Implementation and Evaluation of a Chatbot to Crowdsource Geotagged Images to Detect Mosquito Breeding Sites

Marcel Dechert

Erstgutachter: Prof. Dr. Johannes Schöning

Zweitgutachter: Dr. Thomas Barkowsky

Fachbereich 03: Mathematik und Informatik

Universität Bremen

Diese Thesis wird für den Abschluss *Master of Science* eingereicht.

Universität Bremen April 2019

Eidesstattliche Erklärung

Hiermit erkläre ich, Marcel Dechert, eidesstattlich, dass ich diese Arbeit selbstständig und nur mit Zuhilfenahme der angegebenen Quellen und Hilfsmittel angefertigt habe. Diese Arbeit wurde weder in gleicher noch in ähnlicher Form einem anderen Prüfungsamt vorgelegt oder veröffentlicht.

Marcel Dechert

April 2019

Acknowledgements

I would like to thank Prof. Dr. Johannes Schöning for his advise and commitment. I am also thankful to Dr. Thomas Barkowsky for taking on the role as the second examiner. Very special gratitude is towards Artur Wojcik for the great collaboration and fellowship. I am very grateful to Professor Dr. Peter Haddawy, Dr. Myat Su Yin and all the students of Mahidol University involved in the project. And finally, last but not least, I also want to thank my friends and family for their support and encouragement.

Zusammenfassung

Das Denguefieber ist eine Viruserkrankung, die durch einen Stich der Gelbfiebermücke (Aedes Aegypti) übertragen wird. Weltweit gibt es jährlich bis zu 500 Millionen Infektionen bei Menschen. Das Dengue-Virus ist besonders in Südostasien verbreitet und gefährdet somit stark die Gesundheit der dort lebenden Menschen. Die Brutstätten der Überträger befinden sich meistens in künstlichen Behältern (alte Reifen, Eimer oder Vasen). Es gibt verschiedene Ansätze, um das Dengue-Virus zu überwachen und zu kontrollieren. So versucht das Projekt "Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition" von der Mahidol University, Brutstätten automatisiert zu detektieren. Dazu werden Bilder von Google Street View mit Hilfe eines Objekterkennungs-Algorithmus auf Brutstätten untersucht und klassifiziert. Ein Problem, welches dabei entsteht, ist die Aktualisierungsrate und Verfügbarkeit von Google Street View-Bildern. Im Rahmen dieser Arbeit wird ein Chatbot vorgestellt, der freiwillige Helfer dazu motivieren soll, die Bilderdatenbank zu aktualisieren. Der Chatbot wird auf dem weit verbreiteten Facebook Messenger veröffentlicht, wodurch eine hohe Zahl an Nutzern erreicht werden soll. Zusätzlich zu der Grundfunktion bietet der Chatbot weitere Funktionalität, wie allgemeine Informationen über das Dengue Fieber und den Krankheitsüberträger und die Möglichkeit zu erfahren, wie viele Fälle des Dengue Fiebers in einer bestimmten Region gemeldet wurden. Des Weiteren verfügt der Bot über Gamification-Elemente, wie eine Bestenliste aller Crowdsourcing-Teilnehmer und wöchentliche Herausforderungen. Letztendlich wird der Bot mit Hilfe von Studenten Mahidol University in Thailand evaluiert.

Abstract

Dengue is a viral vector-borne disease, which is transmitted by the Aedes Aegypti mosquito. With an estimated 500 million cases a year and no effective vaccine, dengue poses a threat to public health. It is particular prevalent in south east Asia. Mosquito breeding sites are often in artificial water containers, e.g. old tires, buckets and vases. There are various approaches to monitoring and controlling the disease. The project "Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition" aims to automatically detect those breeding sites. Google Street View Images are processed by an object recognition algorithm to detect and classify the breeding sites. The Google Street view image library is limited to the time when the pictures were taken by Google and limited to locations accessible by car. Therefore, in this work, we present a chatbot to increase the coverage of geotagged images. The chatbot enables volunteers to contribute newer and missing pictures to the database. The chatbot is developed and published on the widely spread Facebook Messenger platform. Moreover the chatbot includes gamification elements, functionality to retrieve the number of reported dengue cases in an area as well as general information on dengue. The proficiency of the chatbot is evaluated in cooperation with students of Mahidol University in Thailand.

Table of contents

List of Figures

Chapter 1

Introduction and Motivation

Dengue is a viral vector-borne disease, which is transmitted by the Aedes Aegypti mosquito (Figure 1.1) [1]. With up to 500 million cases a year and no effective vaccine, Dengue poses a threat to public health [2]. It is particular prevalent in Southeast Asia [3]. To better monitor the spread of the Dengue virus the World Health Organization (WHO) [4] is using three different indicators. The *House Index* (HI) indicates how many percent of households have larvea of the Aedes Aegypti. The *Container Index* (CI) represents the percentage of containers infested with larvae. The *Breteau Index* (BI) measures the amount of containers containing larvae per 100 households. The indices are designed to be auxiliary in predicting future outbreaks and target interventions.

To generate the indices, the breeding sites need to be mapped. Mosquito breeding sites are often found in artificial water containers, e.g. old tires, buckets and vases (Figure 1.1). The mapping is mostly done manually by workers of the Ministry of Public Health, which ensures high quality of data but does not scale to cover bigger areas [5].

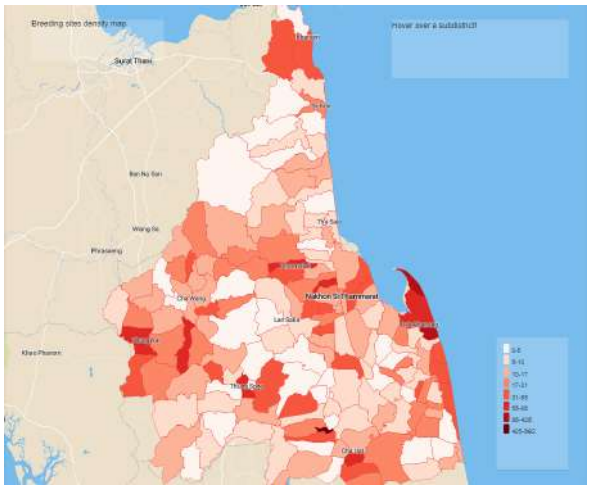
Besides this approach there are several projects, which try to support or replace the manual mapping through information technology. Many different approaches have been attempted, e.g. using satellite images to find breeding sites [6], using mobile application to detect breeding sites with the help of the public [7] or the establishment of a predictive model based on weather data [8]. This work is based upon the project “Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition” [in press]

Fig. 1.1 Aedes Aegypti and its breeding sites

at Mahidol University in Thailand. The project aims to automate the mapping of breeding sites. By leveraging the new technologies in machine learning (region-based convolutional neural networks in particular) and the broad availability of geotagged images, a pipeline for object recognition to detect breeding sites of the Dengue vector was developed. If there is a strong correlation between the existing indices and the frequency of breeding site, the system is an effective alternative to the manual surveys. The breeding sites are recognized by the object detection algorithm and then visualized by the dashboard (see Figure 1.2). *Google Street View* (GSV) is the primary source of images. GSV provides a vast set of images with good coverage and historical image data, which is very useful to compare the results to other indexes in order to validate the approach. On the other hand it is not well suited to function as the sole source of images as most of the pictures are out of date. Furthermore GSV often does not cover places that are not accessible by car or are on private property, which could be problematic, as breeding sites are more likely to be adjacent to houses.

For this reason, we developed the mobile app *Dengue-Detector*. It aims to increase the coverage of geotagged images by (geospatial-) *crowdsourcing*, i.e. enabling volunteers to contribute newer and missing pictures.

(a) Old tires recognized by the system



(b) Potential breeding sites on a heat map

Fig. 1.2 Object recognition and visualisation

One of the most challenging aspects of crowdsourcing is to gain and maintain a big number of active participants [9]. Especially for non-commercial projects, which cannot afford to offer monetary rewards, it is hard to reach enough people to make the project viable. Time consuming steps like installation and registration of an app are regularly big enough obstacles to keep people from participating in citizen science projects or other kinds of humanitarian or public health related endeavours.

Nowadays users rarely install new apps [61]. Only very few apps are added to the home screen and used frequently [10]. Especially in Asia, where desktop platforms were never widely spread [62], the mobile first-paradigm is very present [63]. Instant messaging platforms on mobile have well exceeded the monthly active users of more traditional media such as email or websites [11]. The instant messaging platform *Facebook Messenger* has a market share of 60 % [64]. As one of the de facto standards for peer to peer communication in south east Asia Facebook Messenger is the appropriate target platforms for the *Dengue-Detector*-app, as it provides a comprehensive Application Programmable Interface (API) to integrate chatbots.

1.1 Scope of this Thesis

In this thesis a disembodied *conversational agent* (CA) for the purpose of collecting geotagged images, called *Dengue Detector* (DD), is conceptualized, implemented and evaluated. The chatbot is able to receive and prompt for user locations and is then offering directions on how and where to take pictures. Upon receiving the images, the bot processes the provided images and adds geotags to them. The first step is to design a conversational flow for the interaction with the chatbot. Before implementation the available chatbot technologies are assessed. We analyse the advantages and disadvantages of different frameworks like *Dialogflow* and *Wit.ai*, which are very powerful tools for handling the Natural language processing (NLP) and machine learning aspects. To enable the chatbot to process images and locations a separate component is implemented in Python.

Gamification elements are built into the chatbot to ensure a more long term interest in user contribution. The chatbot provides elaborate statistics of how many pictures were collected and presents a leaderboard of the most active users. It can entice the users by giving public challenges to participate in as well as personalized tasks, such as to take pictures in a specific place. In addition, the server is able to send regular notifications and updates to (inactive) users. Those reminders are designed to keep users active for prolonged periods of time.

An avatar and character for the chatbot is designed. The Dengue Detector chatbot is able to make small talk and respond to many kinds of small requests in a logical way, but is generally specialized in handling requests in relation to the collection of geotagged images. It is unable to answer complicated or complex questions from other domains. The chatbot is also designed to inform and educate about Dengue in general and the public health risks it presents. Through the chatbot users are able to learn about the impact of Dengue on global health, about the infection and the course of the disease, as well as the vector and its breeding sites in particular.

Finally, we evaluate how the chatbot is used and if it is able to motivate users to participate in the crowdsourcing of geotagged images. The evaluation is conducted in cooperation with students of Mahidol University. The results are presented descriptively and interpreted.. We

aim to deduce patterns in the interaction with the bot, that can be taken into consideration for future development and improvement of a chatbot to collect geotagged images.

1.2 Complementary Thesis by Artur Wojcik

In addition to this work, the thesis "Crowdsourcing of geotagged images to detect mosquito breeding sites by using mobile applications" by Artur Wojcik [12] also deals with the collection of geotagged images to detect Dengue breeding sites. The overall project consists of three different channels for collecting geotagged images. The first part is the stand-alone Dengue-Detector app, which guides the user in taking the pictures. Secondly, the functionality of the app is integrated into the existing disaster reporting and alerting platform mobile4D, which is being developed at the University of Bremen. The third part is the development of a chatbot on Facebook Messenger, which is the focus of this work. All of the collected images will be sent to the Dengue Detector-server, which was developed by the students of Mahidol University. As visualized in Figure 1.3 the basic functionality of the chatbot was implemented in collaboration with Artur Wojcik.

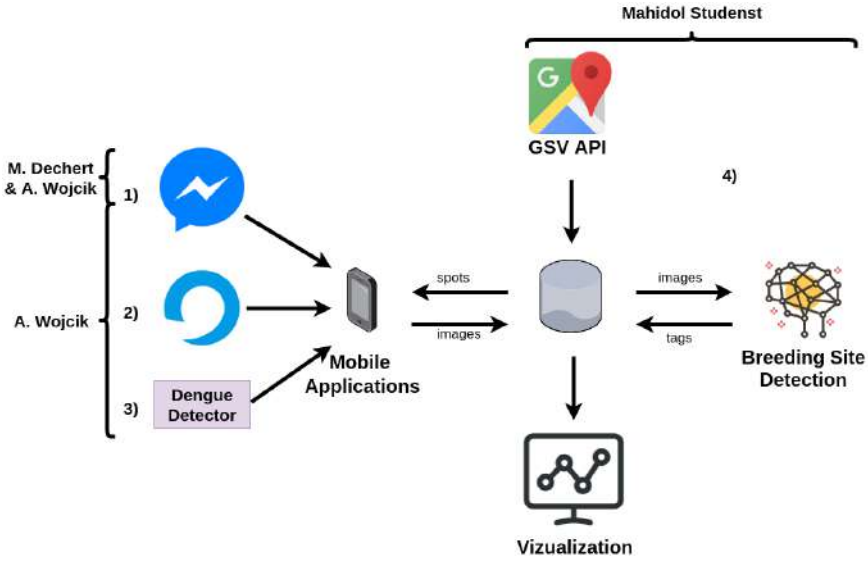


Fig. 1.3 Complementary Thesis of Artur Wojcik

All further work on the chatbot is exclusively part of this thesis. The collaborative part includes the following tasks:

- Implementation of the basic intent using the Dialogflow API.
- Integration with Facebook Messenger platform.
- The Representational State Transfer (REST) API to send the user location to the server and receiving and displaying the places where pictures need to be taken.
- Geotagging the pictures.
- Sending the geotagged images to the server.

1.3 Structure of the Thesis

In chapter 2 we introduce the basic concepts relevant for the thesis by presenting the related work in the fields of Dengue, conversational user interfaces (CUI) and crowdsourcing. The technological foundation, features and the flow of dialogue for the chatbot are conceptualized in chapter 3. In chapter 4 the details of the implementation are described. The results of the usage of the live chatbot during the evaluation study are documented and discussed in chapter 5. Finally, we draw conclusions from the project and present ideas for further research in this area in chapter 6. Every picture used in this thesis is either public domain or created by the author. Every other source is referenced explicitly.

Chapter 2

Related Work

Three main approaches exists to control Dengue. In the following we describe the different approaches in relation to technologically supported attempts to antagonize Dengue (Section 2.1). In Section 2.2 an overview of the field of chatbots is given. Chatbots in the domain of healthcare are highlighted in particular, as they entail technological and design approaches relevant to this work. Finally, we elaborate on what is different in this work compared to the existing work on auxiliary technology to control Dengue (Section 2.3).

2.1 Dengue

The Dengue fever (also known as breakbone fever) viruses are passed on to humans through the bites of an infective female Aedes mosquito. After an incubation time of two to seven days, Dengue can cause headaches, muscle pain, high fever, rashes, nausea and vomiting. The primary infection will incapacitate the infected person for about a week. Recovery from infection by one Dengue virus serotype provides lifelong immunity against that particular virus serotype. Upon a second infection by one of the three other serotypes, Dengue can be deadly [3].

As there is no effective vaccine, the control of Dengue relies on traditional environmental management and applying insecticides to combat vector mosquitoes (Figure 2.1). To better target such interventions, IT-supported techniques to map the risk of Dengue have been

worked on by numerous research groups around the globe. Different approaches to map and predict Dengue have been developed. For lager scales, satellite remote sensing data (RS) can be used. Also proxies such as meteorological and socio-economical metrics can help to predict outbreaks. Manual surveys of breeding containers yield indices that correlate with the actual number of cases, but require a lot of labour. All of these approaches have significant disadvantages, as they either do not scale well, or do not have a high enough resolution. Consequently, various approaches enabled by information technology have been conceptualized and tested by the scientific community as well as governmental and non-governmental organisations. These IT-supported approaches can be distinguished by their point of intervention. First, there is the approach to find breeding sites to kill mosquito larvae before the disease can spread. Secondly, there are predictive models, which help to approximate when and where an outbreak will happen, as this information is useful to plan control measures like a travel lock. In the following we will distinguish between three ways of obtaining meaningful data, where the first uses data mining approaches to predict future events and the latter two focus on vector control.

- Predict outbreaks with data mining.
- Breeding Site Detection via object recognition.
- Crowdsourcing approaches.

2.1.1 Methods to Predict Dengue

There have been several attempts to create predictive models for Dengue by various researchers. They use different kinds of available data ranging from Dengue case data, to meteorological and socio-economical data. Reich et al. developed a real-time forecasting model for Dengue across Thailand based on the reported cases of the years between 1968 and 2014. They compare the results to predictions made with fully reported data [13].
Aburas et al. used Neural Networks to predict the number of confirmed Dengue cases. For their study, six years of records of four Parameters were used to check if they correlated with the future number of reported Dengue cases. The parameters were mean temperature,

mean relative humidity, total rainfall and the total number of Dengue confirmed-cases [14]. Weather data can be good indicators, since factors like humidity, rainfall and temperature ease the survival of the mosquito and its eggs.

Fig. 2.1 Vector Control with insecticides (left) [65] and a Mosquito Trap (right) [66]

In another attempt to predict the distribution of cases of Severe Dengue (also known as Dengue Haemorrhagic Fever) Lauera et al. developed a statistical model, which uses the same biological parameters as Aburas et al. but also taking into account demographics, i.e. population density, as transmission of Dengue is more frequent in regions with a bigger populations [15]. A different approach, presented by Althouse et al., is to look for correlation between the use of Google search terms related to Dengue and the Dengue incidence data in a particular timeframe. This way they found a strong correlation between Google searches and the number of Dengue cases in the Singapore and Bangkok data [16].

2.1.2 Breeding Site Detection via Object Recognition

Another approach to get higher resolution results than with predictive models is to use object recognition on images to identify breeding sites. This method has the potential to be scalable given the vast availability of pictures online. Chang et al. developed a Dengue surveillance system for developing countries using satellite imagery from Google Earth in ArcGIS to accurately identify areas with a high level of mosquito infestation. Existing public health data was used to create a layer to represent each neighbourhood [6]. Mehra et al. present an object recognition process for identifying stagnant water bodies in images. Images from different sources such as Google, Flicker and others are used to detect potential breeding sites

of the Dengue and Zika vector. The results are visualized by a heat-Map [17]. Breeding sites are often hard to find. To address this issue, Suduwella et al. present a process specifically tailored to find breeding sites, i.e. stagnant water bodies by using drone images of otherwise hard to access areas [18].

2.1.3 Crowdsourcing Approaches

There are several projects and apps that collect reports of incidents or breeding sites by volunteers. *Mo-Buzz* (Figure 2.2) is a mobile application in which users can report Dengue hotspots to keep track of the most affected parts of the city. People can use this information to take preventive steps to protect themselves from mosquitoes, if they are living in, or travelling to these areas. Citizens can report Dengue breeding sites and Dengue symptoms to the local authorities using simple forms on their mobile app, which are automatically geotagged. This information is used to monitor the disease and target intervention. The app also includes a health education section with information, e.g. on where the next hospital is to get treatment [19].

Similarly *DengueChat* (Figure 2.3) lets users report breeding sites and communicate peer-to-peer to organise intervention on a grass roots level. The project is a cooperation between the University of California and the Brazilian government piloted in 2015. On the website users can take pictures of breeding site and send subsequent pictures of the destroyed breeding site. This way users can collect rewards like badges and gain points on a leaderboard. In addition, *DengueChat* has a lot of social network features, like a timeline, user profiles, and groups [20].

