DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Detection and Dialog-Based Self-Reporting of Stress for Eating Behavior Prediction

Wenjian Li

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Detection and Dialog-Based Self-Reporting of Stress for Eating Behavior Prediction

Erkennen und dialogbasiertes Self-Reporting von Stress zur Vorhersage des Essverhaltens

Author:	Wenjian Li
Supervisor:	Prof. Dr. Georg Groh
Advisor:	Monika Wintergerst
Submission Date:	15.05.2020

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.05.2020 Wenjian Li

Acknowledgments

I would like to thank everyone who has offered me help and support on this thesis:

PD Dr. Georg Groh, for his supervision throughout the project and the inspirations he's been giving me during my studies at TUM, leading my way into the amazing field of social computing.

Ms. Monika Wintergerst, who has offered me great advice, guidance, and feedback with her expertise and patience, as well as care and understanding in this difficult time of the global pandemic.

Audrey, who has always been supporting and encouraging me not only throughout the project, but also my life, with her beautiful soul and unconditional love.

My mom Chunyan and my dad Zhenliang, who are not only parents but also life teachers, always supporting my education and guiding me with positivity and wisdom.

All the participants in the studies, who have provided crucial input and feedback to the project.

Last but not least, my friends, who have been offering me great social support despite social distancing.

Abstract

Contents

1. Introduction

Eating is an activity that people perform on a daily basis. It is the essential source of ingredients for us humans. Our nutrition intake, in turn, affects our health. However, people's choice of food cannot be simply regulated in terms of time and ingredients to make the best health effect out of it, because it is a highly emotional behavior (Gardner et al. 2014; Locher et al. 2005). According to Gardner et al., both positive and negative moods affect food choices. Especially, having negative moods often leads one to pick indulgent food instead of healthy food to cope with the emotion.

Stress is a common reaction to the environment that is often linked to negative emotions. In fact, Du et al. (2018) suggests that there is a significant positive correlation between the level of stress one has and the degree of negative emotions one experiences. Combining the results from both studies, it is therefore highly likely that food choices could be affected by stress.

A study by Mental Health Foundation (2018) suggests that a majority of the population in the United Kingdom may have been overwhelmed with stress at some time within the year 2018. This suggests that many of the health problems resulted from unhealthy eating behaviors could be linked to stress. However, regulating eating behavior often requires a deep understanding of nutrition and diet, which is not the possession of non-experts. There are professionals who are out there to offer counseling services on people's diet, but this is understandably not always accessible by the general public, given the pervasiveness of stress among them. Moreover, the specific eating behaviors resulted from stress differ among individuals (Torres and Nowson 2007). For example, the same level of stress can lead to overeating for one person, but undereating for another. It is, therefore, crucial to work out an individual's eating behavior under the influence of stress without professional medical intervention. This information can be helpful in building food recommendation systems that can detect stress, and recommend healthy food based on the user's eating patterns. The prerequisite of such is to build another (predictive) system so that given a specific user and his/her stress level, it can predict what the user is likely to eat, especially whether he/she is likely to eat more or less than usual. This thesis focuses on establishing a method to build such a system.

The first step is to collect user data. Specifically, data on users' stress and eating

behavior, as well as the relations between them. One way of doing this is to use a chatbot. Compared with more explicit ways of acquiring data, such as questionnaires or interviews, a chatbot is obviously less intrusive and offers the possibility to collect data in a real-world setting instead of in laboratories. On the other hand, users are likely to be more adherent to chatbots compared to other types of cognitive-behavioral therapeutic (CBT) apps such as self-help web-based therapy (Barak et al. 2008) given their conversational and human-like nature, which is crucial in the context of this research (Fitzpatrick et al. 2017).

