic view of the overall project domain. Originally three key areas of knowledge were identified as areas that would be developed by the implementation of this project; Conducting Academic Research, System Architecture, and Conversational Agent Technologies.

The third area was the one most significantly developed over the course of this project. The required modifications made to the Rasa system architecture required extensive

experimentation and research. However, the time spent on this aspect returned valuable dividends in that a system that fits many of the specifications outlined during the Research Phase.

5.2.5 Implementation Schedule

The original Implementation Schedule for this project outlined 7 sprints, of approximately 1.5 weeks each, adding up to approximately 150 hours of work. There would be four phases to these sprints. The first phase would establish the basic modifications to the Rasa framework. The second would develop the interface for the project, the third phase would be sprints dedicated to content development. Finally, the final phase would be two sprints spent testing and evaluating the project. A key goal of this structure was to have a minimum viable product available at the end of each phase, which was achieved.

The Implementation Schedule for this project is one of the most altered aspects of this project. The original plan for week long sprints had to be altered. Sprints had to be altered and adapted to fit the time demands of projects outside of this module. This mean that Sprints averaged closer to 2 weeks per Sprint. However, this is affected by the initial and final sprints took longer as there was a significant amount of work required for them. The second sprint is also affected by time spent on a non-productive work package. However, the overall hours cost of approximately 150 hours was adhered to.

A Trello board was used to manage the project workflow. Another solution would have been a Jira board, but as this was a one person project, this solution seemed overly complex. An image of this board at the end of the project can be seen in below.

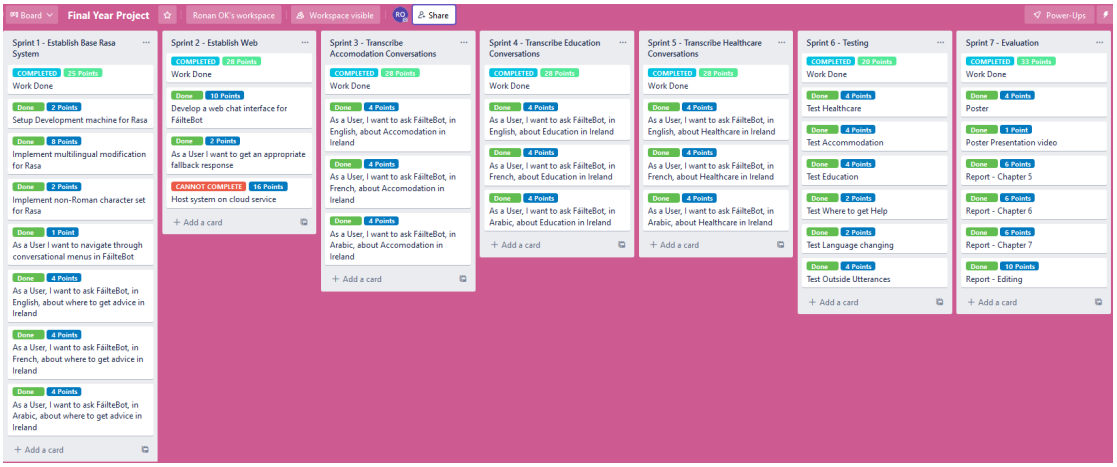


FIGURE 5.3: View of the project's Trello board at the end of the Implementation Phase

5.2.6 Evaluation Plan

The proposed Evaluation Plan for this project was to evaluate the success or failure of 14 outlined objectives, elicited from the domain research conducted. These objectives, and their use as an Evaluation Plan, was not altered between the Research and the Implementation Phase. It speaks to the relative utility of this approach that there were no changes between the two phases.

A full evaluation of these objectives is available in Chapter 6.

5.2.7 Prototyping and Final Product

There were two aspects of the FáilteBot system that were included in the Prototype section. The first is the actual chat interface of the system and the second is the menu system for the chatbot itself.

FIGURE 5.4: Initial prototype for the FáilteBot system.

The initial plan for the interface of the FáilteBot system would be a text-chat type interface, mimicking messaging apps such as Facebook Messenger or Microsoft Teams.

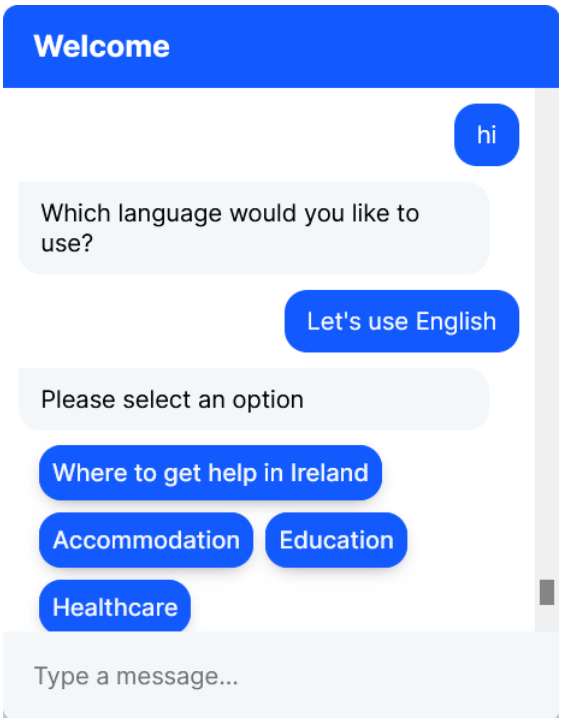


FIGURE 5.5: Final view of the FáilteBot system chat.