Fig. 2.2 Mo-Buzz app

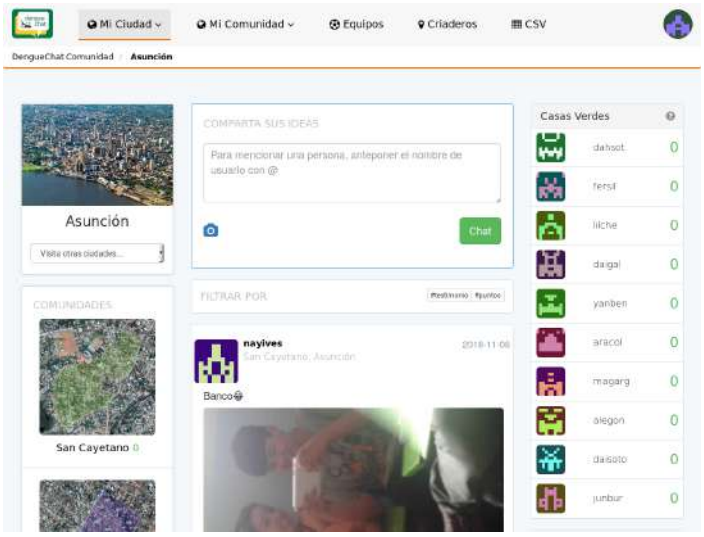


Fig. 2.3 DengueChat dashboard

TargetZIKA (Figure 2.4) is a mobile and web application to report breeding sites in Brazil. It collects data on where and when (spatio-temporal) breeding sites were reported by citizens. Those micro-reports include pictures and a description of the breeding site. On the website, users can view a risk map which also takes other factors into account. Empirical data such as the number of ZIKV cases reported and number of casualties due to Microcephal (which is a symptom of Zika) also influence the risk map [21].

Mosquito Alert (Figure 2.5) is a comparable mobile application from Catalonia, which encourages citizens to find and report mosquitoes and breeding sites. The content of the pictures is then validated by experts or by other users. Each report consists of the kind of report, a picture, a description and further information. The app focuses on user engagement through gamification. Every user is assigned a score between 1 and 100. The score depends on the users participation. For every contributed picture or validation, especially if they were reported from locations far apart, the score will rise [67].

Fig. 2.4 TargetZika risk map screenshot

Fig. 2.5 Mosquito Alert app screenshot

All of the afore mentioned crowdsourcing applications have in common, that they have no automated way to verify the breeding site in the pictures. Also most of them require installation of native Android or iOS apps. This work addresses both those issues by using a object recognition pipeline to analyse the pictures for breeding sites, as well as using the Facebook Messenger platform as the UI, which is already installed on the majority of devices in the target audience.

2.2 Chatbots

A chatbot, also known as a conversational agent, is a computer program, which interacts with a user through natural language dialogue and provides some form of service [22]. Chatbots allow for intuitive and accessible human-computer interaction and are therefore frequently used in various domains of application such as customer support, e-commerce, education and learning, information retrieval, health care and many more [23]. In the scientific literature, healthcare (Section 2.2.1) is a frequent topics. Some publications are similar to this work in terms of structure, content or development methods.

In most of those cases chatbots are goal oriented, i.e. they are programmed to fulfill requests specific to one domain (closed domain). Other chatbots like *Mitsuku* [68] or *cleverbot* [69] are designed for recreational communication, i.e. they respond to questions of any domain often in a clever and entertaining way (open domain). The first conversational agent to receive major attention was *Eliza*, by Weizenbaum, which emulates a conversation with a psychotherapists [24]. *Eliza* was based on simple keyword matching and was very limited in recognising and maintaining contexts within a dialogue. Following a simple set of rules, the keywords uttered by the user were incorporated into simple follow-up questions. If there was no keyword found, a general response (like “tell me more about that!”) was assembled. It was also relying on user input to keep the conversation going.

In the early 2000s Artificial Linguistic Internet Computer Entity (*ALICE*) was considered the most advanced conversational agent. It was able to reply to a greater amount of different requests by utilizing the Artificial Intelligence Markup Language’s (AIML) pattern-matching. With AIML it is possible to implement a nearly human-like dialogue for a limited domain, but those conversation still tend to be repetitive, as the CA struggles to maintain context in longer conversations [70]. More noticeable to the general public, voice-based conversational UI have gained popularity since Apple released its voice assistant *Siri* in 2011. In recent years, natural-language user interface have become ubiquitous in speech-based assistants like Amazons *Alexa*, Microsoft’s *Cortana* and the Google Assistant. Because of their AI capabilities and access to the internet, they are able to respond to almost anything with a proper answer.

This impression is supported by the Gartner hype cycle for emerging Technology. According to Gartner (Figure 2.6), *Virtual Assistants* are already beyond the peak of inflated expectations, while *Conversational AI Platforms* are steadily approaching the peak.

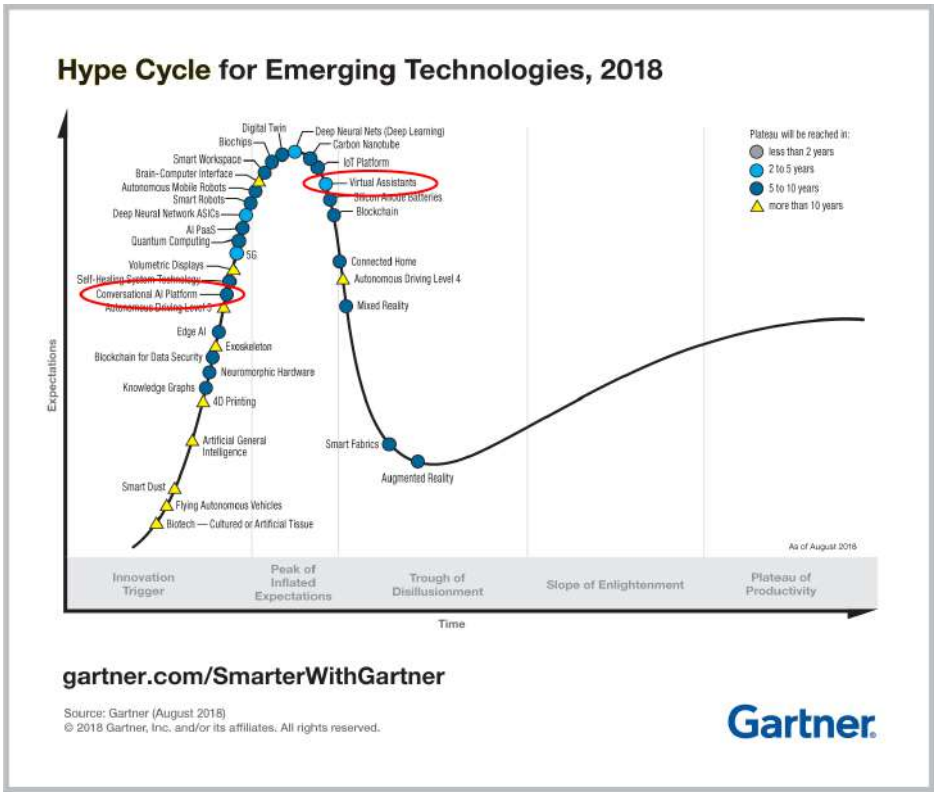


Fig. 2.6 The Gartner Hype Cycle for Emerging Technologies 2018 [71]

Accordingly, the number of publicly available chatbots has grown rapidly since the official support for chatbots by Facebook began in 2016. As of May 2018 Facebook Messenger alone had over 300,000 monthly active chatbots [72]. Essential for this quick development is significant technological progress in certain areas. The most note worthy enabling factors for the increased popularity of chatbots are the ascend of universal chat platforms (Facebook Messenger, Telegram, Slack, etc.) and the advances in Machine Learning (ML) and Natural Language Understanding (NLU), which have become widely accessible as online services [25].

Conversational agents have several advantages over graphical user interfaces. They are intuitive to use for everybody, as they do not require users to navigate through all the

features of the application via a menu. Furthermore they are available around the clock and in comparison to their human counterparts, they are giving users instant response to their request. All of those traits make it attractive for businesses to use chatbots in customer support so they can handle recurring requests timely and with no human involvement.

2.2.1 Chatbots in Healthcare

In the following an overview of the existing work on chatbots in the field of healthcare is given. Healthcare is a popular field of application for research on chatbots. Publication which are only remotely related to the topic of this work will be mentioned briefly, while publications that are most relevant to this thesis will be discussed in more detail. There is a lot of interest in research and development of chatbots in the medical field, as conversational agents have a lot of potential to be more cost-effective, reduce the time spent asking questions to make the right diagnosis and make knowledge more widely accessible [26]. By examining the scientific literature, we identified three major use cases of chatbots in healthcare: Mental health counselling, accessing medical knowledge and diagnosing diseases according to symptoms.

Divya et al. present a medical chatbot that provides the patient with a diagnosis of the illnesses and subsequently delivers detailed information on the disease. The user input is matched for symptoms. From these symptoms the system deducts a shortlist of potential diseases. The chatbot then asks further questions to confirm and presents a shorter list until there is a clear diagnosis. In case of a major disease, the data of the symptoms will be sent to a specialist. For minor diseases the bot suggests some first aid and recommends to visit a doctor [27]. Madhu et al. developed a similar personalized medical assistant, which is also able to list available treatments. The System can additionally deliver information on the composition of the medicines and how to appropriately use them [28]. Other similar systems like Health On-Line Medical Suggestions (*HOLMeS*) encompass data mining to obtain better results [29]. Lokman et al. propose a chatbot for diabetes patients called Virtual Dietitian (*ViDi*), that should function as a virtual physician/doctor. It starts off by asking question and analysing all answers until the disease is diagnosed. In the end it gives advice on the diet [30].

Another interesting use case of chatbots is to remind people to take their medicine, while ensuring it is done properly [31]. Morales-Rodríguez et al. envision a chatbot to diagnose the Generalized Anxiety Disorder (GAD). In the process they present an architecture for emotional responses in the context of clinical psychology [32]. *Gyant.com* is a chatbot which focuses on diagnosis and treatment of non-urgent conditions [73]. It leads the user through a series of easy to answer questions. Because of the frequent use of emojis, memes and a sense of humour, the chatbot appears humanlike (Figure 2.7)). It is also capable of diagnosing mosquito-borne diseases like Zika. *Florance* [74] is a personal health assistant available on Facebook Messenger. Besides information on diseases, it features a point system, which rewards users, who track sportive activities. The most active users are listed on a leaderboard (see Figure 2.7). Many of the above mentioned chatbots include instructions for humans. This is an aspect also present in the Dengue Detector chatbot.

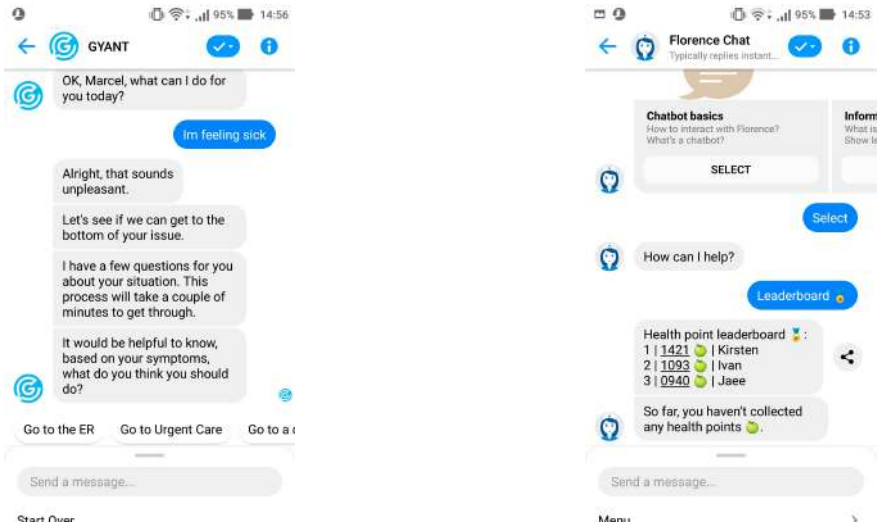


Fig. 2.7 Conversations with Gyant (left) and Florance (right) on Facebook Messenger

There has been a lot of research in regards to the use of chatbots in the field of mental health. The bots can be distinguished by their target audience, i.e by age or the kind of sickness that the bot deals with. Kowatsch et al. present a text-based healthcare chatbot (*THCB*) system that was first tested in the domain of childhood obesity. It got promising result regarding enjoyment and attachment bond between patient and bot [33]. Sometimes it is easier for people to talk to a non-human entity as they will not have to fear judgement

or any other social consequences. Cameron et al. developed a chatbot for counselling on issues such as Anxiety, Depression, Obesity, and Alcohol/Drug misuse. The chatbot aims to simplify users finding and accessing documents in a self-help library. Patients can also talk with the bot about their well-being [34]. Another group in need of attention to their mental health are military veterans. The *SimCoach* attempts to lower the barrier for (former) military service members to open up to information on mental health and to eventually initiate contact to healthcare professionals [35]. In order to make use of chatbots in this sensible field, it is important to recognize the users emotional state. This can be done by detecting emotions and generating responses accordingly. Lee et al. [36] developed an NLP-system, which can recognize eight different kinds of emotions. According to the users intent and emotion, an appropriate response is generated. Chatbot lend themselves to be used as the interface to a knowledge database, as a simple textual conversation is more user friendly than manually searching through a database or a library.

Also chatbots have the advantage of a relativity anonymous interaction, i.e one does not have to talk to a human in order to obtain the desired information. Crutzen et al. developed a chatbot that is specialised in answering questions related to sex, drugs, and alcohol asked by adolescents. In the evaluation the authors wanted to find out how well chatbots are suited as informational source on such sensitive topics. They compared the results to other sources of information like telephone lines and search engines and got positives results in terms of the duration of the interaction [37]. Another prototype and consecutive study was conducted with the *Pharmabot: A Paediatric Generic Medicine Consultant Chatbot*. It is a chatbot designed to prescribe, suggest and give information on generic medicines for children. It achieved good acceptance rates in regards to important factors such as user-friendliness, appropriateness of answer and speed of response [38]. The Taiwan Center for Disease Control (CDC) has a chatbot named *Disease Control Butler* (@taiwancdc), which can provide “updates on disease outbreaks and public health threats of over 200 countries” [75] and answer users questions about over 90 different infectious diseases. It also informs on pre-travel vaccines or preventive medical treatments according to the destination the user wants to travel to next. With over 70000 friends on the Line messenger platform, it is one on the most widely spread

public health related chatbots. The chatbot was developed in cooperation with HTC using state-of-the-art AI technology.

It is noticeable that the aforementioned chatbots all operate in fairly restricted domains. They also showed, that chatbots can compete with more traditional sources of information and can even bear significant advantages such as increased accessibility and patient engagement.

2.2.2 Human-Chatbot Interaction

When humans are talking to each other, they cannot separate having a conversation from thinking about whom they are having the conversation with. In messaging apps the interaction with the chatbot happens adjacent to conversations with real humans, as instant messaging platforms are primarily used to communicate with friends and family. Furthermore humans tend to generally attribute human traits to robots (anthropomorphism) [39]. Many studies examine the factors that contribute to a conversational partner being seen as non-human. We can distinguish between studies that investigate the content of the dialogue and studies, which focus on the form, i.e. the appearance of the bot [40]. An important factor to keep in mind when designing the appearance of a chatbot or any CA is the uncanny valley effect. The term coined by Masahiro Mori in 1970 describes the shift in a person's reaction to a humanlike robot from empathy to aversion as it approaches, but fails to attain, a lifelike appearance [41]. This is especially important for embodied agents, which have the ability to also use non-verbal communication cues, whereas disembodied agents, like our Dengue Detector chatbot, are primarily communicating through a text-based interface. The only physical representation is in the profile picture (avatar), but as [42] shows, embodiment is not a precondition for anthropomorphism. Ciechanowski et al. conducted a study to asses the emotional and physiological responses to a humanlike avatar chatbot compared to text chatbot by measuring psychophysiological reactions. They found lesser uncanny valley effects and less negative affect in cooperation with a simpler text chatbot [43]. Humans even tend to attribute the common gender stereotype to CAs depending on the sound of the voice. For example, a male voice is received as a bigger authority on sports related questions, while a female voice is seen as having more knowledge about love and relationships [44]. In general, studies found

that users are more open, more agreeable, more extroverted, more conscientious and self-disclosing, when interacting with humans than with AI [45]. Nevertheless the use of more human-like language and name contribute to anthropomorphism towards the bot [42]. Some rules of human social behaviour, such as reciprocity, can also be applied to the interaction between humans and bots. Humans are more willing the help a bot, which helped them before, then they are helping a bot they have never met before [46]. People even follow social norms when interacting with bots and may show politeness to them. When bots follow human conversational rules and show high reciprocity, people tend to open up more to the bot [47].

On the other hand, conversations with non-human entities differ in a lot of ways from human to human conversations. Hill et al. [48] compare chatlogs of conversations with humans to conversations with the popular chatbot *Cleverbot* to examine the differences in the content and quality of the conversations. Hill et al. found that human–chatbot interactions tend to last longer than human–human interactions between strangers and involve shorter messages, less complicated vocabulary and more profanity. Other properties of human conversation, such as the tone [49] and adaptive of responses [50] also contribute to the perceived humanness of the bot therefore promote the engagement with the CA.

The studies above have shown the influence the appearance of the bot has on its perception. While it generally provokes more positive human behaviour if the bot is anthropomorphized, the uncanny valley has to be taken into consideration to not have the effect flip in the opposite direction. Other studies have shown that by mimicking human behaviour in terms of tone, conversational patterns and so lets users feel more comfortable and perceive the bot as more human-like, which, in turn, increases the chance of the human helping the bot.

2.2.3 Chatbots in Crowdsourcing

There is very little research on the intersection of crowdsourcing and chatbots. We assume this is the case, because crowdsourcing usually requires a more complex user interface, e.g. the collection of volunteered geographic information (VGI) is mostly done through apps that were designed for that purpose [51]. There are a lot of apps and websites in the field of

civic complaints. Those apps are used to report damaged roads, garbage dumps and other problems in a city. *Citicafe* by Atreja et al. is a bot to help citizens to report problems and gather information related to civic issues for different locations and their neighbourhoods. It operates on platforms like Facebook Messenger, Twitter and Slack using the Watson Conversation Service [52]. Another instance of collecting data from the public is *raheem.ai*. It is a chatbot, which lets users file reports on their recent police interactions. The purpose of this is to discover concrete suggestions for improvement. Within the conversation, the time and place of the interaction is recorded [76]. The conversational agents mentioned above have in common, that they guide the user through a series of questions to generate a report, unlike our Dengue Detector chatbot, which is designed to collect images. The examples above indicate that crowdsourcing via chatbots is possible but further, in depth, research is lacking.