This thesis presents the design, implementation, and testing of a chatbot which collects data on the users' stress information and food consumed whilst being stressed, and presents a method to build a connection between the two on a per-user basis, i.e. building a stress-eating profile for a potential user. The following chapters will provide details regarding the design, realization, and evaluation of such a system. Chapter 2 formalizes the goal and requirements of the project. It will put the terms "stress detection", "dialogue-based self-reporting of stress" and "eating behaviors" into the context of this work. Chapter 3 summarizes previous works related to the topic of this research, which focused on the effect of emotion on eating, chatbots as dietary advisors, and stress detection utilizing data collected by smartphones and wearable smart devices. Chapter 4 offers the details of the system design, focusing on the framework used for building the chatbot, the detection and self-reporting of stress, the collection and processing of eating behavior data, the design of states in the conversation flow, and how data is persisted for processing and analysis. Chapter 5 presents how the chatbot was tested among pilot users, including the recruitment process, the process of data collection, and the methods for evaluation of such experiment. This will be followed by chapter 6 which explains the data gathered, including the data size and content, as well as how the data is trained to result in predictive models. Chapter 7 demonstrates the result of both the data processing and the feedback from the participants.

There are certain limitations of this research which are discussed in chapter 8. Since this study aims at building a foundation for more sophisticated systems related to stress eating, e.g. food recommendation systems, chapter 9 is going to give suggestions to future work. Finally, chapter 10 will conclude this thesis.

1.1. Section

1.1.1. Subsection

See Table 1.1, Figure 1.1, Figure 1.2, Figure 1.3.

Table 1.1.: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

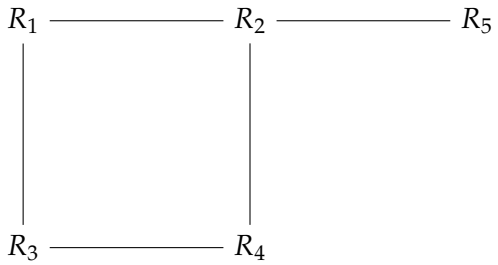

Figure 1.1.: An example for a simple drawing.