In an effort to keep the development time of the application down, a front end framework for the chat interface called 'rasa-webchat' was used. This restricted the format of the chat interface for FáilteBot. However, it closely adhered to the desired look of the FáilteBot interface. One change that may have been useful would be to persist the view of the buttons that had just been clicked in the chat interface. In screenshots the previous button options are invisible, which can be confusing when demonstrating the bot via images.

The planned menu system for the FáilteBot system was, for the most part, adhered to completely. The biggest change was focusing on making the conversation more closely resemble a 'closed-loop' system. The addition of an onboarding message conflicted with this desire and, as a result, was removed. Similarly, the "Thanks for chatting" functionality was removed as users may be confused by this.

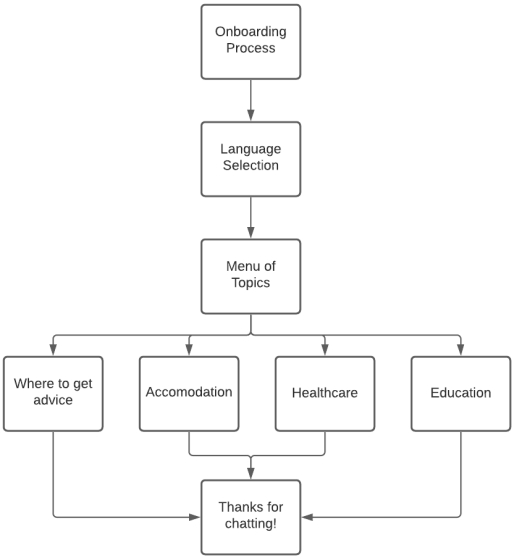


FIGURE 5.6: Initial conversation tree.

The final system can be seen in the below figure. The more looped style of conversation is more clearly visible in this architecture. Additionally, at any point the user can call the "language select" option. However, this ends the conversation they are currently accessing.

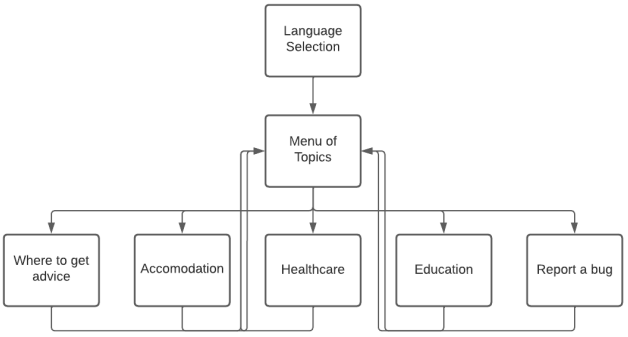


FIGURE 5.7: Final conversation tree.

Chapter 6

Testing and Evaluation

The following chapter will examine the success of the project. Evaluating this success will be based on the objectives elicited during the Research Phase of the project. Each objective will be examined and evaluated on a scale of Failure, Partial success, or Success. The objectives are outlined in Chapter 4. System testing will also be outlined, with self-selected accuracy metrics being used to evaluate the success of the test.

6.1 Objective Evaluation

1. *Objective 1 - To improve upon the current model of settlement for refugees in Ireland, by providing previously available information in a more interactive digital format:* Partial Success
   Full, real world evaluation of the FáilteBot system is out of the scope of a project like this. However, the system does provide previously available information in a more interactive, digital format. Information imparted via conversational agents has been proven to be more readily retained [12] by users, offering an argument in favour of FáilteBot versus existing resources.

2. *Objective 2 - To provide content based on key topics of concern to refugees in Ireland:* Success
   Topics such as Accommodation, Healthcare, and Education have been previously identified as topics of key concern to refugees in Ireland. The FáilteBot system provides information on these topics and can therefore be evaluated as a success.

3. *Objective 3 - To compare the delivery, and ease of use, of this system to the 'Guide to Living Independently' documents published by the Department of Justice:* Success

Due to time constraints, the FáilteBot system does not cover the complete set of information covered by the Guide. However, the information is imparted in an easier to access format, with the user being able to jump to particular topics of interest, rather than scrolling through a web page to find the important information.

4. *Objective 4 - To create a conversational agent that can provide a relevant answer, or helpful fallback response, to user inputs:* Success
   As previously outlined in the research section, there are two broad categories of conversational agents, 'finite' and 'infinite'. FáilteBot can be considered a 'finite' bot, with the system can providing relevant answers to a limited variety of topics. To account for inputs outside the scope of its knowledge FáilteBot offers a fallback response that directs the user to further help.