2.3 Difference to Related Work

In Section 2.1, two ways to control Dengue proactively are pointed out. Firstly, there is the approach to detect breeding sites via object recognition in existing databases such as GSV or Flickr. This is a scalable process, but is prone to missing current images or images in hard to reach locations. Secondly, other projects use crowdsourcing to find mosquitoes and their breeding sites. Crowdsourcing can enable the mapping of breeding sites in remote locations, but does not scale well, as each breeding site has to be reported manually. The goal of this work is to combine the advantages of both approaches, by designing a system, which has the scalability of automated detection, paired with the high resolution and flexibility of crowdsourcing. To reach a broader audience, we present a chatbot as a novel interface in the domain of crowdsourcing-based apps related to the control of Dengue.

Section 2.2 discusses the historic development of CIs and introduces a variety of existing chatbots in the domain of healthcare. The literature review revealed, that there is very little research at the intersection of chatbots and crowdsourcing. The chatbots presented in Section 2.2.3 use crowdsourcing in domains unrelated to epidemiology or images. We could

not find another chatbot, that is used to collect geotagged images. While there are multiple mobile apps (Android and iOS or webapps) for the distinct purpose of collecting images of mosquito breeding sites, using a conversational agent as the interface appears to be a novel approach. Some of those apps, such as DengueChat, use gamification mechanics similar to those used in the Dengue Detector. There are however chatbots, which deal with Dengue or other vector-borne diseases in the way of delivering information and notifications on the disease.

Chapter 3

Concept

In this chapter, first the most common existing technologies and development approaches are introduced. After describing the architecture of our system, the design of the chatbot as well as all its features are discussed in detail.

3.1 Chatbot Frameworks

In this section, the basic architectural components and common concepts of chatbot development are discussed. According to Stoner et al. [53] the architecture of chatbots consists of three parts, in most cases: They distinguish between The *Responder*, *Classifier* and *Graphmaster* (Figure 3.1) . Braun et al. refer to the components *Request Interpretation*, *Response Retrieval* and *Message Generation*, but the underlying idea is similar [25]. The Responder is the interface between the user input and the business logic of the system. The Classifier takes apart the user's natural sentences and divides them into logical components. The Graphmaster does the pattern-matching to decide which response is appropriate.

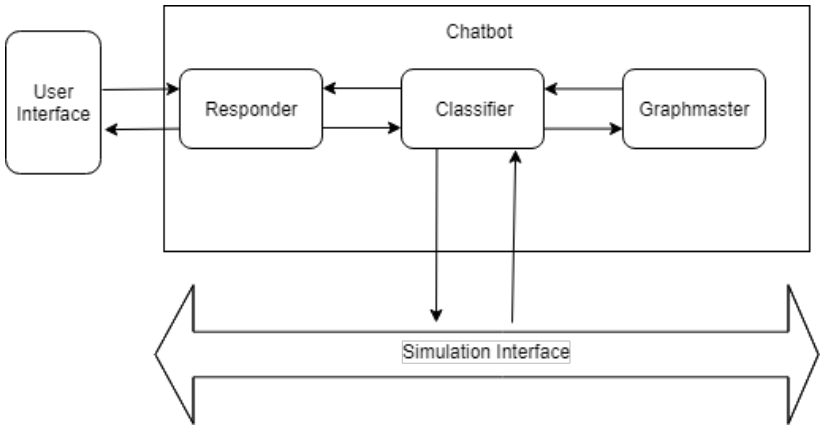


Fig. 3.1 Overview of generic chatbot components adapted from Stoner et al. [53]

In the first step, the users natural sentence needs to be processed. For this task NLP or NLU tools can be employed. Most commonly, NLU systems are based on the Intent-Entity-(Context) paradigm [54]. Following this concept, an intent represents the purpose of a user's input. This can be any operation, action or request for information. An Entity is any term or object that is mentioned in connection to the user's intent, e.g. a location, person, time, organization and so on. The context is the topic of the current conversation [55]. It is apparent, that the bot needs all three components to be able to respond to complicated requests.

3.1.1 Comparison of Frameworks

Chatbots have moved into the focus of the technology industry only in recent years, yet there is an abundance of frameworks and APIs for the development of Conversational Agents. The available frameworks lower the development effort immensely by taking on some of the more complex aspects like machine learning, NLP and integration with the messaging platforms.

The four major tech companies, i.e. Google, Facebook, Microsoft and Amazon are all represented in the space of chatbots and NLP frameworks, as some of the most popular platforms such as *Dialogflow*, *Wit.ai*, *LUIS* and *LEX NLU* are owned by those companies respectively.

The selection of the bot framework is highly dependent on the architecture and vice versa as the different frameworks imply different approaches to development. As per usual in software development nowadays, it is not only a decision for one framework but for an entire stack. We were looking for a framework and system architecture, which is best suited for the purpose of collecting geotagged images. For our comparison we will distinguish between three main kinds of chatbot frameworks.

- **All-in-One** services like Wit.ai and Dialogflow, that integrate with the Facebook API, handle the NLP and run on an existing server.
- **Natural Language Understanding** Services like *Rasa NLU*, *Amazon Lex* or *LUIS*, which specialize on NLP.
- **Do-It-Yourself (DIY) chatbot builders**, like *ManyChat* or *Chatfuel* are all-in-one solutions, that cover all aspects necessary to create a chatbot. They often feature a visual interface and require little or even no coding, which makes them accessible for a wider user base, but are often very limited when it comes to advanced applications [77].

In the following comparison we will not consider frameworks from the aforementioned DIY-category, as they do not include the appropriate interfaces to deal with media and locations in the more advanced ways that we need to. Also frameworks that focus solely on NLP or NLU like the Natural Language Toolkit for Python are not considered, as the NLP is not the focus of this work. Therefore we focus on the more powerful and fully featured APIs. We assess the 5 remaining frameworks on the basis of 7 different criteria (Figure 3.2).

Framework	Platform integration	Diagnostic tools	Community support	Commercial	Accuracy of intent recognition	Usability
Dialogflow	✔	(✔)	✔	Free	(✔)	✔
Wit.ai	✔	(✔)	✔	Free	-	(✔)
Ibm Watson	(✔)	✔	(✔)	Payment according to traffic	✔	(✔)
Microsoft LUIS / Azure Bot Service	(✔)	✔	(✔)	Payment according to traffic	✔	(✔)
RASA	(✔)	(✔)	(✔)	Free (Open Source)	✔	(✔)

Fig. 3.2 A comparison of chatbot frameworks along six dimensions

RASA is a NLP framework at its core and does not come with easy messenger integration out of the box. The commercial product Microsoft LUIS requires the paid Azure web service. Similarly IBM Watson requires the non-free Bluemix cloud service. The community support for the free and/or open source projects is generally better, as users rely more on other users than on the service provider for help. Despite being open source, RASA still does not have a community as vibrant as some of the other frameworks [78]. The intent recognition rating is based on research conducted by Braun et al. [25], who evaluated the proficiency of different online NLU Services. In this comparison, Microsoft LUIS generally performed best, with RASA and Watson behind it. API.ai had the worst results in this test. Wit.ai was given a slightly lower usability rating as it was hard to configure the integration. Also it does not offer slot filling functionality to easily query the user for required information [79].

All in all, RASA, Microsoft LUIS and IBM Watson are either too complex, lack important features or will involve considerable cost in productive use. We found Dialogflow to be easier to set up and use than Wit.ai, as the user interface and NLP-concepts were clearer and more intuitive from the start. Additionally features like the follow-up intents and slot filling make for a comprehensive, easy-to-use package [56].

We decided to use Dialogflow for the NLP and most basic requests. For more complicated messages, e.g. messages that involve locations and images, the Python helper is called.

The system consists of four main components.

- **The Facebook Messenger API** functions as the direct interface to the user.
- **Dialogflow** is the intermediate layer for handling easy requests and small talk, as well as the basic NLP and intent recognition-
- **Our Python backend** handles the more difficult requests involving locations and pictures.
- **The server side** to send locations and receive images.

Our Python backend is connected to the server side via a REST API. It interact with the server in three ways: It can receive a user's location and return a list of locations with old or missing images, receive and store pictures sent by the user and provide the number of Dengue cases by (sub-)district.

3.1.2 Dialogflow

To process the user input, Dialogflow goes through two major steps: intent classification and entity recognition [57]. intents can easily be created using Dialogflow's web interface (Figure 3.3).

Fig. 3.3 Overview of intents in Dialogflow web interface

Every intent can be triggered by either an event or by matching (a combination) of words to an intent. Dialogflow uses machine learning to not only associate the pre-defined phrases with the intent but also different phrases, which are similar in meaning. It also accounts

for typos with automatic spell correction. To improve the intent recognition, the interaction history can be monitored to detect false positives and fix errors in the intent matching. In Dialogflow we can define custom entities. In the case of our chatbot, we define the “Breeding site” entity (Figure 3.4).

Fig. 3.4 Entity definition in Dialogflow web interface

Figure 3.5 shows how keywords are matched to an entity. The matching is highlighted by Dialogflow automatically, but can be changed manually to correct errors. In this example, the user input “old tire”, will be recognized as an input of the entity type “breeding_site”.

Fig. 3.5 Training phrases for an intent in Dialogflow web interface

Also influential on the intent recognition is the current context of the conversation. Every intent can have an input and output context. Each context lasts for a defined period of time (by default for 5 interactions or 20 minutes). The lifespan of the context can be modified by the developer. This is sometimes necessary to prevent mismatching of intents (Figure 3.6).

Dialogflow also has the notion of follow-up intents, which is a user-friendly way to connect two intents via contexts, i.e. the follow-up intent's input context is the same as the parent intent's output context.

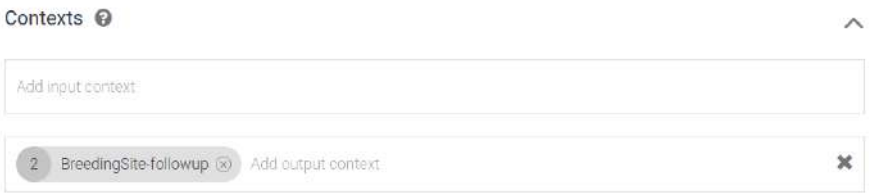


Fig. 3.6 Input and output contexts for an intent in Dialogflow web interface

In the Python backend, we can retrieve the breeding sites mentioned by the user from the Dialogflow response. Every query and response is in the JavaScript Object Notation (JSON) format. This will be discussed in more detail in Section 4.4.1. To generate a response, Dialogflow either uses the pre-defined phrases or queries an external server for a response. In this case, Dialogflow randomly selects one of the variants set by the developer (Figure 3.7). The variation in the answers is intended to let the conversation seem more natural.

Fig. 3.7 Responses to an intent in Dialogflow web interface

3.2 Dengue Detector

3.2.1 Design

This section describes the major considerations in the design process and the resulting characteristics of the bot. As talked about in Section 2.2.2 of the related work chapter, anthropomorphism is generally a desirable effect. Therefore, the fact, that the user is talking to a bot and not a human should not be hidden. It should be made clear from the first message on, that users are talking to a machine. Platforms like *Line* even require the name of the bot to contain the substring "bot" to avoid confusion. The look of the bot is modelled after the Siamese fireback, which is dressed as a detective (see Figure 3.8). It is the national bird of Thailand and eats mosquito larvae.

Fig. 3.8 The Dengue Detector avatar (left), a real life Siamese Fireback bird (right) [80]

When designing the conversational flows, the target platform (in our case, Facebook Messenger) and its traits and constraints have to be taken into consideration. Each platform offers different UI elements such as media (images, audio, video), carousels (a horizontal list of tiles, which can contain images and hyperlinks) and quick replies (a set of predefined answers to a question), which can be used to enhance the conversation [81]. Also the bot's purpose should be apparent from the start. We made sure of this, by sending an introductory message after the user clicks the *get-started button*. The user will only see a longer message

upon the first interaction. Another important aspect is the tone of voice. To find an appropriate tone, one has to keep in mind the target audience as well as the domain of the bot. The bots responses contain more polite phrases such as “I would like to”, which amount to a friendly tone. Phrases like “Do it now!” indicate a more rough tone and are therefore omitted. When designing the dialogue there are several thing to keep in mind. It is important to let users know that their requests are being worked on. All common instant messaging services such as Facebook Messenger indicate the state of the processing of the message. During conversation, the bot should exhibit human-like behaviour, i.e. use natural sounding phrases, maintain continuity and deal with unexpected input and ambiguities. To achieve this, the bot should always keep the conversation going and do not lead users into a dead-end. Every additional feature of the Dengue Detector eventually leads the user back to the main intent of the chatbot, which is to take geotagged images of breeding sites, e.g. when the user has learned about Dengue, the bot asks him to participate afterwards. While keeping the conversation going, the bot should always offer a way to abort the current dialogue branch to avoid frustration. Misunderstandings can always happen, as NLP is still a very flawed process and it is hard for designers to account for every possible user input. These situations should be dealt with in a non-frustrating way, e.g. by abstaining from repetitive answers and expressing error messages in plain enough language [82]. Especially in cases of multiple errors in a row, the chatbot needs to provide the user options to pick the conversation back up.

3.2.2 Features

The features of the conversational agent can be subdivided into seven use cases. Each of the dialogue options is implemented as an intent in Dialogflow. Besides the core feature of taking pictures, it also contains the gamification components *leaderboard* and *challenges* as well as *user statistics*. The user can also get added value from querying how many Dengue Cases were recorded in the nearby area. Additionally, the bot provides information on the Dengue disease, its vector and on the *Dengue Detector* project itself. All of the functions mentioned above can be accessed through the bots *main menu* (Figure 3.9). The Notifications-feature naturally cannot be activated by talking to the bot, but is rather triggered by events not under the control of the user, i.e. the server sends notifications in regular intervals depending on the duration of the users inactivity.

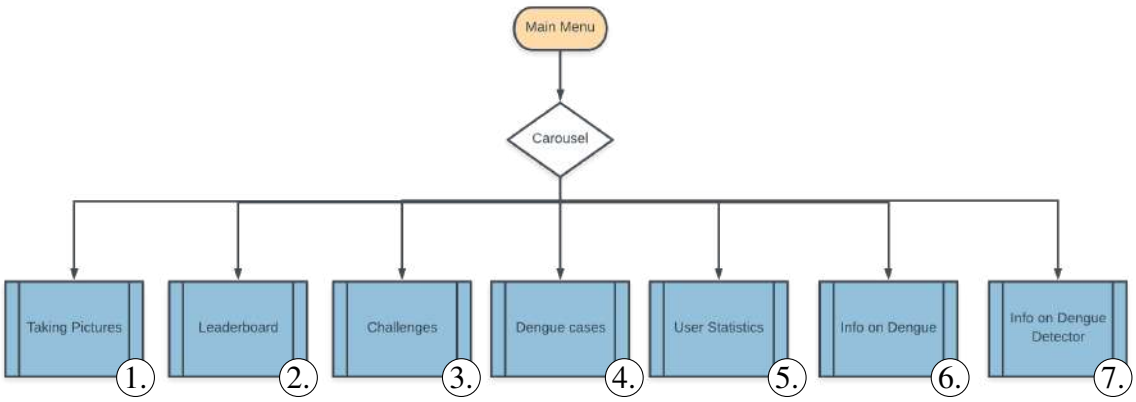


Fig. 3.9 Main menu items of the Dengue Detector chatbot

The orange ellipsis represent the input by the user, the blue boxes are the bot's answers. The branches represent the possible inputs by the user. Usually, quick replies are used to guide the user in the intended direction. Quick replies are a way to display the possible answers to a question in the form of buttons. The user can click on the button to reply with the predefined answer. The options are listed on the edges connecting the diamond and the succeeding responses.

3.2.2.1 Contributing Pictures

This part covers the basic intent of sending the location where to take the pictures to the user, taking the pictures and geotagging them (see Figure 3.10).

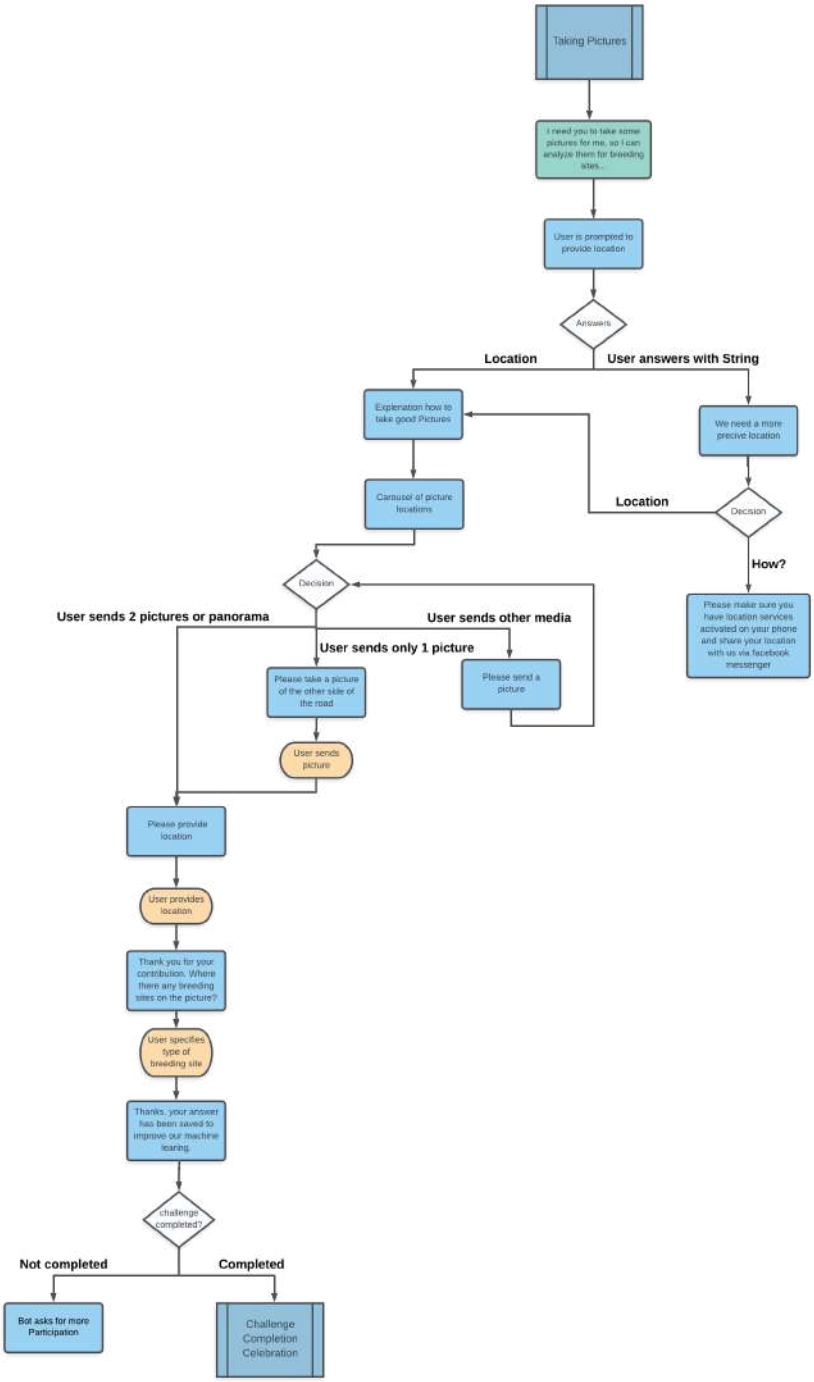


Fig. 3.10 Flowchart of contributing pictures-intent

After agreeing to participate, a brief explanation of the workflow is given. Then the user is asked to share his location. The bot needs to receive the location of the user via the *share location* feature of the Messenger app, as Facebook does not allow bots to track the users with the "live-location" feature. That means, it is not possible, unlike in human-to-human chats, to share the location continuously for a limited amount of time (usually 60 minutes). A precise location is needed, because the system will send the nearest spots, where images are missing based on the radius around the users current location. In case the user does not use the quick reply and answers with a textual input (e.g "I am in Bangkok" or "I am in Thailand"), the system recognizes this and prompts the user once more to send a precise location.