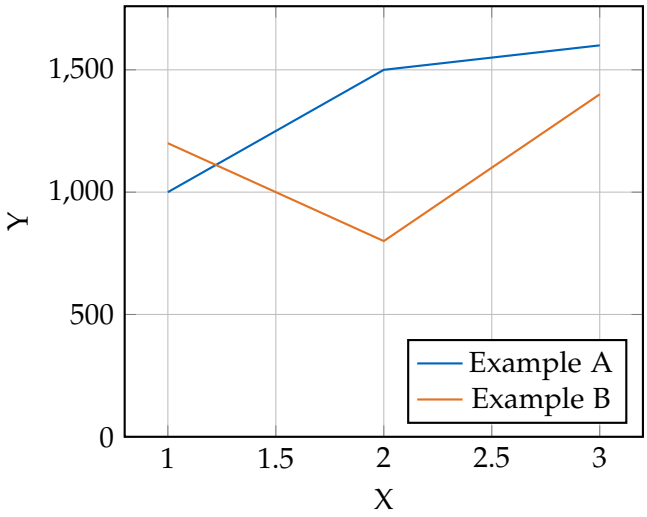

Figure 1.2.: An example for a simple plot.

```
SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 1.3.: An example for a source code listing.

2. Goal and Scope

Developing a system to serve the purposes as introduced in the previous chapter is a complicated task that covers a wide area of knowledge and various domains. Therefore, it is essential to define the scope of the study, i.e. the goal it aims to achieve, and the scope which the key terms of it is supposed to cover.

2.1. Goal

The goal of this study is to find a method to create a stress-eating profile for a random individual, based on his/her stress and eating data. More specifically, it tries to build a pipeline along which the following data can be collected via a conversational agent (which in this project was a chatbot) that interacts with its user on a daily basis:

- Data on the individual's level of stress at the time of measure, which is either detected by the conversational agent or reported by the individual through conversations
- Data on the individual's level of stress during the days for a measurement period of more than 2 weeks
- Data on the individual's food consumed when he/she was stressed at the time of measure, as self-reported by the individual to the conversational agent
- Data on the individual's amount of food consumed during the days for a measurement period of more than 2 weeks

after which it builds a classification model for the individual based on the data collected that predicts whether the individual is likely to eat more or less under the influence of stress.

Furthermore, it is expected that the method can be applied to more users over a longer period of time, and ultimately provide insights and input for other applications, such as a dietary adviser or a food recommendation system.

The following sections will define the scope of this project.

2.2. Scope of Stress Detection

Some of the most predominant research works on stress detection (Zhai and Barreto 2006; Sun et al. 2012; Melillo et al. 2011) all rely on sensor data. Sensors, especially those embedded in smart devices such as smartwatches and smart armbands, have the advantage of accessing health data including heart rate, sleep, and physical exercises, which is, on the one hand, non-intrusive, and on the other, hardly available by other forms of non-intrusive daily-used data-providers such as smartphone apps. Some smartphones are also equipped with such sensors but their data quality is relatively poor, considering that users are not carrying the phone all the time. However, for this study, we choose not to utilize wearable smart devices because the data qualities provided by different smart devices vary, while this factor has to be controlled to obtain an accurate and unbiased prediction of stress across users.

Furthermore, no contextual information such as GPS data and social activities being performed by the participants is utilized, for the chatbot platform (Rasa) (Rasa 2020) and API (Telegram Bot API) (Telegram 2020) chosen to be used do not support the gathering of such data in the background, making it intrusive to try to get these contexts (for example, in order to get the social activity a user is performing, the bot has to ask explicitly for where he/she is, whom he/she is with, and what exactly is he/she doing). More information on the choice of the framework can be found in chapter 4.

Instead, stress detection is done with a two-step process solely relying on simple conversations between the chatbot and the user. First, stress is sampled three times per day at fixed hours for every participant, i.e, the chatbot decides to ask about the user's level of stress at these timestamps. Second, based on the answers collected from the users in the first two weeks, a weighted clustering algorithm is performed on the timestamps to determine the scheduled time of the coming day of stress detection. The number of clusters is determined by the stress states of the user in the past two days. If the user was stressed the day before, stress is going to be sampled three times the day after, hence three clusters. If the user was not stressed the day before but was two days before, two clusters will be trained, leading to asking for stress twice in the coming day. Otherwise, only one cluster is going to be trained. Figure 2.1 illustrates the way this adaptive stress-sampling works.

2.3. Scope of Stress Self-Reporting

Users of the system have the option to actively report stress in the form of a conversation with the chatbot. The self-reporting uses natural language, and specifically in this

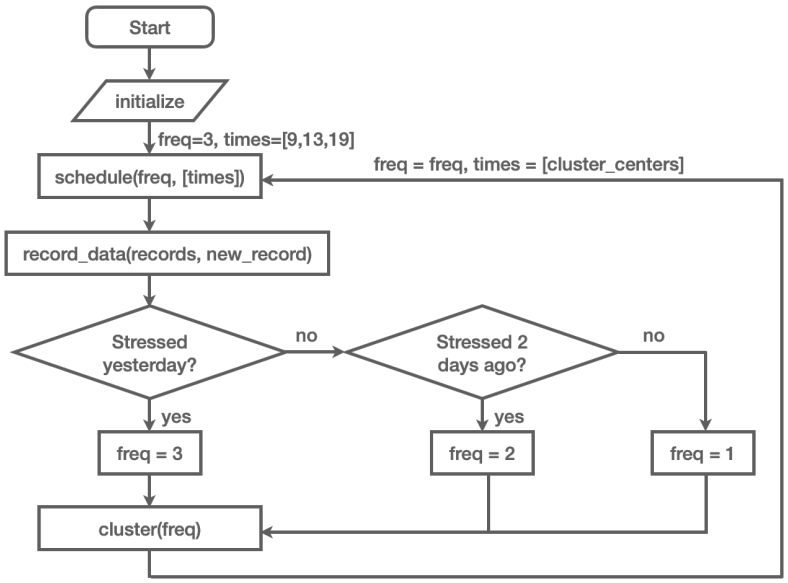

Figure 2.1.: The adaptive stress-sampling process

project, English, as a natural language understanding (NLU) module in English is trained in this project. No other forms of interaction are involved.

2.4. Definition of Eating Behaviors

In this research, the so-called "eating behaviors" refer to patterns in food consumption. The scope of such patterns differs in two scenarios, under one of which is the data the system collects which is related to eating. This data is used as input to a supervised learning scheme which builds predictive models. Under the other scenario, i.e. when such models are put into use, "eating behaviors" refer to food consumption patterns of the participants and future users that the models try to predict.

In the first scenario, the chatbot collects both descriptive and categorical data. It asks the participants to describe the food consumed, in natural language, at the time when they are stressed, or at the end of the day. In addition, it invites the participants to reflect the amount of food consumed during the day, and in case the participants have been stressed during the day, to compare this amount with the amount they are likely to eat when they are not stressed.

In the second scenario, the models try to predict only the categorical data related to food consumption, i.e. the amount of food likely to be consumed by a user given

his/her stress level. In addition, it tries to predict whether the user is likely to eat any comfort food, which is defined in the coming section.