5. *Objective 5 - To allow improvements to the application by receiving and evaluating user feedback:* Success
   Though initially envisaged to be an embedded help system, FáilteBot now provides a system for user feedback via an external website, linked within the chatbot. The reason for this was to maintain the low barrier to entry for maintenance of the project, as well as reducing overall complexity of the system.

6. *Objective 6 - To provide a conversational agent that provides information in multiple languages:* Success
   Three languages were successfully integrated into the FáilteBot system. The system was made modular enough for any language, of any character set, to be successfully integrated so long as translations are available.

7. *Objective 7 - To create a system that is flexible and adaptable to content change:* Success
   Additional languages are easily integrated into the FáilteBot system. The steps for adding an additional language are simply the translation of a language pack and the addition of a menu button for the new language. Edits to existing utterances are also easy, requiring just an edit to a CSV file. However, a degree of technical knowledge is required to write new conversations for the agent. This is a space for possible development if trying to create a chatbot that requires little technical expertise.

8. *Objective 8 - To develop and deploy an open-source multilingual conversational agent:* Partial Success
   FáilteBot, as a system, is an open-source and multilingual conversational agent. As a system it is complete and ready to be deployed. However, to fully succeed under this objective this deployment would need to be successfully demonstrated.

As previously mentioned, this could not be achieved. Therefore this objective is only a partial success.

9. *Objective 9 - To build this agent based on prior research and work done:* Success
   The FáilteBot system was built with achieving its objectives in mind. These objectives were based on knowledge built during the Research Phase. This objective can be considered a success as every objective has been achieved to some degree.

10. *Objective 10 - To provide finite conversations in English and French, with room for integration of a sample language using a non-Roman character set (e.g: Ge'ez, Cyrillic, Urdu):* Success
    A finite set of conversations were integrated in French and English for this system. To fulfil the non-Roman character set requirement Arabic conversations were also added.

11. *Objective 11 - To demonstrate easy to use and modular architecture for developers:* Success
    The architecture for FáilteBot is a modification of the Rasa conversational agent system. The file system is intuitive and is not extremely divorced from the parent framework. In terms of modularity, language packs can be easily added with only a small change needed to add an additional language to the application.

12. *Objective 12 - To demonstrate a linguistically modular system:* Success
    Linguistically, the system is extremely modular, with little work required to integrate new languages, beyond translation. Stories are also not tied to a singular, base language. The overall architecture is language-agnostic.

13. *Objective 13 - To provide documentation for developers wishing to build upon this concept or its principles:* Success
    A design document for creating a multilingual chatbot within this framework was created over the course of development. Best practices pertaining to naming conventions and structuring of code were established. The document is available in Appendix A

14. *Objective 14 - To establish a system that is transparent, trustworthy, and does not store personal information:* Success
    The FáilteBot system does not collect any information from the user. Analytics data is not collected either. Though these data-points may be of interest, they could negatively affect use. The form for collecting bug reports or feature suggestions is via an anonymous web form, further divorcing the system from any retention of data.

6.2 System Testing

The following section will evaluate the usability of the FáilteBot system by examining three test cases. Each test case will rate the accuracy of intent retrieval, accuracy of agent response, and accuracy of text transcription. These metrics were chosen based on their importance for this particular use case, i.e; digitising existing resources. Utterance accuracy was evaluated with Google Translate and the source documentation for content.

6.2.1 Test Case 1 - A user wishes to change the bot's language from English to French

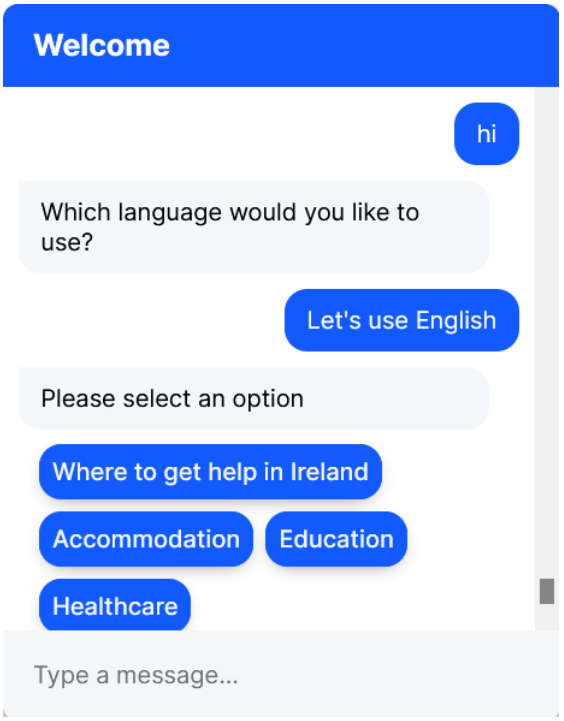


FIGURE 6.1: The user is accessing the project in English

In this test a user is accessing FáilteBot via English utterances. They inform the chatbot that they wish to speak to it in English and the chatbot successfully offers them the language select page. They choose French and proceed.