When the user's location is known, the chatbot presents instructions on how to take a photo. This is only explained in full detail in the first interaction and will be summarized in later interactions. The user is then shown a carousel of geolocations (see Figure 3.11). By clicking on the image, the Google Maps app opens and navigation to the location can be started.

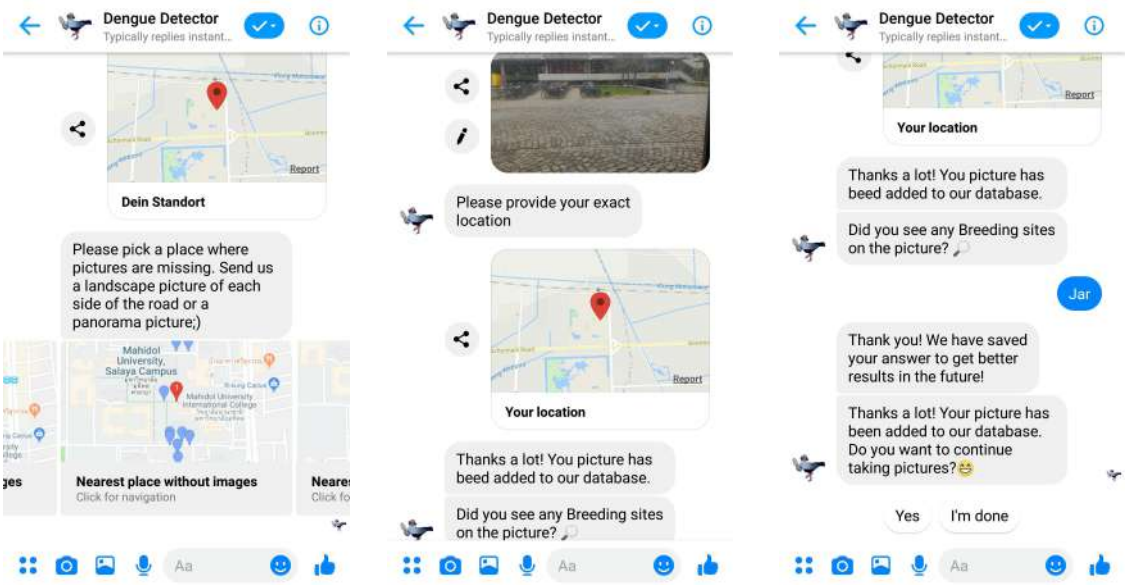


Fig. 3.11 Carousel of missing spots near the user's location (left), user sending a picture (central), user geotagging the picture (right)

When the user is at the location, the picture can be taken through the Messenger app directly or using the native camera application of the device. In the latter case, the picture needs to be added via the gallery. Upon receiving a picture, it is first checked if the received "Facebook_Media" is actually a picture, as Dialogflow only generically recognizes media, but not of which type it is. Our system checks the media for .jpg and .png file extensions. If the media was not a picture, the bot rejects it and tells the user to send a picture. As explained to the user, we expect to receive two pictures or a panorama picture. This is to ensure, that both sides of the road are covered. To take a Panorama picture, the user would be required to take the extra steps of switching to the native camera app on his phone and upload the picture to Facebook Messenger through the gallery. When the user sends only a single picture, that is not a panorama picture, the chatbot requests another one. When all the necessary pictures have been taken, the bot asks the user to share his location again to geotag the picture precisely. If the user carries it out as expected, the chatbot proceeds by asking if the user saw any particular type of breeding site on the picture. In case the user has finished the current challenge through this contribution, the challenge completion celebration (see Section 3.2.2.3) is triggered. Finally, the bot kindly asks if the user wants to continue taking pictures.

3.2.2.2 Leaderboard

The leaderboard shows the ten most active crowdsourcing participants. It highlights the best placed user and the current user, who is viewing the board. The chatbot also has some motivational words depending on the performance of the user. The system offers users the option to not be displayed on the leaderboard. After selecting that dialogue option, the next time the user calls the leaderboard, his name will be left off of the list. The quick replies also include the option to set a synonym by which they will be refereed to on the leaderboard and anywhere else in the conversation (Figure 3.12). For this feature, the lifespan of the context is one, to prevent consecutive commands to be interpreted as the users preferred synonym.

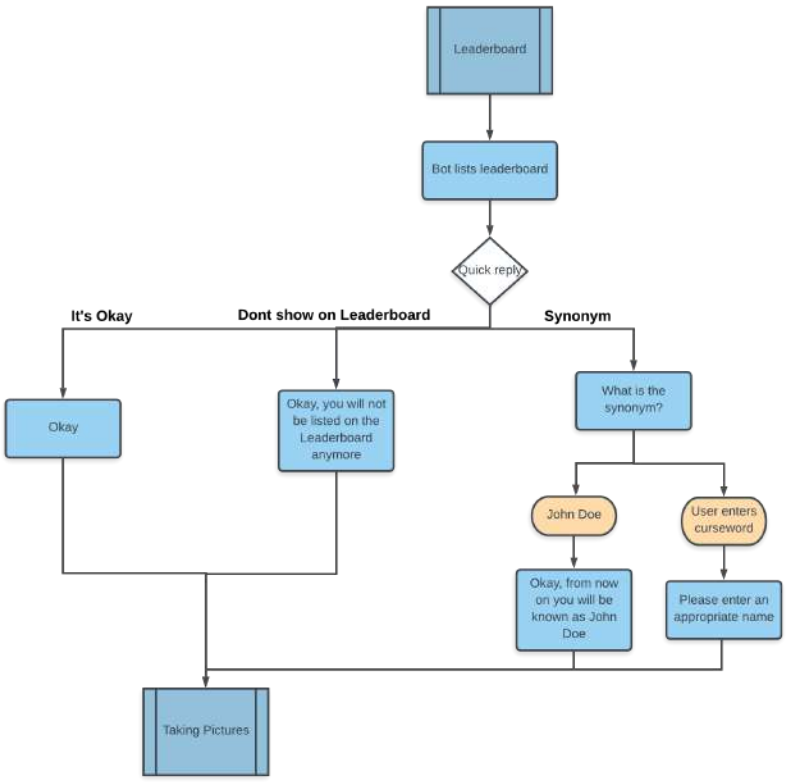


Fig. 3.12 Flowchart of leaderboard Intent

3.2.2.3 Challenges

The user can ask the bot “What are the current challenges?” or click on the corresponding item in the main menu. The bot lists all of the currently running challenges and the users standings in the challenge. Afterwards the bot asks the user if he wants to get started (Figure 3.13). A positive answer leads into the Contributing Pictures intent. When asked, the bot also gives a brief explanation of what a challenge is and how it works. Every time the user takes a picture within the boundaries of the challenge, i.e. in the defined timeframe and location, the system tracks the user’s progress automatically in the background.

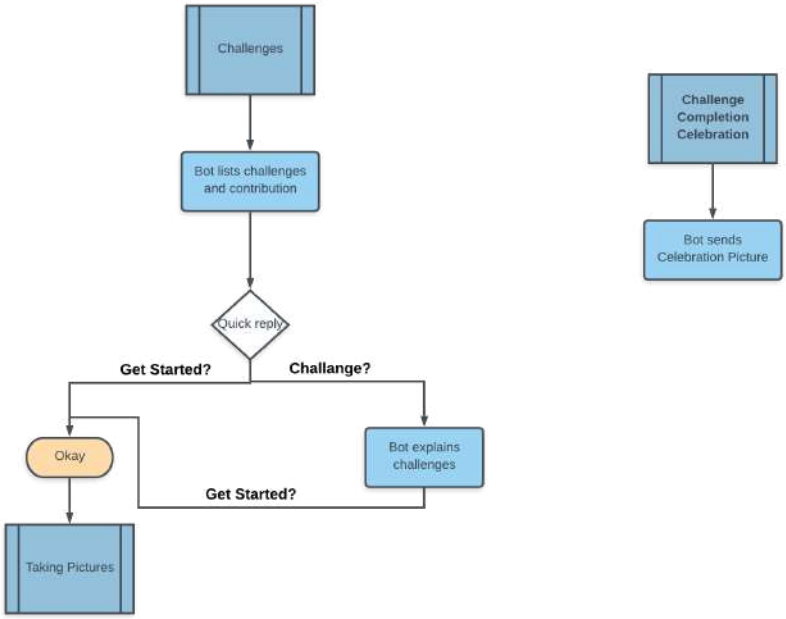


Fig. 3.13 Flowchart of challenges-intent

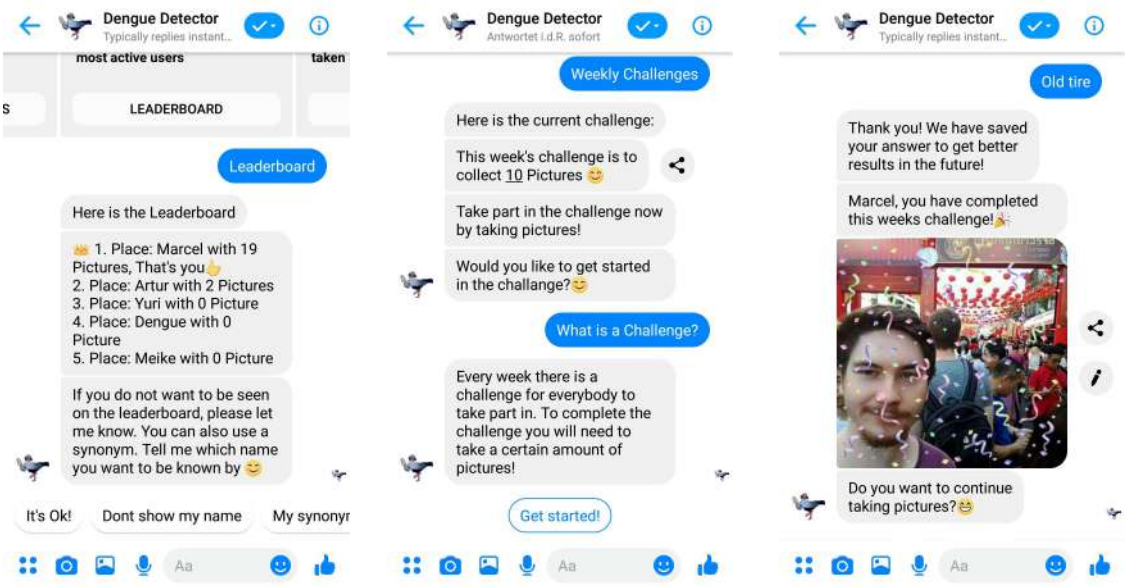


Fig. 3.14 Leaderboard of most active crowdsourcing participants (left), the current challenges (center) and the celebration after the completion of a challenge (right)

After completion of a weekly challenge, the user's work is appreciated with a celebration. The profile picture is augmented with confetti. The Python library Pillow [83] is used to download and edit the original profile picture. A creative common picture of falling confetti with a transparent background in .png format is also downloaded. The confetti picture and the original profile picture are scaled to 400x400 pixels, as the standard Facebook profile picture is square. Finally, the confetti-picture with transparent background is put on top of the original picture to create the final result (See Figure 3.14).

3.2.2.4 Number of Dengue Cases in an Area

The chatbot can be queried as to how many cases of Dengue fever were reported to the Ministry of Public Health Thailand in the last year. There are different ways to retain the information. The user can select the Dengue cases function from the main menu or ask “how many Dengue cases were there in my district?” or something similar. In case the question did not contain the location, e.g. “Bangkok”, the user will be asked to share his location via quick reply. Additionally the user can ask for the number of cases by naming the district (see Figure 3.15).

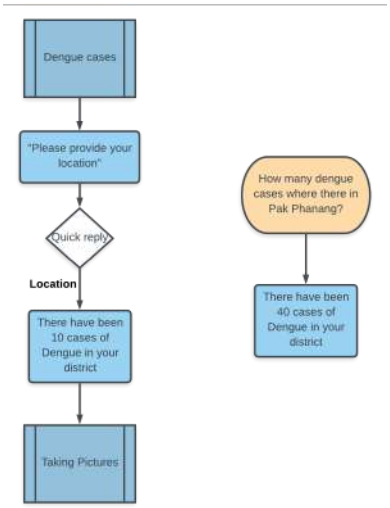


Fig. 3.15 Flowchart for number of Dengue cases in an area via menu (left) or via NL (right)

3.2.2.5 Statistics

Users can retrieve statistics on how many pictures were taken by them individually and by the community as a whole. The system can handle various requests, which can be particularized with different parameters:

- **Subject**: The user can ask about who took the picture. Possible values are himself or about all users as a whole.
- **Timeframe**: The user can query how many pictures were taken in the last day, week, month or year.
- **Location**: The user can ask for a district or subdistrict.

That means, users can enter queries like “How many pictures did I take in Bangkok last week?” and get the appropriate answer. In case the user does not specify all of the parameters, the system falls back to default values. By default, all of the users will be considered as the subject, for the timeframe the last week will be considered as the default value and for location there is no restriction by default. The default values have been chosen arbitrarily based on our assumption of what might be most interesting to the people.

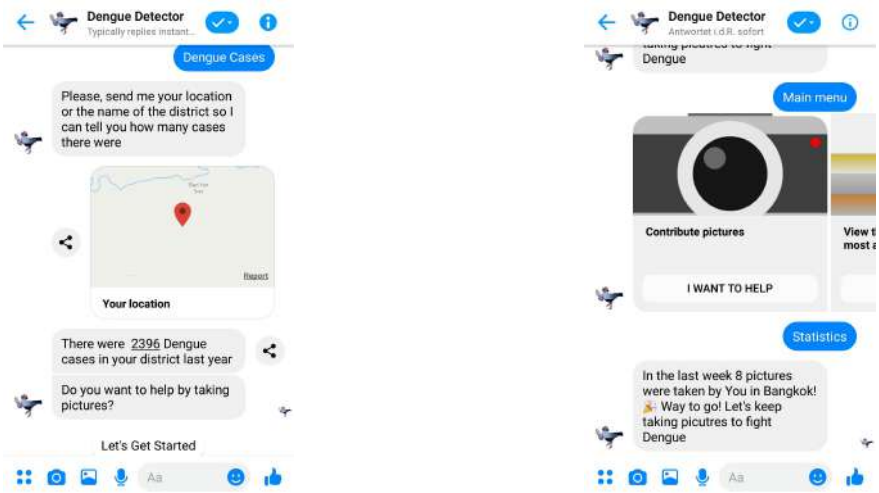


Fig. 3.16 Number of Dengue cases in an area (left) and statistics on pictures taken (right)

3.2.2.6 Information on Dengue

In order to incline users to contribute photos, we educate them on the burden of Dengue to raise awareness of the issue. To increase the chance the user will be willing to participate, the informational texts are personalized according to the users location. By providing information that is more relevant to the user, we hope to achieve higher engagement. Using the openly available information on Dengue, we try to educate the users on the basics of the disease. There is a special emphasis on the mosquito breeding sites, since those are central to this project. The bot can show a carousel in which all the breeding sites the system can recognize are listed. For further information there are links to the websites of organisations like the WHO or the Ministry of Public Health (Figure 3.17). The user can inquire about more information at any point in the conversation, e.g. by requesting “tell me more about Dengue”.

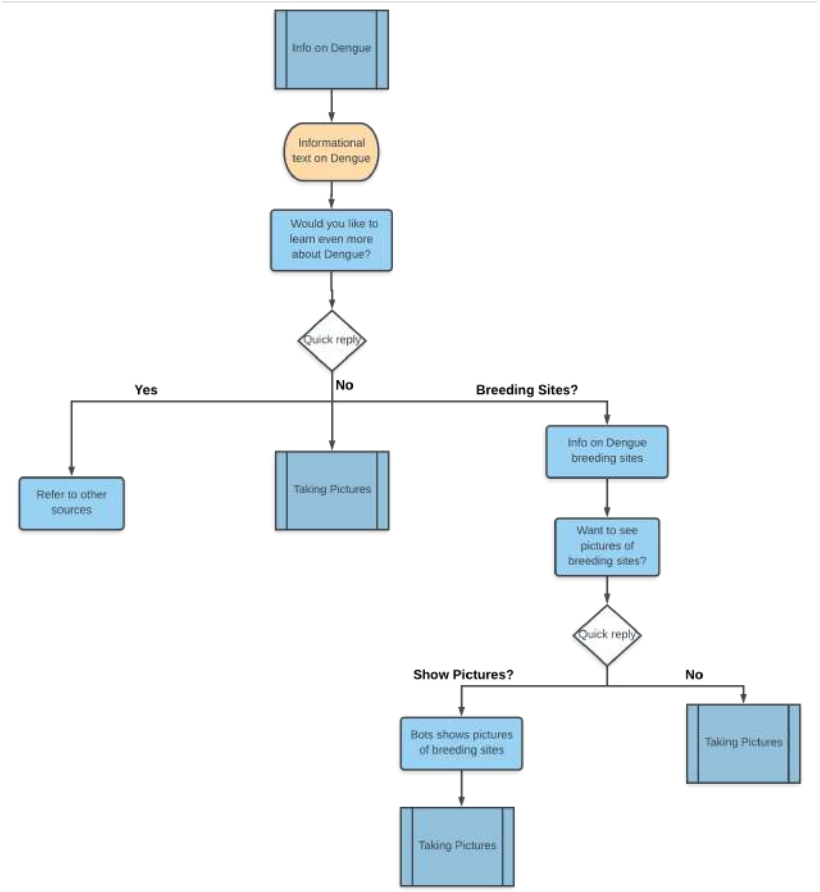


Fig. 3.17 Flowchart of Dengue Information intent

3.2.2.7 Information on Dengue Detector Project

The user can learn more about what the project is, what the goals are and who is involved. After displaying an informational text, the user is prompted to participate.