2.5. Definition of Comfort Food

The term "comfort food" is often used to describe food, which offers comforting feelings to someone, eaten by this specific person who is being stressed. However, it is often ambiguous and its exact meaning differs in various contexts. Tomiyama et al.; Locher et al.; Spence (2011; 2005; 2017) focus on different properties of comfort food, while agree on the fact that it often contains calorie-dense food. Among them, Locher et al. (2005) offers a comprehensive categorization to comfort food into nostalgic foods, indulgence foods, convenience foods, and physical comfort foods. Comfort foods from all four categories are likely to be associated with the relief of stress, which is closely related to the topic of this research.

Since the chatbot was designed to be non-intrusive, it is unlikely to identify the social contexts the users are in and subsequently categorize the food they have eaten into one of the four categories Locher et al. presented. However, as Locher et al. also suggested, the definition of comfort food is highly personal. Therefore, this project lets the participants define what comfort food is for themselves at an individual level. This works, because in the end, individual models are built for each participant, and the definition of comfort food for one user is irrelevant to that of another. Technically, this is done by providing each user a survey towards the end of the chatbot trial (data collection phase) which contains all food items he or she has reported. The user needs to rate how likely each food item offers him or her comfort and relief when he or she is stressed. Accordingly, each stress-eating entry is labeled with the likelihood of being comfort food, which becomes part of the training data.

3. Related Work

This chapter presents some of the related work focusing on topics concerning this project, as discussed in the previous chapters. This includes work in the areas of the effect of emotions and stress on eating, chatbots as dietary advisors and other forms of behavioral change, stress detection using smartphones, and the intrusiveness and timing of digital intervention, followed by a brief review of a similar study which uses dialogue systems to assess stress eating.

3.1. The Effect of Emotions and Stress on Eating

As was discussed in chapter 1, both emotions and stress affect eating choices. This topic has been widely researched in a multitude of fields, including psychology, nutrition, and computer science.

3.1.1. The Effect of Emotions on Eating

Gardner et al. (2014) investigated how mood influences food choice. Here, mood is treated as an equivalence of emotion. The paper paid particular attention to the different mechanisms with which positive and negative moods affect food choices, pointing out that positive moods are more likely to benefit long-term goals such as health, resulting in choices of healthy food, while negative moods cue the need of relief from such moods, leading to choices of indulgent foods which often contain high levels of calories. The latter is physiologically plausible since foods with high levels of fat and sugar trigger the release of endorphins which helps with creating affectionate and happy mood (Gold et al. 1995).

Meinel (2019) also researched on the relations between an individual's emotional state and eating behavior, focusing on which specific emotions impact eating behavior the most. He made a clearer distinction among affects, emotions, and moods, which are otherwise often used interchangeably in research works. "Core affect" is defined as a rather general feeling, e.g. a sense of pleasure or displeasure. Emotions are caused by external or internal events and are triggered immediately after the events. In contrast, moods often last longer and do not necessarily connect to specific events. Based on

these definitions, Meinel conducted a survey among nutrition and diet experts, finding that emotions and moods which are on the positive and neutral spectrum of core affects are less likely to relate with eating behaviors. These moods and emotions include calm/relaxed, joy, hostility, self-assurance, confusion, and excitement. On the contrary, the most affecting emotions and moods on eating turned out to be frustrated/irritated, anger, tense, fear, sad, hopeless, bored, tired, and guilt.

3.1.2. The Effect of Stress on Eating

Torres and Nowson (2007) explored the widely accepted belief that stress influences human eating behavior, researching on the mechanism of such influence. They found that stress indeed influences eating patterns in humans. In case of such influences, stress can alter one's food intake in two ways, resulting in either over-eating or under-eating, which could be influenced by the severity of the stressor. Additionally, like Gardner et al., Torres and Nowson looked into not only the immediate effect of such influence, but also the long-term effect, concluding that chronic life stress might lead to a stronger preference towards energy and nutrient-dense foods, which in turn is one of the causes of weight gain, and in worse cases, obesity. Demographic factors were also taken into consideration, where it appeared that such effect is more significant in men than women.

3.2. Chatbots as Behavioral Change and Dietary Advisors

The use of chatbots as behavioral change advisors, or specifically, dietary advisors, has been commonplace, thanks to the benefits of chatbots as discussed in chapter 1. A typical process of such advising or intervention consists of two steps. The first step is to understand the conditions of the user, and the second is to give advice based on the conditions. The particular conditions depend on the goal of the application. For instance, as Meinel investigated the influence of moods and emotions on eating, he implemented a chatbot to track the emotional states of the user, especially those related to the most affecting moods and emotions. This system serves the data collection purpose only, and could potentially serve the data to other applications. However, it itself is not a dietary advisor. This thesis follows a similar approach that the chatbot collects data but does not provide recommendations. Nevertheless, the data includes both stress and food information, and a method to build prediction models is also presented.