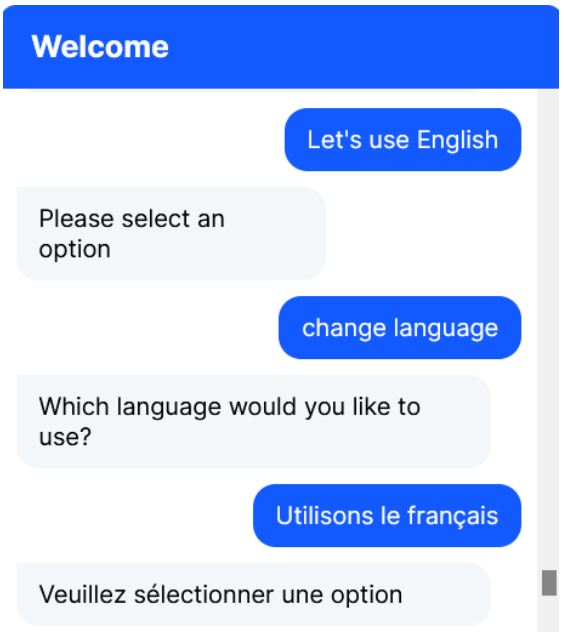


FIGURE 6.2: The system has changed to French

The quantitative evaluation of Test Case 1 is as follows:

Accuracy of Intent Retrieval	Accuracy of Agent Response	Accuracy of Agent Utterance
The user entered 'change language' and the agent successfully interpreted this as a desire to change language.	The agent displayed the full change language menu.	The agent's utterances were all correct, in English and French.

6.2.2 Test Case 2 - A user speaking Arabic wishes to see where to get help in Ireland

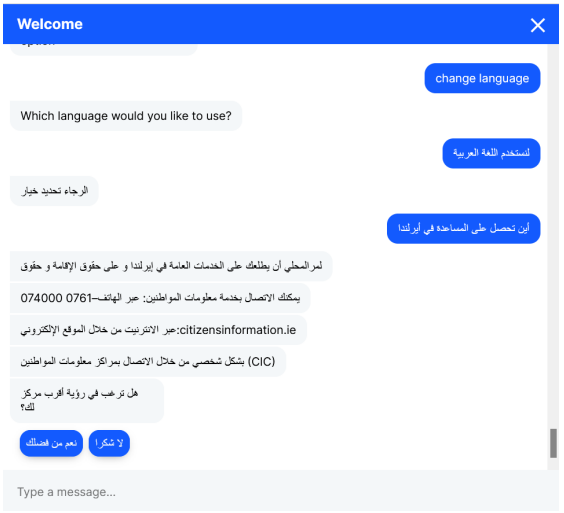


FIGURE 6.3: The user is accessing the project in Arabic

In this test a user is accessing FáilteBot via Arabic utterances. They wish to access the conversation on where to get help in Ireland. They navigate to the main menu and then select the option for 'Where to get help in Ireland'. This test case completed successfully, with all utterances correct as per Google Translate testing.

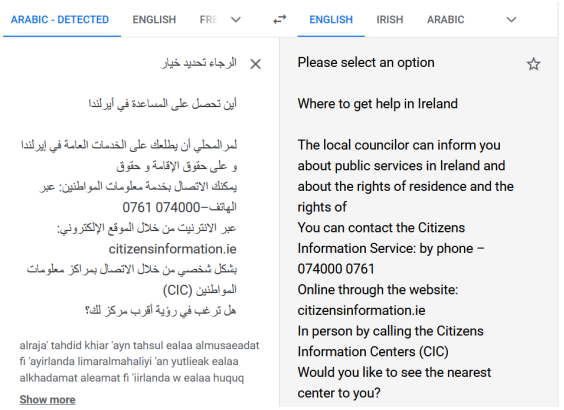


FIGURE 6.4: The system has displayed the conversation correctly

The quantitative evaluation of Test Case 2 is as follows:

Accuracy of Intent Retrieval	Accuracy of Agent Response	Accuracy of Agent Utterance
The user clicked on the 'Where to get help' conversation and the agent successfully began running this conversation script.	The agent displayed the conversation script and paused at points where a user input was required.	The agent's utterances were all correct in Arabic.

6.2.3 Test Case 3 - A user speaking French wishes to report a fault in the FáilteBot system

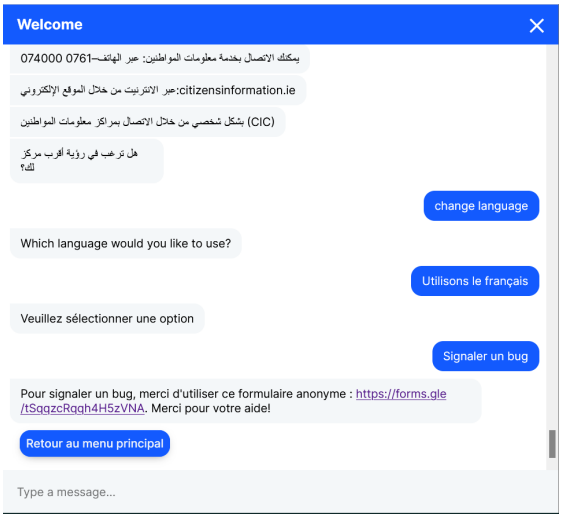


FIGURE 6.5: The user is accessing the project in French

In this test a user is accessing FáilteBot via French utterances. They have identified a mistake in the system and wish to report it back to the developers. They navigate to the main menu and select 'Report a bug'. All utterances were correct for this test case as per Google Translate.