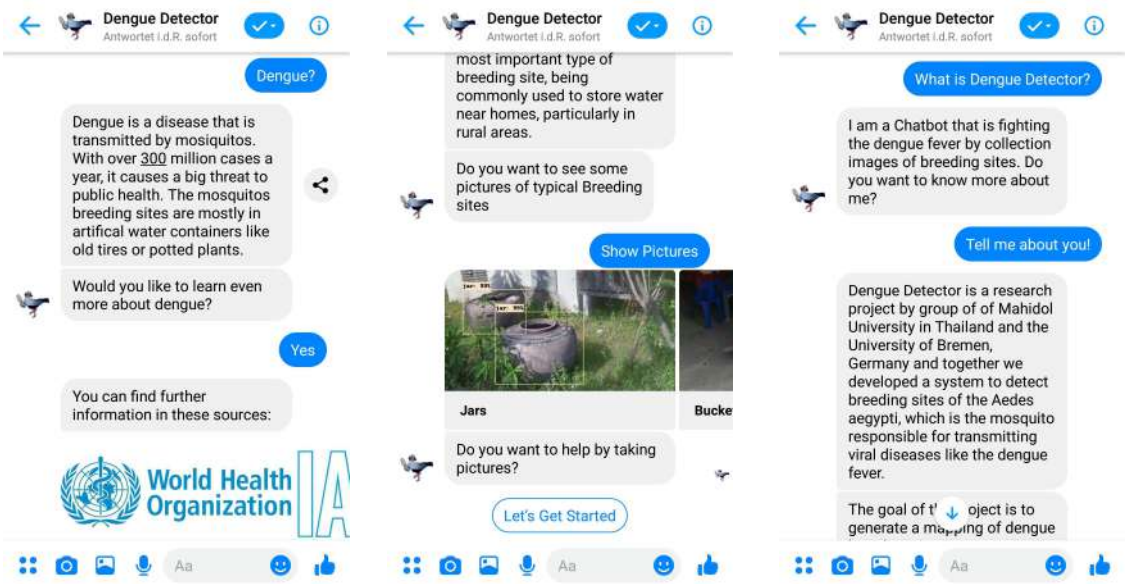


Fig. 3.18 Info on Dengue in general (left), Info on breeding sites(central), Info on the Dengue detector project (right)

3.2.2.8 Notifications

Similar to other forms of communication like Email and SMS, push-notifications are important to keep users engaged to the product. Notifications are used to remind users that the bot is still live by informing about updates, new features or current events. The notifications cause the conversation with the chatbot to rise in the list of chats in Facebook messenger. When people want to chat with their human friends, they will be reminded that the bot exists and it invites them to pick the conversation back up. It is advisable to send relevant, timely, and useful messages without requiring the user to initiate the communication [84]. Not every user is the same. Therefore the messages in a campaign should be targeted to different segments of the audience classified by characteristic such as demographics or previous user behaviour [85]. The Facebook messenger has the policy of the “24-hour messaging window” [86], which means that 24 hours after each user-initiated messenger interaction, the

bot can send a follow-up message. Outside of the 24-hour window, Facebook allows a single additional non-promotional message. Facebook also offers “sponsored messages”, which bypass all limitations and allow companies to sent promotional messages outside of the 24 hour window at a price [87]. As our bot is non-commercial, the *Subscription messaging* feature [88] is more relevant. It allows organisations to send regular content updates to users, who once started a conversation with the bot. In order to be able to send subscription messages, we needed to formally apply to a review. The permission is granted by Facebook on the page level. The application is usually approved after a couple of days (Figure 3.19). More on that process can be found in Section A.1 of the appendix.

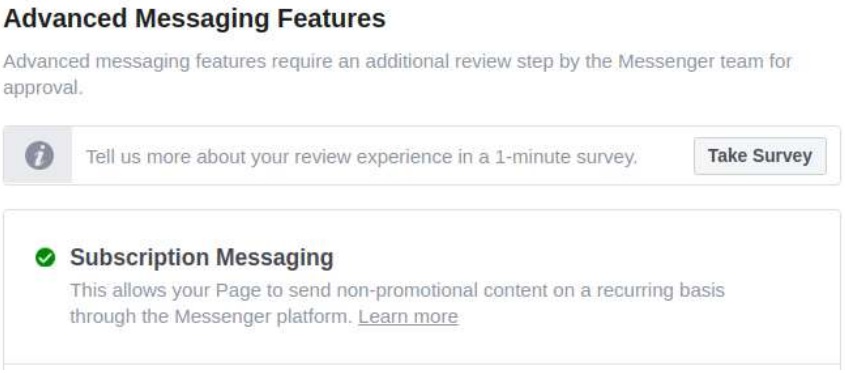


Fig. 3.19 Subscription Messaging activated in the Facebook page settings

We chose to implement different triggers and different notification messages for each trigger. Users are notified one week after inactivity. Different Messages will be sent to the user according to how much information is available, e.g. if the user has already sent a location, we can use it to send location-sensitive messages. If multiple types of messages are eligible, we leave it up to chance which one will be sent to the user. This is a way to make the bot seem more human as it is not sending the same message multiple times.

- "Hi [Username] you have not taken any pictures in [timeframe]. We would be happy if you contributed again!".
- "Hi [Username] we havn’t heard from you in a while. If you want to help again, just text me".

- "Hey, [Username] there were [number of pictures] pictures taken in your area within the last [timeframe]! Keep it up!".
- Hi [Username], We havn't talked in some time. Are you interested in participating in a challenge?

In order to not annoy users we set the *notification_type* to *SILENT_PUSH*. This way, the users will see the notification but will not have any sound or vibration associated with it. To send the messages, we use the Chatfuel API, which simplifies sending push notifications with a GUI (see Figure 3.20). The target users can be filtered by categories such as the overall number of sessions. The trigger for sending the messages can be set to the last/first interaction or depending on any user attribute. In the replies, different message elements can be included, e.g. media and quick replies.

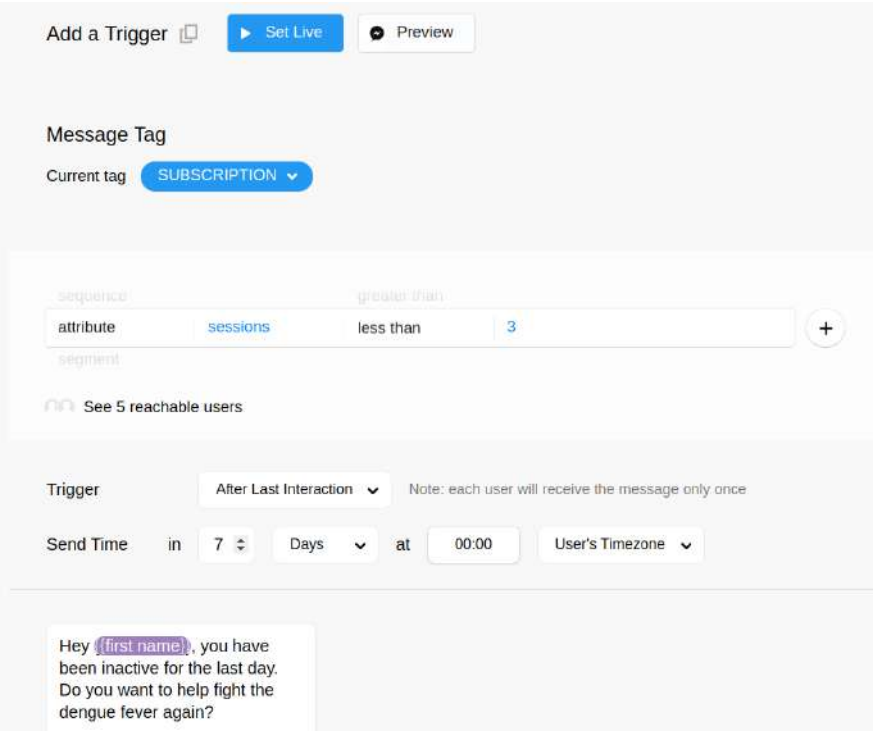


Fig. 3.20 Push notifications triggers, filters and content in the Chatfuel GUI

Chapter 4

Implementation

In this chapter, each component (see Figure 4.1) and their interfaces will be discussed in detail. This encompasses the Facebook Messenger and Dialogflow API integration, as well as the Python backend and Structured Query Language (SQL) database hosted on a Google Cloud Platform (GCP) virtual machine (VM).

4.1 Architecture Overview

Facebook Messenger acts as the main user interface and handles the basic inputs and outputs. The user input is forwarded to Dialogflow, which functions as the NLP engine, i.e. it deconstructs user inputs into logical parts like intents and entities. The more complicated intents, which involve locations, images or the handling of user data, are processed by our Python Backend, which generates answers accordingly and sends them back to Dialogflow's REST API. The leaderboard, challenges and statistics features require a SQL Database to store all the relevant user data. The Python backend also interacts with the server side, where the missing spots are calculated and the pictures are stored via a REST API. Our Python code is hosted on a Google Cloud Platform VM. The SQL Database is running locally on the same Google Cloud VM. This ensures quick access to the data and shields the database from external adversaries. The Python backend communicates securely with the Dialogflow API using an SSH-Tunneling.

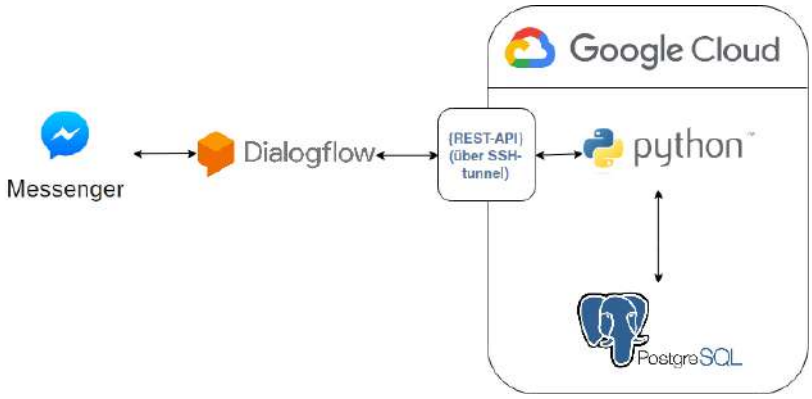


Fig. 4.1 Architecture overview

4.2 Facebook Messenger

The Facebook Messenger serves as the main interface to the user. It is the most popular platform for companies and individuals to host their chatbots, as the user base of the Facebook social network is sizeable in a lot of parts of the world. The Facebook Messenger APIs are very versatile. With the appropriate permissions, chatbots can interact with profiles, pages, groups, posts and so on. The User Interface offers more components than plain text. All kinds of media, carousels, buttons and quick replies can be integrated to improve the user experience. Facebook removes all the metadata from images, which means that every time a user sends a picture, we have to request the location to be shared manually to add the geotag to the image.

To host a chatbot on Facebook, first a facebook page has to be created and secondly a Facebook app. To create a Facebook app, registration on the Facebook Developer Console (developers.facebook.com) is necessary. In the console a *Page Access Token* is generated. Before the app can go online, a data privacy policy needs to be added. The webpage containing the privacy policy declaration is hosted on our VM. We used the privacy policy bot [89] and added the specific ways we use the data to the policy. We consulted a lawyer to assure we are in compliance with the privacy protection laws of the country, where the chatbot is hosted and used.

Through the Facebook Graph API we can retrieve information on the user. By default, attributes such as first and last name can be queried. For other types of data such as the user location according to their profile, a separate permission is required. This can be obtained by submitting the app for review. In that process the applicant has to make clear for what purpose the permission is required.

Before the chatbot can answer users, it has to undergo a review process. To get access to the *pages_messaging* feature, which allows the bot to send and receive messages using a Facebook Page, the submission form (see Figure A.1) has to be filled out. The review process is designed to ensure that the chatbot is responding in a timely manner and adheres to the Facebook community guidelines. More on this process can be found in Section A.2 of the appendix.

4.3 Dialogflow

In the Dialogflow Console a new agent can be created and named. To establish the connection to Facebook, the previously generated Page Access Token for the Facebook Messenger app is entered in the integrations tab of the web interface. To illustrate the flow of information through the Dialogflow system we follow the example of receiving a location. Upon receiving a location through the Facebook integration, Dialogflow will recognize this a a platform event, which is Dialogflow's way to handle non-text user actions. As the event *Facebook_Location* is set as a trigger for the *LocationReceived* intent, Dialogflow will handle the corresponding intent. Fulfilment is activated for this intent in Dialogflow (Figure 4.2). This means, Dialogflow will forward the request to the REST API of our Python backend. The URL at which the Fulfilment webhook can be reached, was previously set (Figure 4.3).

Fig. 4.2 Activated fulfilment for an intent in the Dialogflow developer console

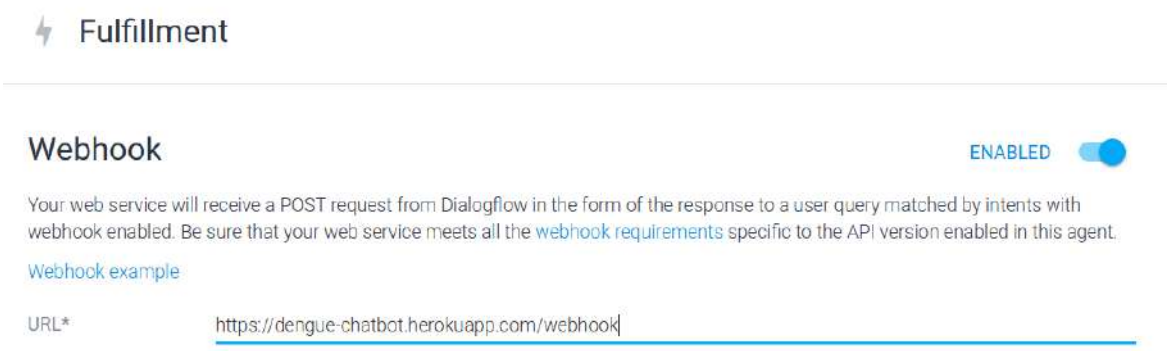


Fig. 4.3 Fulfilment webhook configuration in the Dialogflow developer console

Our Python application called by the webhook is able to recognize the action (see Listing 4.1) and trigger the appropriate function. In this instance, the missing spots around the location or a previously sent picture will be geotagged with those coordinates. In order to distinguish those two cases, we refer the contexts of the conversation.

```
1 if req.get("result").get("action") == "locationReceived":
```

Listing 4.1 Recognizing the action in Python

4.4 Python Backend

The Python backend contains the computational logic for the chatbot and is the intermediate component between Dialogflow and the database. The script is written in Python version 3.7. The Python backend is based on the *Flask* web microframework, which is a comprehensive collection of tools to host a webserver using Python [90]. The backend is designed to handle all the tasks that could not be fulfilled by Dialogflow, i.e. that require more complex processing.

4.4.1 Interaction with Dialogflow

Every time a JSON is sent to the endpoint */apiai*, the method *apiai_response* is called. Inside the method the received JSON is processed by the call to *request.get_json*. The function

processRequest is central in the Python script, as it parses the incoming JSON request and generates the response, see Listing 4.2.

```
@app.route('/apiai', methods=['POST'])
def apiai_response():
        req = request.get_json(silent=True, force=True)
        res = processRequest(req)
```

Listing 4.2 Handling of webhook using flask

The response sent back to Dialogflow has to comply with the specifications of the Facebook API. Dialogflow will then forward the JSON as a custom payload to the Facebook Messenger API. For example, the carousel of missing spots (shown in Figure 3.11) is implemented as a Facebook Messenger *generic template*. The code below illustrates the structure of the JSON.

```
"payload":{
 "facebook":{
  "attachment": {
   "type": "template",
    "payload": {
     "template_type":
       "generic",
     "elements":
       listOfCards[:10]
   }
  }
 }
}
```

Listing 4.3 JSON of Facebook generic template

```
{
  "title":
    'Nearest place without
       images',
  "image_url":
     imageLink,
  "subtitle":
    "Click for navigation",
  "default_action": {
    "type": "web_url",
    "url": link,
  }
}
```

Listing 4.4 JSON of a single card in the carousel

The *listOfCards* mentioned in Figure 4.3 consists of 10 cards, as this is the limit set by Facebook for generic templates. In Figure 4.4 the structure of a card is shown. It contains the a static Google Maps image as a thumbnail and the link to navigate to the location using the Google Maps app as a *default action*. As shown in Listing 4.5, the generated response is encoded as a JSON and is send to Dialogflow.

```
res = json.dumps(res, indent=4)
r = make_response(res)
r.headers['Content-Type'] = 'application/json'
return r
```

Listing 4.5 Follow-up of 4.2: Encoding of response JSON

Dialogflow has a five second timeout limit on request fulfilment, which means that the Python backend server and the tunnelling have to be quick enough to deliver the response to Dialogflow in time.

4.4.2 Interaction with Server Side

The Dengue Detector server side fulfils three main purposes. Each service is implemented as an REST API endpoint. In order to add further capabilities to the chatbot, some development was done in the scope of this thesis. The changes are described in detail in Section 4.7.

1. Getting Spots in Range.
2. Sending pictures.
3. Getting numbers of Dengue cases for a district.

4.4.2.1 Spots in Range

To guide users to take the missing or out of date pictures, geo-coordinates are needed to display directions to those spots to the user. During the project two ways of finding spots where GSV images are missing or outdated were implemented. The first implementation uses the LineString format, the second is based on structuring the area as a grid. The First approach uses the LineString structure provided by the GSV API, in which each LineString represents a road. The user is guided along the road until all pictures are taken. The LineStrings with an image attached to each are then sent back to the server. The second approach is to divide the GSV images in a grid and look for existing images in each area of the grid. Each spot that does not have an image and that is close to other existing images, is sent to the server. This is done to ensure that only useful pictures are taken, as the Street View car drove along roads

and we assume that most breeding sites are close to houses and therefore roads. Additionally in places of low population density the risk of a Dengue outbreak is smaller. To receive the coordinates, a HTTP POST request with a GeoJSON in the format below has to be sent to the server. The JSON contains the *coordinates*, i.e. latitude and longitude, as well as the radius in meters, in which the spots should be situated (see Figure 4.6). The response contains the coordinates of missing sports formatted in GeoJSON. Both approaches are implemented on the sever side and can be used by sending the same JSON request to different endpoints. The LineString implementation is reachable by the route */dengue/get/jobs/*, while the Grid-method is used at the route */dengue/get/missing-boxes/*. For our evaluation we opted to use the Gird-based approach.

```
{
  type: "Feature",
  geometry: {
    type: "Point",
    coordinates: [lng, lat],
  },
  properties: {
    radius: r,
  },
}
```

Listing 4.6 JSON to retrieve spots from server

4.4.2.2 Sending Images to the Server

After the pictures were taken, the Facebook URLs of the images are sent to the server with an HTTP POST request. This is more efficient and reliable than downloading the pictures in the Python backend and encoding it to send it to the server. The geoJSON format is used again, only this time the coordinates are to be understood as geotags for the appended images (see Figure 4.7). Upon receiving the images the server responds with a `Success` message.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [lat, lng]
    "breeding_sites": list_of_bs
  },
  "properties": {
    "image_urls": listOfUrls
  }
}
```

Listing 4.7 JSON to send image URLs to server

4.4.2.3 Number of Dengue Cases in District

The number of Dengue cases in a district according to the Ministry of Public Health in Thailand can be retrieved via a HTTP GET request to the *dengue/get/cases/* endpoint of the REST API. The request JSON contains the coordinates (Figure 4.8) or the name of the location (Figure 4.9), as well as the organizational level. The level can be set to either "district" or "subdistrict". The server will sent back the number of cases, which can then be displayed to the user.