Like Meinel, Horner (2019) also tried to collect data using chatbots for future recommendations. Instead of focusing on the impact of moods and emotions, he targeted at

collecting food data according to the Food Frequency Questionnaire and 24 Hour Dietary Recall, which are two popular questionnaires for standardizing food consumption. It is worth noting that Horner paid particular attention to the frequency of intervention, designing the system to ask fewer questions as possible to reduce the intrusiveness of the chatbot. Other related work and theories on the timing of intervention are to be presented in section 3.4. Likewise, my research also utilizes food data. Compared to Horner, however, the data collected in this thesis is quantitative instead of qualitative. This is because this thesis aims at providing data showing the trend of a user's stress eating patterns, while the particular food items are of secondary importance.

There are, nonetheless, a wide range of works that have already explored the recommendation part of the story. Leipold et al. (2018) developed a nutrition recommender system that is personalized, compared to many of its counterparts which only focused on the trend in the general population and the goal of losing weight. They underlined the importance of visual feedback by presenting a considerable proportion of information as graphics. The underlying recommender system is knowledge-based, consisting of four main components: a nutritional food database, a user nutrition profile, a recipe database, and a knowledge-based utility function for each nutrient (Leipold et al. 2018). It turned out that usability remained a major challenge of developing such a system, which can understandably be generalized to many similar dialog systems.

3.3. Stress Detection using Smartphone

This section summarizes two closely related research papers (Ciman, Wac, and Gaggi 2015; Ciman and Wac 2016) that have exploited smartphone usage data to detect stress. Such data includes, but is not limited to, the applications used, the usage patterns of the phone such as screen time, the patterns of typing using a keyboard such as typing speed and mistakes made, and speed and pressure of certain gestures (scrolling and swiping). The authors categorized the human-smartphone interaction (HSI) patterns into a layered structure. Figure 3.1 shows this structure. The first research (Ciman, Wac, and Gaggi 2015) conducted in a laboratory setting, utilizing level 0 and level 1 of HSI data. The second research (Ciman and Wac 2016) was a continuation of the first one, which was conducted in both laboratory and in-the-wild settings, utilizing levels 0, 1, and 3 of HSI data. User studies showed that typing speed, typing errors, and the types of applications used during the day are the most prominent features related to stress.

While these two research papers provided valuable insight into a completely non-intrusive way of gathering user data for stress prediction which does not rely on any external devices other than smartphones, implementing its theory in reality is rather challenging. On the one hand, few mobile operating systems provide developers the

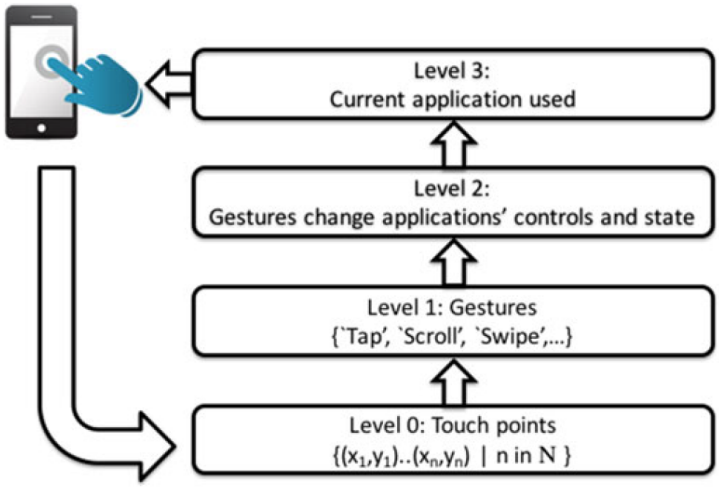

Figure 3.1.: Human-smartphone interaction levels (Ciman and Wac 2016)

freedom of tracking every gesture and keyboard input of the user. While on the other, gathering information on application usage greatly depends on the platform. It by no means rules out the possibility of building an application that is able to acquire such data, but it is likely to be an integrated one operating on both the OS level and application level. Therefore, it does not serve the goal of this thesis, where the chatbot is based on a light-weight open-source platform that has no access to users' phone data.

3.4. Intrusiveness and Timing of Digital Intervention

The timing of intervention is a choice that every designer of a chatbot has to make. Understandably, for a dialog system, the more frequently it interacts with the user, the more information it is likely to acquire, but the more intrusive it becomes, and vice versa. This section presents some of the previous research done in the area of timing.

In a study on using reminders or prompts to improve user engagement, Alkhaldi et al. (2016) made a systematic review on 14 studies with 9774 participants and came to the conclusion that technology-based strategies such as prompts can promote engagement, although most reviewed researches were conducted with relatively small sample size. It also pointed out that according to one study, notifications "sent to users shortly after they started using the digital intervention were more likely to engage users".

While the previous study focused on better engaging users, Leiva et al. (2012) emphasized on the cost of interruptions that the interaction with mobile applications results in. The study was based on the Android operating system. The combination of a switch from one foreground application to another for some duration, and then a switchback is considered an interruption. A user study was conducted with the help of a dataset provided by an Android background service. It found out that the interruptions occur in a relatively small fraction of smartphone usage, but the impact per interruption is considerable, with delayed completion of tasks by up to 4 times. This finding enhances the assumption that the timing of digital intervention is crucial not only in keeping users attached to the application but also in interfering with less of the users' normal tasks.