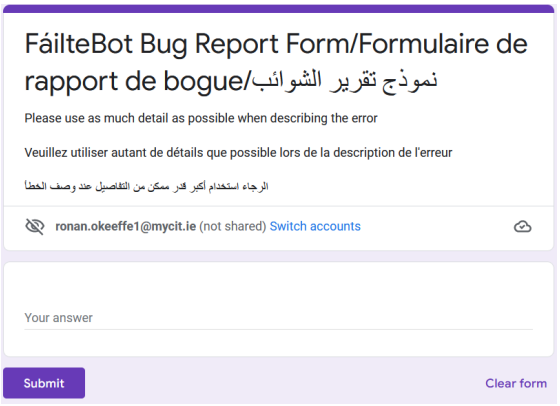


FIGURE 6.6: The system has linked them to a bug reporting form

The quantitative evaluation of Test Case 3 is as follows:

Accuracy of Intent Retrieval	Accuracy of Agent Response	Accuracy of Agent Utterance
The user clicked on the 'Report a Bug' conversation and the agent successfully began running this conversation script.	The agent displayed the conversation script and the link to the bug report worked successfully	The agent's utterances were all correct in French.

Chapter 7

Discussion and Conclusions

This chapter aims to conclude the research done and discuss the life-cycle of the project. First it will examine the merits of the proposed solution from the Research Phase. The work done during the Implementation Phase will then be examined. After Conclusions the chapter will offer discussion on further work that could stem from this project.

7.1 Solution Review

The proposed solution offered for this project is an extremely novel one, stemming from a confluence of research on refugee integration and conversational agents. The solution was inspired by the national discussion on the Irish government's refugee policy. The 2020 Program for Government indicated that the government's approach, going forward, would be an innovative one. From here the initial research question was built; can technology aid in the settlement of refugees in Ireland? Housing, medical, and monetary support aside, the research conducted indicated that refugees found the societal processes of the host country confusing. While this is only natural, a new environment is always confusing, the language barrier exacerbates this issue.

From this the question became how to provide a digital solution that provides information to users in a language of their choice. Conversational agents are a proven method of imparting information to users, creating a more human form of communication in comparison to static web pages. Unfortunately, the existing conversational agent market does not provide an agent capable of speaking any language. A modification to an existing framework would be required. After an extensive review of the existing conversational agent market, the open-source framework Rasa was chosen for modification. Rasa is built with a high level of modification in mind, making it the ideal candidate for this use case.

With a research question outlined, background research conducted, and technologies selected, the next step was to outline the objectives to measure the success of the proposed project. Objectives were split into two categories, domain objectives and technical objectives. These objectives were created based on the domain and technical research conducted to date. They outlined, specifically, the desired output of the project by the end of the Implementation Phase.

The final task of the Research Phase was to design the workflow of the project. An Agile approach to project management was selected as Agile presents prebuilt success metrics that could be easily applied to the project. In addition to this, incremental development suits the form of software development conducted in a Research Project. It was important that the project be able to easily adapt to the peaks and troughs of a typical software development life cycle, whilst maintaining the overarching goal. In January 2022 the findings of the Research Phase were presented to staff members at the university. With their seal of approval the project transitioned to an Implementation Phase.

7.2 Project Review

As previously mentioned, the workflow of the project was preplanned in the Research Phase. After approval from faculty the first sprint began, with the skeleton of the FáilteBot system being developed relatively quickly. The modifications required to the Rasa system functioned as planned.

The second sprint developed the front-end for the FáilteBot system, a basic web-chat interface that could be run locally on a computer. The subsequent sprints focused on developing conversations for the agent, with topics such as Healthcare, Education, and Accommodation being created. A design document for multilingual conversations was also developed, as lessons were learned from mistakes that presented themselves over the course of development.

Additional time was not found to fully deploy the system on a cloud service, leaving one objective only partially fulfilled. However, all other objectives were successfully met and the last two sprints were focused on system evaluation and the collation of documentation for the project.

An unintended aspect of this project's contribution to the field is the sidestepping of system complexity based on number of languages. The addition of languages to the FáilteBot system does not increase the complexity of model training or resources used. This is a relatively novel system in that respect, as the current literature[14] indicates,

the number of languages within a conversational agent exponentially increase the complexity of the system. FáilteBot avoids these complexities by maintaining a language agnostic approach, with a system completely blind to the language of its utterances. It should be noted, FáilteBot is a finite system, capable of responding in only to conversation about a selection of topics. In addition to this, its ability to parse non-button linked intents is limited. These intents have not been separated by language and so words of similar structure but different meanings could be misinterpreted by the chatbot. This could be resolved by separating intents by language.