```
{
"type": "Feature",
"geometry": {
  "type": "Point",
  "coordinates": lat, lng
},
  "properties": {
  "level": l,
}
}
```

Listing 4.8 Dengue cases JSON with geo-coordinates

```
{
"type": "Feature",
"geometry": {
  "type": "Name",
  "name_string": name
},
  "properties": {
  "level": l,
}
}
```

Listing 4.9 Dengue cases JSON with location name

4.5 SQL Database

To store all information related to user data, the gamification component and much more we opted for creating an SQL database. They are the most proven way to store data and sufficient for our requirements. We did not see the need to use a noSQL database, as we expected to only be dealing with structured data. To ensure scalability and reliability we make use of enterprise-level tools.

4.5.1 Database Management System

A (object-relational) Database Management System (DBMS) is used to define, create, maintain and control access to the database [58]. For the Dengue Detector chatbot, we chose *PostgreSQL*, as it is one of the most widely used open source DBMS, that is also used by many professional organisations. With its high degree of customisation reliability and community support it fitted all our needs. The Postgres server is started with the Unix command *sudo service postgresql start*. After starting the service, the Postgres command line interface (CLI) *psql* can be accessed as the default Postgres user with the command *sudo -u postgres psql*. In the Postgres shell, the database can be created by using the command *Create DATABASE dengue_chatbot*.

4.5.2 SQLAlchemy

An Object-Relational Mapper (ORM) is used to interface between the Python code and the database. This means, that Python objects are easily mapped to database tables without the need for any explicit conversions by the developer. The ORM handles all the underlying SQL statements. All of the four basic functions of a database, i.e. Create, Read, Update and Delete (CRUD) are supported. There are several ORMs like *Django ORM*, *SQLObject* and *SQLAlchemy*, that adhere to the *Python Database API Specification* (DBAPI), which means they are used as an interface between Python and a database.

The Postgres Server URI has to be passed to the SQLAlchemy configuration at the beginning of the Python script, with *DB_URL_ONINSTANCE* representing the URI. In our

case it has the value https://127.0.0.1:5432, as the database server is running locally on the port 5432 (Listing 4.10).

```
app.config['DEBUG'] = True
app.config['SQLALCHEMY_DATABASE_URI'] = DB_URL_ONINSTANCE
db = SQLAlchemy(app)
db.init_app(app)
```

Listing 4.10 Database initialisation in Python

On the database object the *create_all* function can be called to create the schema according to the model. Objects of a class, which was defined in the model specifications, can be created just like any other Python object. From the ORM-perspective these newly created objects are in *transient* state. In the case of SQLAlchemy, the command *db.session.add(object)* puts the object at hand into *pending* state, which means they will be saved to the database with the next flush. Complementary, the db.session.merge(object) command, transfers the changes to existing objects on the Python level to the database level. To actually write the changes to the database, the operation *db.session.commit()* has to be called. The objects are now in *persistent*-state [91].

Each time the webhook is called, there is a lookup in the database to check if the current user's data is already present. If it is, each user-attribute retrieved from the Facebook API is compared to the corresponding field in the database and, in case of any changes, it is updated. If not, a new entry in the database is created. Every time the user sends a message, which triggers the webhook, the message is saved as an interaction.

4.5.3 Data Model

In this chapter we briefly describe what data is stored in the database. Figure 4.4 shows the data model as a UML (Unified Modeling Language) class diagram. The User-class contains all the information about the user. This includes the information received through the Facebook graph API [92]. This encompasses the Facebook User-ID, the first and last name, the profile picture URL and timezone. Additionally, we store a synonym, which can be set by the user as well as an invisibility flag for the leaderboard. The Picture-class contains

the Facebook URL that leads to the image, a timestamp, the location (latitude and longitude) where the picture was taken, the breeding site visible on the picture (if any) and a reference to the corresponding interaction. The Challenge-class is connected to the user class via the ChallengeParticipation association class. The challenge class stores all relevant parameters, i.e. the time frame, area, description. The association class stores how many pictures were taken by each user in the context of a particular challenge. In order to know whether users have already seen certain pieces of dialogue, we track each interaction with a timestamp an d the associated action. This also enables us to conduct analysis of the user behaviour.

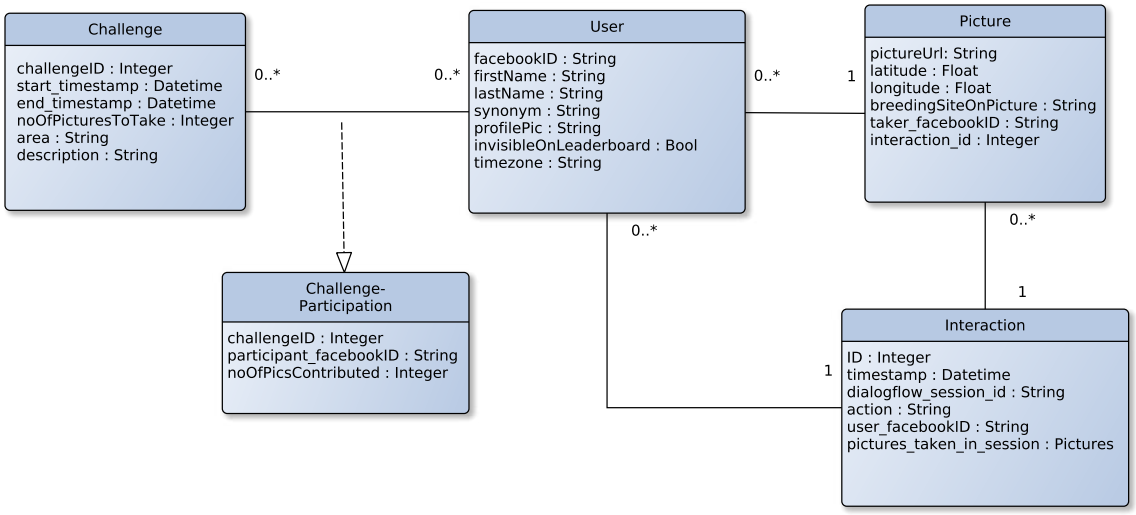


Fig. 4.4 UML class diagram of the data model

4.6 Google Cloud Platform

Google Cloud Platform (GCP) is one of the leading providers to host a great variety of web servers. Among its competitors Amazon Web Services (AWS), Microsoft Azure and others, it offers one of the most comprehensive tool-sets to cost-efficiently host web services of many kinds.

There are two different ways to host the Python code: using the *app engine* or the *compute engine*. The compute engine is a standard Linux (in our case Ubuntu 16.04) VM with practically no limitations. This has the benefit of complete control over all aspects of the server but also the downside of having to deal with every detail manually. The app engine simplifies a lot of the configuration and automatically hosts the web service on the *appspot.com* domain. As the app engine does not support outgoing TCP connections, the connection to our database could not be established. Even using the GCP CloudSQL service, no connection from the app engine to our database server worked. Therefore the compute engine is used to host the Python code, as well as the database.

4.6.1 SSH-Tunneling

As Dialogflow only accepts encrypted webhook connections via HTTPS, we need to expose the flask app to the internet via a tunnelling service. We elected to use burrow.io, which is a easy-to-use tool to forwards a localhost server on a specified port to the web via HTTPS.

After testing several services like ngrok [93] and serveo [94] we settled on using Burrow.io as the SSH-tunnelling service for the Python backend, as it was the most consistently fast and reliable of the tested options. To open the SSH tunnel, the following command (Listing 4.11) has to be entered in the command line:

```
1 curl -Ls https://burrow.io/g76pEWDE-3LZ***** | bash -s
```

Listing 4.11 Command to launch SSH-tunnel

4.6.2 Running the Code

On the GCP VM, the command line tool tmux [95] is used to run multiple terminal sessions at once. In order to reach the Dialogflow API, we need to open up the necessary outgoing firewall port for the Google Cloud VM. To ensure continuous service, a permanent SSH-connection to the GCP VM was necessary, as all the tmux sessions died when the SSH session is closed. The following programs are executed simultaneously in different tmux sessions:

1. Python3 main.py in Dengue-chatbot-backend folder to run the Python backend.
2. Python3 index.py in Dengue_Mobile4D folder to run the server side.
3. Jupyter notebook in Dengue_Mobile4D folder to see the images on. the server. Open http://35.204.179.175:8888/tree/static/uploads in browser.
4. The burrow command to enable SSH tunnelling.
5. The website containing the data privacy policy.

In order to ensure that all of the servers are consistently running, we have set up a separate VM hosted by the faculty of mathematics/computer science of University of Bremen. In this VM, an SSH-connection to the Google Cloud VM is continuously kept alive.

4.7 Extension of the Server

To accompany some of the features implemented in this work, changes on the server side were also made in the scope of this thesis. The three main additions are:

1. Integration of Mahidol Campus missing boxes.
2. Ability to receive the name of breeding site the user saw on a picture. to augment machine learning training data.
3. Query the number of Dengue cases by the name of the district. (previously the query was only by geo-coordinates)

4.7.1 Mahidol Campus Spots

The first task is trivial. The JSON-file was provided by students of Mahidol University and added to the server folder. A reference to the path in the Python code was inserted.

4.7.2 Breeding Site Types

To be able to send the type of breeding site that the user saw on the picture, the REST API is augmented with an additional field *breeding_sites*. On the server-side, code is added to accesses and handle the value of the field while parsing the JSON described in section 4.4.2.2. The data is stored in a Comma Separated Value (CSV) file with two columns. The fist column is the folder name in which the image is stored (which consists of the geo-coordinates the picture was taken at) and the names of the breeding sites the user selected.

4.7.3 Dengue Cases by District Name

In order to accomplish this requirement, it is first necessary to process the natural language input and match it to geo-coordinates. Subsequently the *addresscode* for those coordinates can be retrieved. Lastly the addresscode is used to obtain the number of cases in the specified area. The Google Maps API supports two kinds of conversions. *geocoding* is the process of calculating the coordinates based on a string. The opposite operation, called *reverse geocoding*, returns the readable name for a geo-coordinate. The Google Maps API is not able to geocode the addresscode. Therefore we used the shape files also provided by the Ministry of Public Health to check if a geo-coordinate is in a specified addresscode. Afterwords the string entered by the user is mapped to geo-coordinates by approximation of the centre of the administrative entity. The cases data was provided by the Ministry of Public Health in a CSV format with each row consisting of the addresscode, date and number of cases, e.g. *100101,2014-11-01,1*. The file is then filtered by the addresscode and the number of cases is accumulated.

Chapter 5

Evaluation

The chatbot is evaluated in a real life scenario in Thailand. Through promotion at Mahidol University we aim to motivate students to use the bot and collect geotagged images around the campus. In this chapter, the results of the evaluation are presented and discussed.

5.1 Evaluation at Mahidol University

The evaluation of the chatbot was conducted in collaboration with the faculty of ICT at Mahidol University starting in late march 2019. The results until the end of march will be discussed. The data collection will continue after that. The goal of the user testing was to get a sense of how a chatbot to crowdsource geotagged images is received by the students at Mahidol University. This includes an investigation into usage patterns such as how the participation developed over time and whether the bot is able to keep users interested for a longer time. The study was conducted with students only. Due to the group of participants not being representative and the small size of the area, the results are limited. As mentioned in 4.7.1, the missing GSV picture for the campus of Mahidol University were calculated and included in the chatbot. Students were asked to start a conversation with the bot. In this chapter, the data gathered from the user interactions is described and discussed. There are several datasets from the different architectural layers available to conduct analysis of the chatbot usage (Facebook Analytics for Apps, the Dialogflow API and from our own

SQL database). The dataset of the contributed pictures was exported from the Postgres database as a CSV file. The overall message interaction data is retrieved from Facebook app analytics. The frequency of intents, i.e. how many times the different branches of conversation were reached, is gathered from Dialogflow. All the personal information in the datasets is anonymized. The raw data is stored on the attacked medium (Section B.2 of the appendix). For the plotting of the datasets, the Python libraries pandas and matplotlib were used. To promote the chatbot, a handout was designed (see Section A.4 of the appendix). It contains basic information on Dengue, the Dengue Detector project and the chatbot.

5.2 Results and Discussion

The dataset used for this evaluation spans from March 20 to March 31. Six users were recorded to have used the chatbot in that timeframe. Figure 5.1 illustrates the total amount of contributed pictures over time. As we can see, the growth was fast in the beginning but started to slow down after a few days. Similarly, the number of crowdsourced pictures per day is not rising after the start.

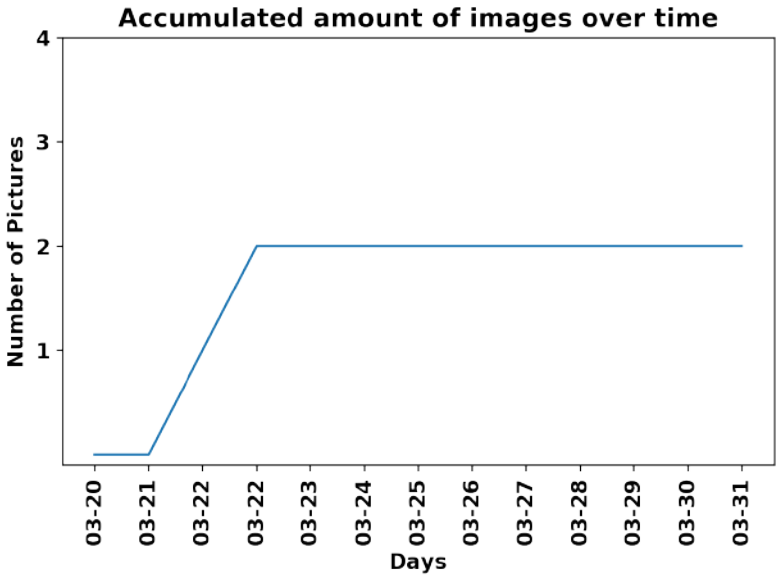


Fig. 5.1 Total amount of pictures

Boakes et al [9] categorized the volunteer engagement into three levels, called “dabbler” (for little participation), “steady” and “enthusiast”. In their study, they found, that around two thirds of participants are dabblers, approximately 30 percent are steady and only a small group of one to five percent are enthusiasts. Figure 5.2 roughly reflects this pattern with only a small proportion of the user base fulfilling the task.

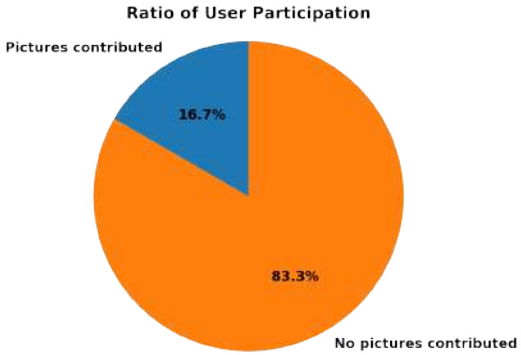


Fig. 5.2 Ratio of picture contribution

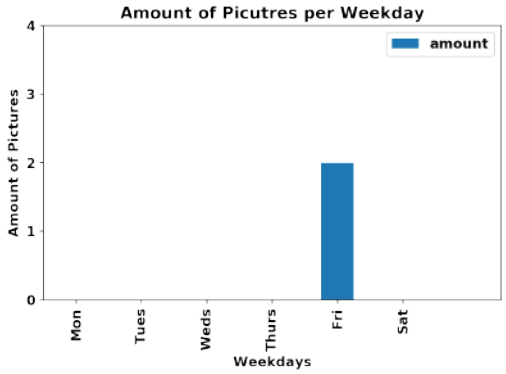


Fig. 5.3 Amount of pictures per weekday

Figure 5.4 shows the total amount of messages of any kind by any user over time. The amount of daily messages decreased over time. This can probably be traced back to the novelty effect discovered in previous research. For instance, the study by Brandtzaeg et al. found that approximately 15% of chatbot users, are motivated by curiosity [59]. The amount of unique users per day supports this (Figure 5.5), as it declines over time as well. The spikes in the interactions can be explained by the notification the users received. Most of the time a notification was sent, the users interacted with the bot again. Those interactions were mostly shorter than the first interaction.

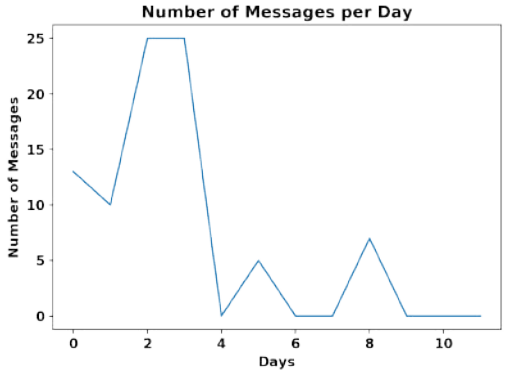


Fig. 5.4 Total amount of daily messages

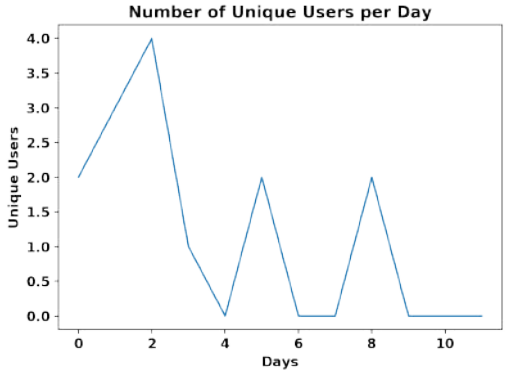


Fig. 5.5 Unique users per day

Some general problems with using chatbots to crowdsource geotagged images surfaced. Users can pick the location inside the Facebook Messenger app freely, as well as upload any picture they want through the gallery of their phone. During the testing phase, this has presumably lead to picture uploads, which do not match the associated location and vice versa. Some pictures sent to the bot were quite clearly wrongly geotagged. The geo-coordinates attached to the pictures in Figure 5.6 pointed to a place on the Mahidol campus, although the pictures do no resemble a place on the university campus.

Fig. 5.6 Two pictures received through the chatbot during the evaluation phase

The chatbot allows a multitude of paths to be taken within a conversation. By investigating how many users took which paths (see Figure 5.7), we can learn about how people use the bot, i.e. which features are popular and which are used seldom. These results are highly influenced by the design of the bot. For instance it is no surprise, that the contribute-intent is the most used besides the welcome-intent, as nearly all other intents lead the user to trigger the intent to contribute pictures.

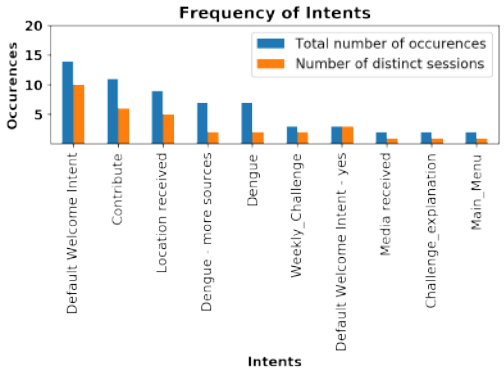


Fig. 5.7 Frequency of intents

As the data shows, there are some features which were never used. The Dengue cases and leaderboard components were never called. This is likely due to their rather hidden links in the main menu. As expected, the welcome dialogue was trigger most often, followed by the central picture contribution workflow and the sent location intent. Slightly less often but still frequently, users selected to receive information on Dengue. This is also unsurprising, as it is a frequently referenced feature, e.g. during the welcome dialogue. There is a big discrepancy between the number of times people sent their location to receive the spots and the amount of times a picture was sent. This suggests that during the picture contribution workflow, the carousel of missing spots was an obstacle. Some users stopped the conversation at that point, presumably because they did not intuitively know how to continue. The process would be much smoother, if there was a quick reply button to trigger the in-app camera, which is not supported by the Facebook Messenger API at the moment. In the current form it might seem ambiguous to some users how to proceed after they received the carousel. Generally, we observed that users tried to follow the main flow of conversation given by the bot. Users deviated from the pre-defined path by sending unexpected messages only very few times. From some of those messages, we also could derive some desired features, e.g. more real time and location based information on Dengue. This evaluation has shown, that contributing geotagged images via a conversational interface is still a novel experience and that further improvement in usability is required to make the interaction with the chatbot more accessible and enjoyable.

Chapter 6

Conclusion and Future Work

In this chapter, the design and implementation of the chatbot and the previous work in this context is summarized. We cover the results of the evaluation and elaborate on the limitations of the work. Possible future developments of the chatbot are discussed.

6.1 Conclusion

In the context of Dengue control and prevention, there are various manual and technological approaches, which either lack in scale or precision. The project "Large Scale Detailed Mapping of Dengue Vector Breeding Site by using Street View Images and Object Recognition" addresses this problem by using GSV images to detect breeding sites. If the breeding site index has a meaningful correlation with the WHO indices, it can be used to better target mosquito control interventions. The GSV image database contains useful historic pictures, but lacks in coverage and recency.

In this thesis a novel way to collect geotagged images to detect mosquito breeding sites was presented. A chatbot on the Facebook Messenger platform was conceptualized, designed and implemented. Previous work on crowdsourcing images of breeding sites are presented and discussed. Apps like Mo-Buzz or Mosquito Alert run natively on smartphones and enable the public to report mosquitoes, breeding sites and cases of Dengue and other vector borne diseases. All of the applications require an additional app to be installed or depend

upon the manual inspection of the images. The Dengue Detector chatbot is addressing those disadvantages by utilizing the Facebook Messenger app, which is already installed on over 60% of devices in Thailand. In addition, the images are analysed for breeding sites automatically by an convolutional neural networks.

We examined the existing research into chatbots in the medical field, chatbots in relation to crowdsourcing and the interaction between humans and CAs. Chatbots focused on improving healthcare such as florance, have integrated gamification elements, e.g. a leaderboard, to track user progress. There are some chatbots designed to collect data from volunteers. For instance raheem.ai guides the user through a questionnaire about interactions with the police. We did not find any chatbots, which aim to collect geotagged images.

To choose the best fitting frameworks for implementation of the Dengue Detector chatbot, the state-of-the-art technological approaches to chatbot development were introduced and evaluated. The Natural-Language-Processing-Framework Dialogflow, Python, Postgres and GCP were selected to implement the chatbot. The bot's main purpose is to collect geotagged images taken by volunteers and send them to the server in order to examine them via object recognition to detect mosquito breeding sites. Besides this, the bot offers a number of additional features. The bot provides information on Dengue, the vector, its breeding sites and the number of reported cases of Dengue in the user's area. It uses gamification to increase user engagement by listing the most active participants on a public leaderboard and presenting challenges.

In order to gain insights into the proficiency of chatbots to crowdsource geotagged images, we conducted an evaluation study at Mahidol University in Thailand. The evaluation has shown, that it is possible to use chatbots to collect get-tagged images in principal, but it also revealed a lot of challenges. The usability of the workflow needs to be improved. The issue of not knowing whether the geotags of the pictures are accurate raises some questions about how to insure the quality of the crowdsourced contributions. Unlike using native apps for corwdsourcing geotagged images, there is no direct access to the GPS location. Also pictures sent via Messenger do not have any metadata, meaning they have to be geotagged in the server side, which leads to problems regarding the validity of the geotags.There are several

approaches to assure quality in crowdsourcing efforts. One option is to let different users do the same task. Secondly, the quality can be controlled manually by an expert. A third way is to use automation [60].

Conversational interfaces are naturally limited in their range of possible input mechanisms. This is why the chatbot has a lot of compromises compared to the native Dengue Detector Android app in order to accommodate the Facebook Messenger platform. This is noticeable in the extra step required to receive the current location and subsequently geotagging the picture. The navigation to the target location also relies on an external app. In the testing phase, this step proved to be a problem for some users, as it was presumably unclear to them how they should proceed. A navigation solution inside the Facebook Messenger app would promote a more frictionless experience. Generally, the picture contribution workflow needs to be more intuitive. Additional limitations are the loss of quality when uploading the pictures to Facebook Messenger and the lack of a panorama camera mode inside the app. Panorama pictures are the preferred format, as they ensure all the surrounding environment of the spot is covered entirely.

6.2 Future Work

There were a lot of ideas discussed but ultimately discarded in the process of this work. Some of them might be worth perusing in the future. An interesting feature from the user perspective would be to get instant feedback on the pictures taken. This way, users know right away if there was a breeding site on the picture, which could lead to better pictures in the future. Some features were considered but ultimately excluded to the constraints of time and resources. The chatbot could provide more information on the Dengue disease, similar to other open data chatbots. The ability to answer more elaborate kinds of questions, e.g. for sick people or to diagnose the disease would be desirable. Also more timely and location-sensitive information, such as notifications on outbreaks or updates on scientific research and other developments in relation to Dengue could be considered. The conversational skills of the bot could also be further improved, i.e. by taking a more AI focused approach or making

more use of the NLU capabilities of Dialogflow besides designing a linear conversational flow. Complementary, the gamification could be extended by a bigger variety of challenges, a complex point system and more. Currently, the spots database only contains locations for a few provinces in Thailand. Also the number of Dengue cases can only be retrieved in one province. To enable more people to participate, those databases could be extended to cover the entire country of Thailand or even expand the project to other countries, where Dengue is endemic.

In the future, testing could be done on a larger scale and with different demographics of users. The test users at Mahidol University are very homogeneously aged and have a similar technological skill level. By observing how people with different demographic characteristics and social environments interact with the bot, we could better examine if the bot is able to engage people from diverse groups. Another aspect worthy of further investigation is whether the quality of the contributed pictures varies depending on the medium, i.e. whether the pictures gathered via chatbot are in accordance with the instructions given (cover both sides of the road, no people on it) or if a native app, which guides the user step-by-step, will produce higher quality pictures. Furthermore, investigation into how successful chatbots are at retaining crowdsourcing motivation should be conducted over a longer period of time with a larger audience to obtain more conclusive data.

The chatbot technologies and the designed workflow presented in this work can be applied to other fields of crowdsourcing to establish a novel channel for participation. For instance, a disaster reporting and alerting system could be based upon the picture contribution workflow. The system could also be used to collect and map data on objects apart from mosquito breeding sites, e.g. to collect images and locations of trash in urban areas or find damaged infrastructure. Overall, this thesis has shown that chatbots have the potential to be suitable for the crowdsourcing of geotagged images but also that a lot more research into this topic is required.

Bibliography

[1] N. Vasilakis and S. C. Weaver, "Chapter 1 the history and evolution of human dengue emergence," vol. 72 of *Advances in Virus Research*, pp. 1 – 76, Academic Press, 2008.

[2] S. Bhatt, P. W. Gething, O. J. Brady, J. P. Messina, A. W. Farlow, C. L. Moyes, J. M. Drake, J. S. Brownstein, A. G. Hoen, O. Sankoh, M. F. Myers, D. B. George, T. Jaenisch, G. R. W. Wint, C. P. Simmons, T. W. Scott, J. J. Farrar, and S. I. Hay, "The global distribution and burden of dengue," *Nature*, vol. 496, pp. 504–507, Apr. 2013.

[3] *Dengue: Guidelines for Diagnosis, Treatment, Prevention and Control: New Edition*. Geneva: World Health Organization, June 2013.

[4] D. M. B. N. Dr Philip McCall, Dr Linda Lloyd, *Dengue: Guidelines for Diagnosis, Treatment, Prevention and Control: New Edition*. Geneva: World Health Organization, June 2013.

[5] L. C. Harrington, T. W. Scott, K. Lerdthusnee, R. C. Coleman, A. Costero, G. G. Clark, J. J. Jones, S. Kitthawee, P. Kittayapong, R. Sithiprasasna, and J. D. Edman, "Dispersal of the dengue vector aedes aegypti within and between rural communities," *Am. J. Trop. Med. Hyg.*, vol. 72, pp. 209–220, Feb. 2005.

[6] A. Y. Chang, M. E. Parrales, J. Jimenez, M. E. Sobieszczyk, S. M. Hammer, D. J. Copenhaver, and R. P. Kulkarni, "Combining google earth and GIS mapping technologies in a dengue surveillance system for developing countries," *Int. J. Health Geogr.*, vol. 8, no. 1, p. 49, 2009.