It is therefore obvious that system designers and software developers need to find ways to determine the best time to interact with the user, and one approach is getting to know the users' needs. In order to get the notification time right, Gültepe (2019) developed a system to predict the time when a user is likely to eat based on historical data of eating time. He collected data on a daily basis, and selected five features, i.e. day of the week, the number of meals the user has had on the day, the number of snacks, the type of the latest entry (meal or snack), and the time of that entry. The target to predict was the time until the next entry, i.e. when the user is most likely to eat next. Machine learning was used to train the regression models for prediction, where random forest and linear regression had proven to be the most prominent models. Both qualitative and quantitative measures showed that the models brought with high accuracy in predicting when a user would want to eat.

3.5. Review of a Similar System

With all the insight into the intrusiveness and timing of intervention, this section presents a very brief review of a system (Juodytė 2019) which is similar, both in terms of purposes and in terms of scope, with the chatbot that I developed.

Juodytė proposed a method to estimate the likelihood of a user overeating under the influence of stress, and support the user with a dialog system. Like this thesis, Juodytė's result can also be potentially used to design recommender systems, or be used by therapists for making dietary plans. Instead of using a machine learning approach that is popular among similar dialog systems, the author has chosen the Eating and Appraisal Due to Emotions and Stress (EADES) Questionnaire (Ozier et al. 2007) which helps to determine the likelihood of stress eating. In addition, physical data of the user such as gender, age, weight, and height were extracted. Combining the

data, a lexical analysis determines the stress level of the user and what comes next in the conversation.

Juodytė's study provided useful insight into finding the relations between stress and eating using conversations and reacting to it. However, there are limitations in a number of areas. First, it only took overeating but not undereating into consideration. Second, the system was not tested among users, leaving the resulting model not evaluated. Last but not the least, the questionnaire contains 49 questions, which takes a considerable amount of time for a user to finish, which is, based on the theories presented earlier in this section, potentially intrusive to the user.

This thesis addresses the limitations of the above study, providing a relatively non-intrusive chatbot that was tested with and evaluated by users.

4. System Design

This chapter presents the system design in detail. It will start with an introduction of the main features of the framework used for developing the chatbot, Rasa, including its features and how it interacts with the Telegram Bot API that is used to communicate with the users. Then it will show how stress detection is done with adaptive sampling. After that, it will explain how eating behavior data is collected and processed. Furthermore, the conversation flow is going to be delivered in detail. The last section will present how data is persisted.

4.1. The Rasa Framework

Rasa (2020) is an open-source framework based on natural language processing (NLP) for developing chatbots. This section introduces why it is used and its main components and features.

4.1.1. Why Rasa

The most important reason for choosing Rasa was that the software developed in this thesis is based on the project of Horner (2019) who chose Rasa as the underlying framework. However, apart from that, there are certain benefits that Rasa comes with that serve very well the goals of this project. Firstly, Rasa is completely open-source, which means it is not only free to use but also easy to customize. For example, the chatbot was designed to collect user's location data for automatic timezone conversion and image data for food reflection, and this feature can be easily fulfilled by modifying the social network platform connector of Rasa, which will be shown in detail later this section. Secondly, Rasa comes along with a sophisticated natural language understanding (NLU) model for common English. Developers only need to provide a small set of sample data for it to predict user intents with a low error rate. Thirdly, unlike its commercial counterparts such as Amazon's Lex and Google's Dialogflow, it does not depend on cloud infrastructure, and can instead be run as a piece of self-host software (Braun et al. 2017). This makes it easy to integrate other components into the chatbot. For example, a separate scheduler which not only schedules messages but also does

on-the-flight training of the data was developed alongside the chatbot and interacts well with the latter.

4.1.2. Main Components of Rasa

Meinel (2019) gave a general review of the components of Rasa. This subsection is therefore based on his review and the requirements of this project.

Rasa NLU

As the name suggests, this is the NLU module of the Rasa framework which is responsible for understanding the user input. Specifically, this module helps identify the intent of the user and entities in the application domain from English sentences the user sends to the bot. This is done by training an NLU model based on sample data provided by the developer. In the context of this work, the usage of this module is mainly limited to identifying users' intend to describe food and report stress. For each user input, another module called Rasa Core utilizes the NLU model to calculate the probability of each pre-defined intent and predicts the intent with the highest probability.

Rasa Core

While Rasa NLU takes care of understanding what a user says, Rasa Core takes the initiative to respond to the user properly. Like Rasa NLU, Rasa Core uses machine learning to train its predictive model. The training data is a set of sample conversations in the markdown format. This training data is called Rasa stories.

Rasa Stories

The markdown file records all stories used as training data for Rasa Core. Every single story starts with "##" followed by its name, and contains one or more user intent(s), denoted with "*", and one or more Rasa action(s), denoted by "-". Here is one example of a story:

```
## reflect food
* reflect_food
  - action_send_image
* describe_food
  - action_save_data
  - utter_more_or_less
```

Rasa Actions

An action is a response the chatbot performs to the user intent. There are two types of actions, namely *utterances* and *custom actions*. Utterances have to be defined in a markdown file following a certain format. Below is an example of an utterance which asks the user if he or she has eaten anything:

```
utter_did_you_eat:
- text: "Did you eat anything whilst you were stressed?"
  buttons:
  - title: Yes
    payload: /affirm
  - title: No
    payload: /deny
```

The uttered messages can have one or multiple types, which in the above case are a piece of text with two buttons. Each button can carry a payload which refers to a particular intent. When the user presses the button, its corresponding intent is sent back to the bot.

Unlike utterances that are relatively restricted, custom actions call a web server that can perform any task defined by the server and send any number of messages to the user. A typical way of implementing custom actions is to use the Python SDK offered by Rasa. Subsection 4.1.5 looks into details of custom actions implemented in this project.

Rasa Slots

Slots are key-value stores that provide memory to the chatbot. It can be viewed as variables shared among different actions in the Rasa service, as well as a bridge between the metadata used for training and the code used for handle actions. More specifically, slots can be get and set via stories (Listing 4.1) or by calling getter and setter functions in custom actions (Listing 4.2). This is otherwise impossible as, without slots, actions would become stateless.