As conversational agents become more and more prolific, a gap is being created between "majority" languages, such as English, and "minority" languages, spoken only in certain cultural groups. It is notable that Botpress, the framework with the most available languages, only covers 157 languages. With approximately 7,100 languages worldwide, this means 98% of world languages are unsupported by the current conversational agent market. If conversational agents are to become a common method of entry to source of information, it is important that there are systems built with minority languages in mind. The alternative is to purposefully deny informational resources to people based on their cultural group. An advantage of FáilteBot's decoupling approach is the closing of this gap.

7.3 Lessons Learned

A variety of lessons were learned from completing this project. The greatest was the importance of careful research and planning before beginning any project. The relative success of the project in achieving its goals. It is also notable that the one work package incorrectly planned was the one that was not fully achieved. This only reinforces the value of prudent planning. Another key aspect learned over the course of this project is the multi-disciplinary nature of new technology projects. The field of conversational agents alone covers computer science, mathematics, linguistics, sociology, and user-experience design. Introducing the element of refugees greatly expands the fields within the scope of this singular project.

In technical terms many lessons were learned. The most important is the utility of modifying preexisting technologies, especially open-source ones, to solve niche problems. This greatly reduced the overhead costs of this project whilst also positively contributing to technology's body of work. Finally, the value of clear documentation of new processes was extremely present. This allowed for the creation of a design document that ensured subsequent development within the project was less problematic. It also provides a

resource for developers wishing to work from this preexisting concept to create their own multilingual chatbot.

7.4 Conclusion

Overall the project can be considered successful in its desired aims. Of the 14 objectives, outlined during the Research Phase, all 14 fell under some kind of success. Only one objective was less than a full success. The project followed the work packages outlined in the Research Phase, with minimal deviations required. Work was completed in a timely and sustainable manner.

The project has demonstrated a possible digital solution to improving the integration of refugees in a host country. This has been done by creating a conversational agent that speaks multiple languages and is able to engage with users on a finite number of topics or direct them to additional resources. The project has also demonstrated the utility of modifying open source technologies to provide bespoke solutions for niche use cases. This offers a low cost and low complexity solution to developers. It also offers a practical solution for the closing of the access gap between languages. Finally, the project has created a usable framework for designing a multilingual chatbot in Rasa, which may be of use to developers outside this given use case.

7.5 Future Work

There are two broad directions in which the FáilteBot system could be taken. The first would be to improve the project and attempt to have it deployed in a real life context. Hosting the system and creating a mobile version would be two basic tasks, allowing for it to be more easily tested among real users. Deploying the project to a focus group of newly settled refugees in Ireland would be an extremely informative and useful step in this project's research. In addition to this, updating and properly translating the materials within would also be of use. The 2022 Russo-Ukrainian War has caused a surge of refugees to Ireland from the region. This would require Ukrainian and Russian language packs to be developed.

Alternatively the concept can be used by anyone seeking to create a multilingual chatbot using the Rasa framework. The required modifications are easy to make, and the codebase for FáilteBot has been made open-source using a Github repository. A paper on the research done by the FáilteBot concept is currently underway.

Bibliography

[1] "Chatbot market: 2021 - 26: Industry share, size, growth - mordor intelligence." [Online]. Available: https://www.mordorintelligence.com/industry-reports/chatbot-market

[2] B. Miller, "Government chatbots now a necessity for states, cities, counties," Jul 2021. [Online]. Available: https://www.govtech.com/products/government-chatbots-now-a-necessity-for-states-cities-counties.html

[3] L. Lai, K. A. Wittbold, F. Z. Dadabhoy, R. Sato, A. B. Landman, L. H. Schwamm, S. He, R. Patel, N. Wei, G. Zuccotti, and et al., "Digital triage: Novel strategies for population health management in response to the covid-19 pandemic," *Healthcare*, vol. 8, no. 4, p. 100493, 2020.

[4] A. B. E. Mabrouk, M. B. H. Hmida, C. Fourati, H. Haddad, and A. Messaoudi, "A multilingual african embedding for faq chatbots," 2021.

[5] "Rasa architecture overview," Nov 2021. [Online]. Available: https://rasa.com/docs/rasa/arch-overview/

[6] J. Mylet, "Lex," 2012. [Online]. Available: https://docs.aws.amazon.com/lex/index.html

[7] U. N. H. C. for Refugees, "Figures at a glance." [Online]. Available: https://www.unhcr.org/en-ie/figures-at-a-glance.html

[8] S. Malekmian, "Pleading for sanctuary through a language barrier was tricky enough – doing that online is even harder, some asylum seekers say," *Dublin Inquirer*, Jul 2021. [Online]. Available: https://bit.ly/3q9gBUb

[9] L. Higgins, "Health at risk over language barriers," *The Irish Times*, Jan 2007. [Online]. Available: https://www.irishtimes.com/news/health/health-at-risk-over-language-barriers-1.1292399

[10] *White Paper on Ending Direct Provision.* Department of Children, Equality, Disability, Integration and Youth, Feb 2021. [Online]. Available: https://www.gov.ie/en/publication/7aad0-minister-ogorman-publishes-the-white-paper-on-ending-direct-provision/

[11] *Guide to Living Independently.* Department of Justice, Aug 2021. [Online]. Available: https://www.gov.ie/en/publication/f6f7d-guides-to-living-independently/

[12] S. Altay, A.-S. Hacquin, C. Chevallier, and H. Mercier, "Information delivered by a chatbot has a positive impact on covid-19 vaccines attitudes and intentions," 2021.