[7] M. O. Lwin, S. Vijaykumar, O. N. N. Fernando, S. A. Cheong, V. S. Rathnayake, G. Lim, Y.-L. Theng, S. Chaudhuri, and S. Foo, "A 21st century approach to tackling dengue: Crowdsourced surveillance, predictive mapping and tailored communication," *Acta Trop.*, vol. 130, pp. 100–107, Feb. 2014.

[8] K. S. Nicholas G. Reich, Stephen A. Lauer, "Challenges in real-time prediction of infectious disease: A case study of dengue in thailand," *Online J. Public Library of Science*, vol. 7, no. 1, 2016.

[9] E. Boakes, G. Gliozzo, V. Seymour, M. Harvey, C. Smith, D. B. Roy, and M. Haklay, "Patterns of contribution to citizen science biodiversity projects increase understanding of volunteers' recording behaviour," *Scientific Reports*, vol. 6, p. 33051, 09 2016.

[10] R. Dale, "The return of the chatbots," *Natural Language Engineering*, vol. 22, pp. 811–817, Sept. 2016.

[11] S. Ho, "Why southeast asia is leading the world's most disruptive mobile business models," 2015. https://techcrunch.com/2015/09/08/why-southeast-asia-is-leading-the-worlds-most-disruptive-mobile-business-models/.

[12] A. Wojcik, "Crowdsourcing of geo-tagged images to detect mosquito breeding sites by using mobile applications," Master's thesis, University of Bremen, 10 2018.

[13] N. G. Reich, S. A. Lauer, K. Sakrejda, S. Iamsirithaworn, S. Hinjoy, P. Suangtho, S. Suthachana, H. E. Clapham, H. Salje, D. A. T. Cummings, and J. Lessler, "Challenges in real-time prediction of infectious disease: A case study of dengue in thailand," *PLOS Neglected Tropical Diseases*, vol. 10, pp. 1–17, 06 2016.

[14] H. M. Aburas, B. Gultekin Cetiner, and M. Sari, "Dengue confirmed-cases prediction: A neural network model," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4256–4260, 2010.

[15] S. A. Lauer, K. Sakrejda, E. L. Ray, L. T. Keegan, Q. Bi, P. Suangtho, S. Hinjoy, S. Iamsirithaworn, S. Suthachana, Y. Laosiritaworn, D. A. Cummings, J. Lessler, and N. G. Reich, "Prospective forecasts of annual dengue hemorrhagic fever incidence in thailand, 2010–2014," *Proceedings of the National Academy of Sciences*, vol. 115, no. 10, pp. E2175–E2182, 2018.

[16] B. M. Althouse, Y. Y. Ng, and D. A. T. Cummings, "Prediction of dengue incidence using search query surveillance," *PLoS Negl. Trop. Dis.*, vol. 5, p. e1258, Aug. 2011.

[17] M. Mehra, A. Bagri, X. Jiang, and J. Ortiz, "Image analysis for identifying mosquito breeding grounds," in *2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, 2016.

[18] C. Suduwella, A. Amarasinghe, L. Niroshan, C. Elvitigala, K. De Zoysa, and C. Keppetiyagama, "Identifying mosquito breeding sites via drone images," in *Proceedings of the 3rd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet '17*, 2017.

[19] M. O. Lwin, K. Jayasundar, A. Sheldenkar, R. Wijayamuni, P. Wimalaratne, K. C. Ernst, and S. Foo, "Lessons from the implementation of Mo-Buzz, a mobile pandemic surveillance system for dengue," *JMIR Public Health Surveill*, vol. 3, p. e65, Oct. 2017.

[20] J. Coloma, H. Suazo, E. Harris, and J. Holston, "Dengue chat: A novel web and cellphone application promotes community-based mosquito vector control," *Annals of Global Health*, vol. 82, p. 451, 05 2016.

[21] S. M. Quadri, T. K. Prashanth, S. Pongpaichet, A. A. A. Esmin, and R. Jain, "Target-ZIKA: Epidemic situation detection and risk preparedness for ZIKA virus," in *2017 10th International Conference on Ubi-media Computing and Workshops (Ubi-Media)*, 2017.

[22] J. O'Shea, Z. Bandar, and K. Crockett, *Systems Engineering and Conversational Agents*, pp. 201–232. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[23] N. Albayrak, A. Özdemir, and E. Zeydan, "An overview of artificial intelligence based chatbots and an example chatbot application," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, May 2018.

[24] J. Weizenbaum, “Eliza—a computer program for the study of natural language communication between man and machine,” *Commun. ACM*, vol. 9, pp. 36–45, Jan. 1966.

[25] D. Braun, A. Hernandez-Mendez, F. Matthes, and M. Langen, “Evaluating natural language understanding services for conversational question answering systems,” in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 174–185, Association for Computational Linguistics, 2017.

[26] D. ILIĆ and B. MARKOVIĆ, “Possibilities, limitations and economic aspects of artificial intelligence applications in healtcare,” *Ecoforum Journal*, vol. 5, no. 1, 2016.

[27] S. Divya, V. Indumathi, S. Ishwarya, P. M, and S. Kalpana Devi, “A self-diagnosis medical chatbot using artificial intelligence,” *Journal of Web Development and Web Designing*, vol. 3, 2018.

[28] D. Madhu, C. J. N. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, “A novel approach for medical assistance using trained chatbot,” in *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 243–246, March 2017.

[29] F. Amato, S. Marrone, V. Moscato, G. Piantadosi, A. Picariello, and C. Sansone, “Chatbots meet ehealth: automatizing healthcare,”

[30] A. Lokman and J. Mohamad Zain, “Designing a chatbot for diabetic patients,” 08 2007.

[31] A. Fadhil, “A Conversational Interface to Improve Medication Adherence: Towards AI Support in Patient’s Treatment,” *ArXiv e-prints*, Mar. 2018.

[32] M. Morales-Rodríguez, J. J. González Barbosa, R. Florencia, H. Fraire-Huacuja, and J. A. Martínez Flores, “Emotional conversational agents in clinical psychology and psychiatry,” 11 2010.

[33] T. Kowatsch, M. Nißen, C.-H. I. Shih, D. Rüegger, D. Volland, A. Filler, F. Künzler, F. Barata, D. Büchter, B. Brogle, K. Heldt, P. Gindrat, N. Farpour-Lambert, and D. l?Allemand, “Text-based healthcare chatbots supporting patient and health professional teams: Preliminary results of a randomized controlled trial on childhood obesity,” in *Persuasive Embodied Agents for Behavior Change (PEACH2017) Workshop, co-located with the 17th International Conference on Intelligent Virtual Agents (IVA 2017)*, August 2017.

[34] G. Cameron, D. Cameron, G. Megaw, R. Bond, M. Mulvenna, D. Siobhan O’ Neill, C. Armour, and M. Mctear, “Towards a chatbot for digital counselling,” pp. 1–7, 01 2017.

[35] A. Rizzo, B. Lange, J. Buckwalter, E. Forbell, J. Kim, K. Sagae, J. Williams, J. Difede, B. Rothbaum, G. Reger, T. Parsons, and P. Kenny, “Simcoach: An intelligent virtual human system for proving healthcare information and support,” 10 2010.

[36] D. Lee, K.-J. Oh, and H.-J. Choi, “The chatbot feels you - a counseling service using emotional response generation,” in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 437–440, Feb 2017.

[37] R. Crutzen, G.-J. Y. Peters, S. D. Portugal, E. M. Fisser, and J. J. Grolleman, "An artificially intelligent chat agent that answers adolescents' questions related to sex, drugs, and alcohol: An exploratory study," *Journal of Adolescent Health*, vol. 48, no. 5, pp. 514 – 519, 2011.

[38] B. E. Comendador, B. Michael B. Francisco, J. S. Medenilla, S. Mae T. Nacion, and T. Bryle E. Serac, "Pharmabot: A pediatric generic medicine consultant chatbot," vol. 3, pp. 137–140, 01 2015.

[39] B. Reeves and C. Nass, *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. New York, NY, USA: Cambridge University Press, 1996.

[40] C. L. Lortie and M. J. Guitton, "Judgment of the humanness of an interlocutor is in the eye of the beholder," *PLOS ONE*, vol. 6, pp. 1–7, 09 2011.

[41] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics Automation Magazine*, vol. 19, pp. 98–100, June 2012.

[42] T. Araujo, "Living up to the chatbot hype: The influence of anthropomorphic design cues and communicative agency framing on conversational agent and company perceptions," *Computers in Human Behavior*, vol. 85, pp. 183 – 189, 2018.

[43] L. Ciechanowski, A. Przegalinska, M. Magnuski, and P. Gloor, "In the shades of the uncanny valley: An experimental study of human–chatbot interaction," *Future Generation Computer Systems*, 2018.

[44] C. Nass, Y. Moon, and N. Green, "Are machines gender neutral? gender-stereotypic responses to computers with voices," *Journal of Applied Social Psychology*, vol. 27, no. 10, pp. 864–876, 1997.

[45] Y. Mou and K. Xu, "The media inequality: Comparing the initial human-human and human-ai social interactions," *Computers in Human Behavior*, vol. 72, pp. 432 – 440, 2017.

[46] B. Fogg and C. Nass, "How users reciprocate to computers: An experiment that demonstrates behavior change," in *CHI '97 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '97, (New York, NY, USA), pp. 331–332, ACM, 1997.

[47] C. Nass and Y. Moon, "Machines and mindlessness: Social responses to computers," *Journal of Social Issues*, vol. 56, pp. 81–103, 03 2000.

[48] J. Hill, W. R. Ford, and I. G. Farreras, "Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations," *Computers in Human Behavior*, vol. 49, pp. 245 – 250, 2015.

[49] T. Hu, A. Xu, Z. Liu, Q. You, Y. Guo, V. Sinha, J. Luo, and R. Akkiraju, "Touch your heart: A tone-aware chatbot for customer care on social media," *CoRR*, vol. abs/1803.02952, 2018.

[50] R. M. Schuetzler, M. Grimes, J. S. Giboney, and J. Buckman, "Facilitating natural conversational agent interactions: Lessons from a deception experiment," in *ICIS*, 2014.

[51] L. Lin, L. F. D'Haro, and R. Banchs, "A web-based platform for collection of human-chatbot interactions," in *Proceedings of the Fourth International Conference on Human Agent Interaction*, HAI '16, (New York, NY, USA), pp. 363–366, ACM, 2016.

[52] S. Atreja, P. Aggarwal, P. Mohapatra, A. Dumrewal, A. Basu, and G. B. Dasgupta, "Citicafe: An interactive interface for citizen engagement," in *23rd International Conference on Intelligent User Interfaces*, IUI '18, (New York, NY, USA), pp. 617–628, ACM, 2018.

[53] D. J. Stoner, L. Ford, and M. Ricci, "Simulating military radio communications using speech recognition and chat-bot technology," *The Titan Corporation, Orlando*, 2004.

[54] S. A. Abdul-Kader and D. J. Woods, "Survey on chatbot design techniques in speech conversation systems," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 7, 2015.

[55] S. Karve, A. Nagmal, S. Papalkar, and S. A. Deshpande, "Context sensitive conversational agent using dnn," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 475–478, March 2018.

[56] D. Dutta, "Developing an intelligent chat-bot tool to assist high school students for learning general knowledge subjects," 2017.

[57] A. Patil, M. K, N. R. A, and N. R, "Comparative study of cloud platforms to develop a chatbot," *International Journal of Engineering & Technology*, vol. 6, no. 3, pp. 57–61, 2017.

[58] T. M. Connolly and C. E. Begg, *Database Systems: A Practical Approach to Design, Implementation and Management (4th Edition)*. Pearson Addison Wesley, 2004.

[59] P. B. Brandtzaeg and A. Følstad, "Why people use chatbots," in *Internet Science* (I. Kompatsiaris, J. Cave, A. Satsiou, G. Carle, A. Passani, E. Kontopoulos, S. Diplaris, and D. McMillan, eds.), (Cham), pp. 377–392, Springer International Publishing, 2017.

[60] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh, "Quality control in crowdsourcing: A survey of quality attributes, assessment techniques and assurance actions," *CoRR*, vol. abs/1801.02546, 2018.

Online Sources

[61] S. Perez, "Majority of u.s. consumers still download zero apps per month, says comscore." https://techcrunch.com/2017/08/25/majority-of-u-s-consumers-still-download-zero-apps-per-month-says-comscore/, Aug. 2017. Accessed: 2018-8-13.

[62] S. Globalstats, "Desktop vs mobile vs tablet market share asia." http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/asia/, 2018. Accessed: 2018-9-1.

[63] S. Kahn, "Asia's mobile-first world." https://asia.googleblog.com/2014/10/asias-mobile-first-world.html, 2014. Accessed: 2019-4-2.

[64] W. are social, "Penetration of leading social networks in thailand as of 3rd quarter 2017." https://www.statista.com/statistics/284483/thailand-social-network-penetration/, 2018. Accessed: 2018-9-1.

[65] P. A. H. Organization. https://www.paho.org/hq/images/stories/2018/vector-control-500x276.jpg.

[66] U. A. F. photo/Don Peek. https://www.pittsburgh.afrc.af.mil/News/Art/igphoto/2001614464/.

[67] MosquitoAlert, "www.mosquitoalert.com." http://www.mosquitoalert.com/, 2017. Accessed: 2019-2-04.

[68] Mitsuku, "Mitsuku." https://www.pandorabots.com/mitsuku/, 2015. Accessed: 2019-4-2.

[69] Cleverbot, "Cleverbot." http://www.cleverbot.com/, 2006. Accessed: 2019-4-2.

[70] K. Wu, "Chatbots are getting unsettlingly good at conversations." https://www.inverse.com/article/37615-best-chatbot, 2017. Accessed: 2019-4-2.

[71] Gartner, "Hype cycle for emerging technologies, 2018." https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/.

[72] K. JOHNSON, "Facebook messenger passes 300,000 bots." https://venturebeat.com/2018/05/01/facebook-messenger-passes-300000-bots/, 2019. Accessed: 2019-4-2.

[73] gyant Inc., "Gyant.com." https://gyant.com/deutsch, 2017. Accessed: 2018-5-31.

[74] P. C. BV, "Florence - your health assistant." https://florence.chat/, 2017. Accessed: 2019-4-2.

[75] T. C. for Disease Control, "Taiwan cdc's chatbot upgraded on its 1st birthday: Answering infectious disease-related questions via line." https://www.cdc.gov.tw/english/info.aspx?treeid=BC2D4E89B154059B&nowtreeid=EE0A2987CFBA3222&tid=6307AE48792ED3BE, 2019. Accessed: 2019-4-2.

[76] Raheem, "Raheem.ai." https://www.raheem.ai/, 2018. Accessed: 2019-4-2.

[77] A. Debecker, "8 reasons not to use a diy chatbot builder." https://blog.ubisend.com/optimise-chatbots/8-reasons-not-to-use-a-diy-chatbot-builder, 2017. Accessed: 2019-4-2.

[78] "A comparative analysis of chatbots apis." https://medium.com/activewizards-machine-learning-company/a-comparative-analysis-of-chatbots-apis-f9d240263e1d, Apr. 2018. Accessed: 2018-8-13.

[79] A. Kang, "Understanding the differences between alexa, api.ai, wit.ai, and luis/cortana." https://medium.com/@abraham.kang/understanding-the-differences-between-alexa-api-ai-wit-ai-and-luis-cortana-2404ece0977c, Apr. 2017. Accessed: 2018-8-11.

[80] www.thainationalparks.com, "Lophura diardi commonly known as siamese fireback." https://commons.wikimedia.org/wiki/File:Siamese-fireback-Lophura-diardi-khao-yai-national-park.jpg. licensed under CC-BY-SA-4.0.

[81] Facebook, "Message templates." https://developers.facebook.com/docs/messenger-platform/send-messages/templates, 2019. Accessed: 2019-4-2.

[82] M. POSTALCIOĞLU, "Chatbot ux - design tips and considerations." https://www.toptal.com/designers/ui/chatbot-ux-design, 2017. Accessed: 2019-4-2.

[83] Pillow, "Pillow." https://pillow.readthedocs.io/en/stable/, 2019. Accessed: 2019-4-2.

[84] C. Phillips, "Why your facebook messenger chatbot is too shy." https://chatbotsmagazine.com/why-your-facebook-messenger-chatbot-is-too-shy-a40e30ea66e1, 2018. Accessed: 2019-4-2.

[85] V. Perminova, "Ways to increase chatbot user engagement by sending push-notifications." https://chatbotslife.com/ways-to-increase-chatbot-user-engagement-by-sending-push-notifications-e76f7f2fcf92, 2017. Accessed: 2019-4-2.

[86] Facebook, "24 hours policy." https://developers.facebook.com/docs/messenger-platform/policy/policy-overview#24hours_window, 2019. Accessed: 2019-4-2.

[87] Facebook, "Sponsored messages." https://developers.facebook.com/docs/marketing-api/guides/messenger-sponsored/2.8#sponsored, 2019. Accessed: 2019-4-2.

[88] Facebook, "Subscription messaging." https://developers.facebook.com/docs/messenger-platform/policy/policy-overview#subscription_messaging, 2019. Accessed: 2019-4-2.

[89] "Privacy policy bot." https://www.facebook.com/messages/t/privacypolicybot.

[90] Flask, "Flask." https://docs.sqlalchemy.org/en/latest/orm/session_state_management.html. Accessed: 2019-4-2.

[91] SQLAlchemy, "Sqlalchemy state management." https://docs.sqlalchemy.org/en/latest/orm/session_state_management.html. Accessed: 2019-4-2.

[92] Facebook, "Facebook graph api user." https://developers.facebook.com/docs/graph-api/reference/v2.6/user.

[93] ngrok, "ngrok." https://ngrok.com/. Accessed: 2019-4-2.

[94] Serveo, "Serveo." http://serveo.net/, 2019. Accessed: 2019-4-2.

[95] tmux, "tmux." https://github.com/tmux/tmux, 2019. Accessed: 2019-4-2.

Appendix A

A.1 Subscription Messaging

To enable Subscription Messaging on the Facebook developers site, the developer has to put forth a brief description of what the Facebook app is able to do and why that is beneficial for the users. This is what we submitted for the Dengue Detector Facebook Messenger bot.

Description of the project

"We are a group of Students of Mahidol University in Thailand and the University of Bremen, Germany and together we developed a system to detect breeding sites of the Aedes aegypti, which is the mosquito responsible for transmitting viral diseases like the dengue fever. The goal of the project is to generate a mapping of dengue breeding sites on unprecedented scale using Google street view and crowdsourced images, which will be helpful for public health agencies to better target interventions. The chatbot sends a location where there is insufficient coverage of google street view images to the user. The user is than tasked to visit that location to take pictures. Subsequently, the pictures are added to the database and scanned for breeding sites using object recognition empowered by a Region-based Convolutional Neural Network (R-CNN).

In addition, the chatbot is able to provide detailed information about the dengue disease itself, it's vector and the typical breeding sites of the vector. It also provides information on how many cases of dengue were recorded in a given (sub-)district in Thailand."

Example 1

“There were 100 cases of Dengue Fever in your subdistrict last year. Please consider taking pictures of mosquito breeding sites to help fight the dengue fever.”

Example 2

“Hey, there were 50 pictures taken in your area within the last week. Way to go!”

Example 3

“With an estimated 390 million cases a year, dengue is a major public health concern. By continuing contributing pictures, you can help us to contain the disease.”

A.2 Facebook App Review

Before a chatbot on Facebook Messenger is allowed to respond to real users, Facebook carries out a manual review process. To obtain the pages_messaging permission, which is nessessary to enable the bot to respond to users, some sample inputs are tested by the review team. We picked the inputs “Hi”, “Dengue?” and “Breeding Site?” and the corresponding responses from the chatbot.

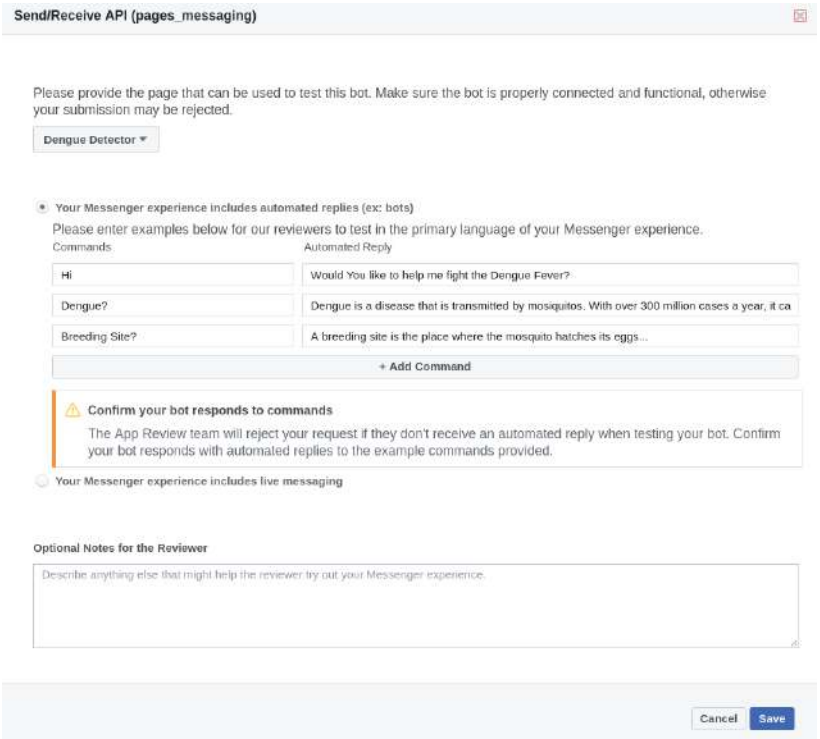


Fig. A.1 Main submission form for the pages_messaging permission review process

After going through the rest of the process and waiting for the review team to try the chatbot, we received the confirmation, that the review was successful and the bot is approved for the *pages_messaging* permission.

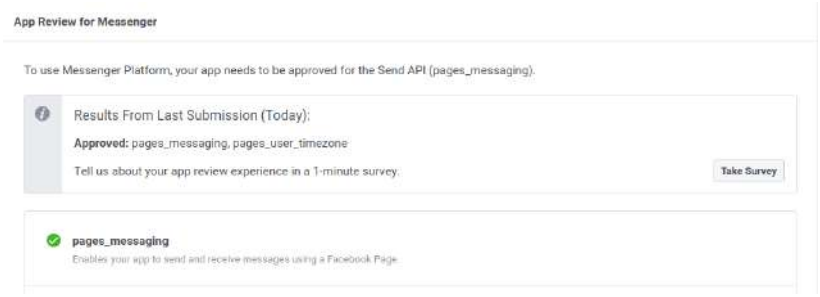


Fig. A.2 Successful Facebook review

A.3 Full Flowchart

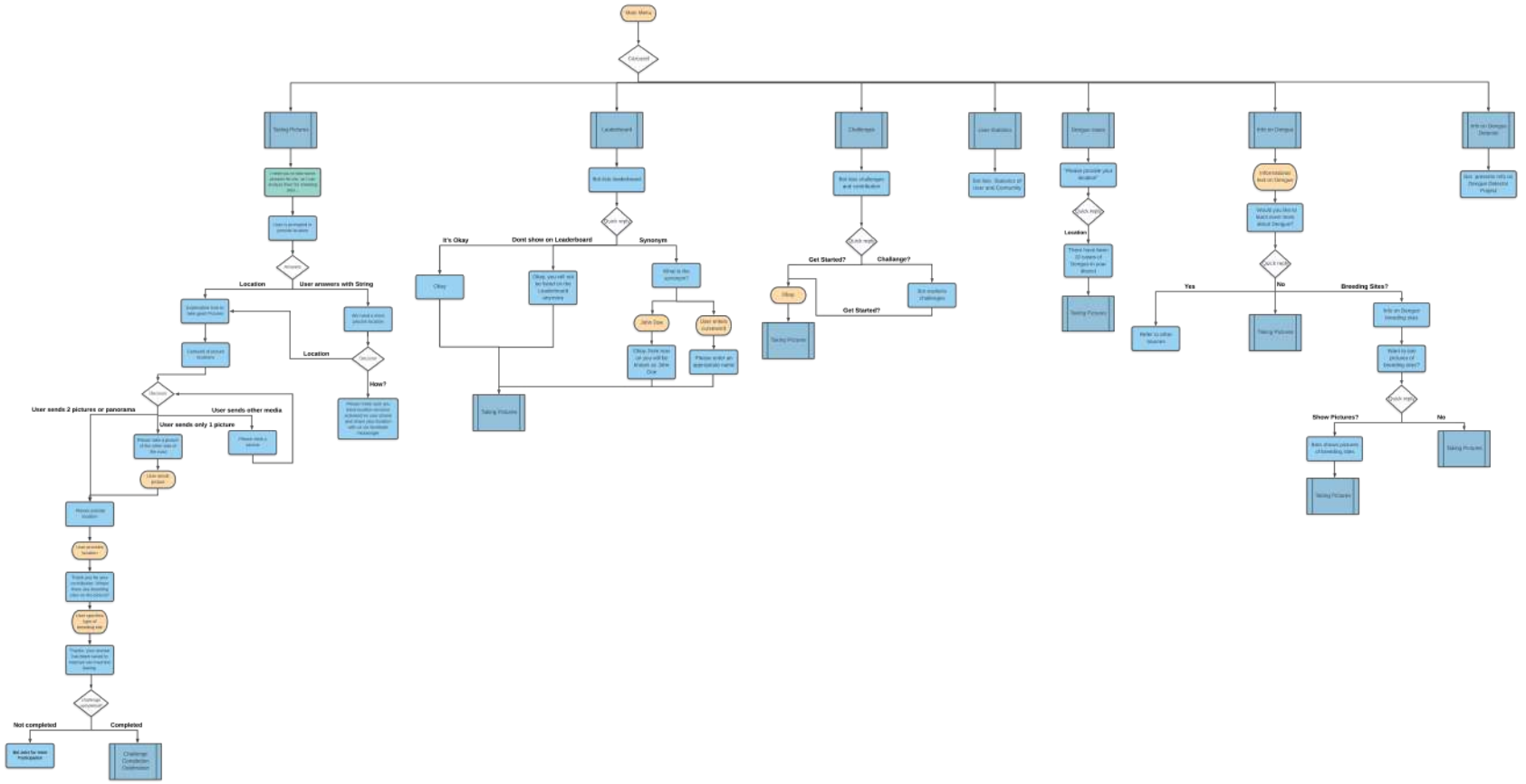


Fig. A.3 Flowchart of all intents

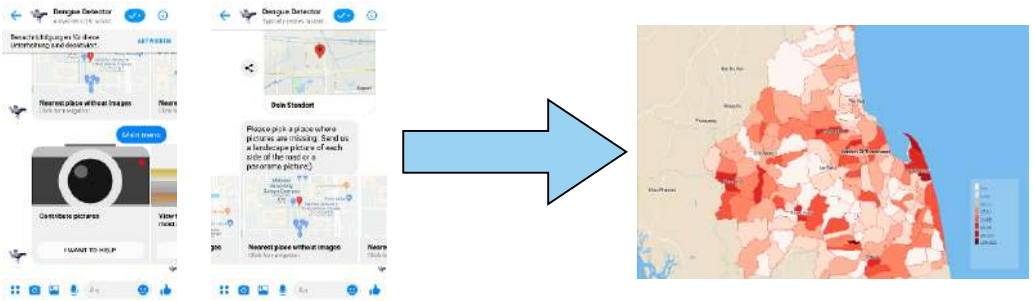
A.4 Promotional Material

DENGUE DETECTOR CHATBOT

Take pictures of breeding sites to help fight the Dengue Fever!

WITH OVER 300 MILLION CASES A YEAR AND NO EFFECTIVE VACCINE, DENGUE POSES A THREAT TO PUBLIC HEALTH

The Dengue fever is transmitted by the Aedes aegypti mosquito. Mosquito breeding sites are often in artificial water containers, e.g. old tires, buckets and vases. The goal of this project is to generate a mapping of dengue breeding sites on unprecedented scale using Google Street View and crowdsourced images, which will be helpful for public health agencies to better target interventions.



The Dengue Detector chatbot on Facebook Messenger is designed to improve the insufficient coverage of Google Street View images. Users are tasked to visit locations, where Street View images are either missing or out of date. After the user has taken the needed pictures, the pictures are added to the database and scanned for breeding sites using object recognition empowered by a Region-based Convolutional Neural Network (R-CNN). In addition, the chatbot is able to provide detailed information about the Dengue disease itself, the mosquitos and their typical breeding sites.

Johannes Schöning, Peter Haddawy, Marcel Dechert

Start chatting now at
m.me/DengueDetector

Fig. A.4 Flyer

Appendix B

Contents of Compact Disc

B.1 Sourcecode

- Sourcecode of the Dialogflow Agent
- Sourcecode of the Python Backend
- Sourcecode of the Dengue Detector-Server

B.2 Further Materials

- Raw data of evaluation in CSV format
- Video of chatbot usage