[13] "Compliance and risk." [Online]. Available: https://www.staffnet.manchester.ac.uk/compliance-and-risk/risk-registers/recording-and-scoring/

[14] L. Tan and O. Golovneva, "Evaluating cross-lingual transfer learning approaches in multilingual conversational agent models," 2020. [Online]. Available: https://arxiv.org/abs/2012.03864

Appendix A

Appendix A

A.1 Introduction

The intention of this document is to provide a series of best practices to follow when building a multilingual chatbot under the FáilteBot template. In the experience of the author, it is easy to incorrectly transcribe utterances to a multilingual chatbot. Following these guidelines drastically reduces the chances of errors and greatly improves readability at a glance.

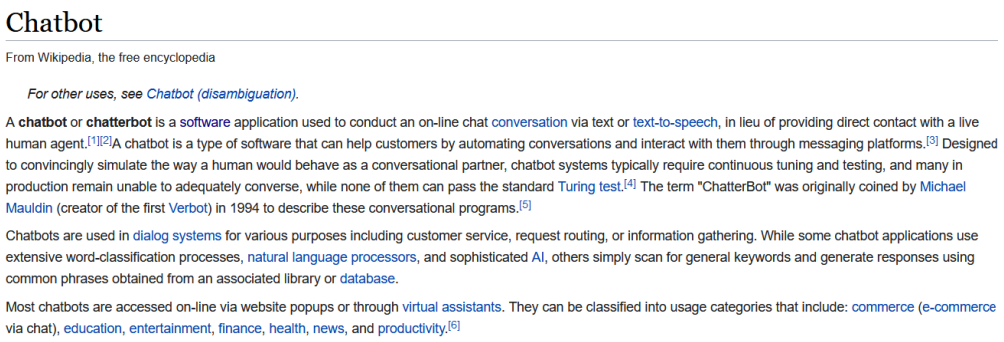
This document will provide a tutorial on how exactly to build a conversation. It assumes the developers are working from a text document and attempting to transcribe this to chatbot format. However, if the developers are facing a different use case, there are still important lessons contained within.

This document assumes that the reader is familiar with development of chatbots in Rasa systems and has familiarised themselves with the layout of a FáilteBot style chatbot. The source code for the FáilteBot project can be viewed at github.com/ronankff99/FailteBot.

A.2 Conversation Design

Before designing each conversation begin with the basics. Where does the conversation start, and where does it end?

Consider the following task: A developer has been instructed to turn the Wikipedia page for "Chatbot" into a conversation for a FáilteBot style chatbot. Examine the following screenshot.



Chatbot

From Wikipedia, the free encyclopedia

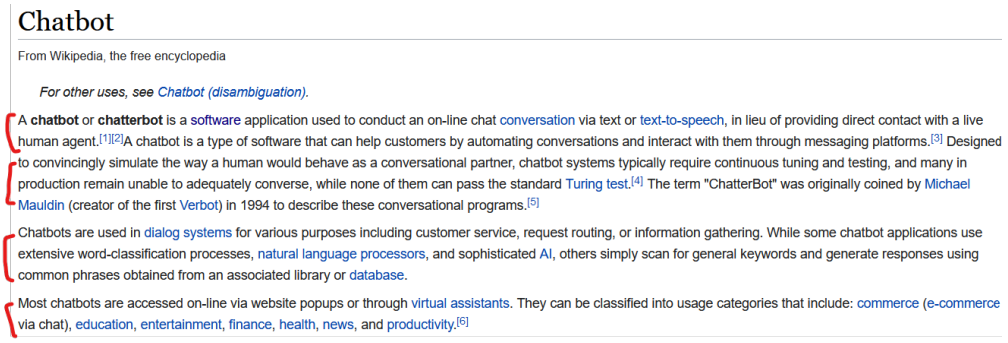
For other uses, see Chatbot (disambiguation).

A **chatbot** or **chatterbot** is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent.[1][2]A chatbot is a type of software that can help customers by automating conversations and interact with them through messaging platforms.[3] Designed to convincingly simulate the way a human would behave as a conversational partner, chatbot systems typically require continuous tuning and testing, and many in production remain unable to adequately converse, while none of them can pass the standard Turing test.[4] The term "ChatterBot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.[5]

Chatbots are used in dialog systems for various purposes including customer service, request routing, or information gathering. While some chatbot applications use extensive word-classification processes, natural language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases obtained from an associated library or database.

Most chatbots are accessed on-line via website popups or through virtual assistants. They can be classified into usage categories that include: commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity.[6]

Generally paragraphs should be turned into individual text messages. Paragraph headers or line breaks offer opportunities to insert a button prompt for users. This breaks up the conversation, and keeps the user engaged. Examining the above screenshot, we can see that there would probably be three separate text messages from the bot. The first paragraph is also quite long, so we might break this into two.



Chatbot

From Wikipedia, the free encyclopedia

For other uses, see Chatbot (disambiguation).

A **chatbot** or **chatterbot** is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent.[1][2]A chatbot is a type of software that can help customers by automating conversations and interact with them through messaging platforms.[3] Designed to convincingly simulate the way a human would behave as a conversational partner, chatbot systems typically require continuous tuning and testing, and many in production remain unable to adequately converse, while none of them can pass the standard Turing test.[4] The term "ChatterBot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.[5]

Chatbots are used in dialog systems for various purposes including customer service, request routing, or information gathering. While some chatbot applications use extensive word-classification processes, natural language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases obtained from an associated library or database.

Most chatbots are accessed on-line via website popups or through virtual assistants. They can be classified into usage categories that include: commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity.[6]

We now have a rough sketch of how our conversation will look like. To keep the user engaged we will insert a button midway through the four texts, and another at the end.

A.3 Story Design

With a general outline of conversation in mind, we turn to the development process. Proceeding to the stories file, we should carefully label the story for this conversation. If a conversation requires multiple blocks of stories, try to keep them together. This makes maintenance and correction of errors much easier. Finally, try to maintain the conversational ordering within your story scripts.

```
<!-- CHATBOT INFO STORY -->
## Learning about chatbots story
* chatbot
```

As you can see, we have carefully labelled the section for chatbot info stories, and also labelled the story block we are about to write. Finally, we have begun the conversation with a relevantly named intent. Try to name intents after the core of the conversational intent, this makes referencing it much easier. Whilst there is nothing preventing you from just calling intents "intent_1, intent_2....etc." this change makes story creation much easier.

The next step is inserting an action. The best practice for this is to have an action for uttering content between each user input. It is possible to split each text into its own action. However, this reduces system performance and creates overly complex stories. In terms of naming an action, if an action is for an utterance, begin it with "action_utter_". Then take the first three words of the utterance and add them to the action name. It is rare that actions will share the same name, if this occurs add a number or another word.

```
<!-- CHATBOT INFO STORY -->
## Learning about chatbots story
* chatbot
  - action_utter_a_chatbot_or
```

This action will account for the first two messages, and the button prompt before the last two. The next step is adding in the intent for the button press. In the case of

the Wikipedia post, a button that says "That's interesting! Tell me more..." would be useful in this context. While it is possible to add multiple buttons, this is unnecessary when transcribing a simple block of text like this. Multiple button options are useful when offering the user a way out of a conversation, or asking them to pick a topic to be discussed.

There are no set criteria for when to insert buttons, or what to label them. The important factor is to consider what would be a natural interjection in the conversation. Consider playing through the conversation yourself and seeing what would be a useful button prompt in this case. Consider also the overall tone of the chatbot for its particular use case.

```
<!-- CHATBOT INFO STORY -->
## Learning about chatbots story
* chatbot
  - action_utter_a_chatbot_or
* interesting
  - action_utter_chatbots_are_used
```

This is the final layout of the story for our example post. As you can see, there are only two actions, and two intents. We can also tell where each utterance begins in the conversation. This drastically improves the maintainability of the system. As a developer, at a glance you can tell what this story is about, what utterances are used, and what inputs the user selects.

A.4 Action Design

The next step is designing the actions we named in the stories file. Proceed to the actions.py file and create a blank action as follows.

```python
## CHATBOT INFO ACTIONS
class ActionUtterAChatbotOr(Action):
    def name(self) -> Text:
        return "action_utter_a_chatbot_or"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        return []
```

Notice the labelling and consistent naming of the action. We know that this action will account for two texts, and will end with a button prompt.

```python
## CHATBOT INFO ACTIONS
class ActionUtterAChatbotOr(Action):
    def name(self) -> Text:
        return "action_utter_a_chatbot_or"

    def run(self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        user_lang = tracker.get_slot("user_lang")
        dispatcher.utter_message(text=getText('utter_a_chatbot_or', user_lang))
        dispatcher.utter_message(text=getText('utter_design_to_convincingly', user_lang), buttons=[{
            "title": getText('utter_interesting', user_lang),
            "payload": "/interesting"
        }])
        return []
```

A "dispatcher.utter_message" call has been inserted for each text message. The second, which will have a button prompt, has a button array attached as well. Notice the naming for the utterance strings. To complete this conversation, repeat the above process for the next action.

A.5 Utterance Design

The final aspect of this is the transcription of utterances. When using multiple languages it is recommended that the developer begin utterance design in their native language. When the utterances are completed in this language, add in the translations for other languages. The reason for this is utterance transcription is the most tedious aspect of this process. It is best completed in large pieces.

Utterance values, as with actions, should all begin with "utter_" followed by the next three words in the sentence. If the sentence is shorter than three words, simply include the whole sentence.

A.6 Conclusions

Thank you for taking the time to read this very basic design guide for multilingual chatbots in the FáilteBot design pattern. Hopefully this document gives you the tools to begin building your own multilingual chatbot.