```
* previous_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
```

```
  - utter_did_you_eat
* deny
  - utter_see_you
```

Listing 4.1: Using slots in stories to save user's stress level

```
class ActionSaveData(Action):
    def name(self) -> Text:
        return "action_save_data"
    # some lines omitted
    def run(self,
            dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        # some lines omitted
        reference_ids = []
        photo_timestamps = []
        if (intent == "describe_food"):
            entity = "food"
            try:
                reference_ids = ';'.join(tracker.get_slot('reference_ids'))
                photo_timestamps = ';'.join(
                    tracker.get_slot('photo_timestamps'))
        # some lines omitted
        entry_id = self.addToProfile(
                tracker.latest_message['text'].replace(',', ';'),
                tracker.sender_id,
                intent,
                entityValue,
                photo_timestamps,
                reference_ids)
        return [SlotSet("photo_timestamps", []),
                SlotSet("reference_ids", [entry_id])]
```

Listing 4.2: Getting and setting slots in a custom action to relate entries in persistent data. "photo_timestamps" and "reference_ids" are used to find the correct photos to be send to the user in the future which help the user to reflect food.

While Rasa was selected as the underlying platform the chatbot is based on, Telegram was chosen as the social platform to run the chatbot on. This was a straightforward choice since Horner was already using Telegram in his project (and it's worth reminding that the code of this project is based on the one of Horner). And it turned out that it was easy to find enough Telegram users from a wide variety of timezones to participate in the user trial.

Rasa provided a Telegram connector which acts as an agent connecting to the Telegram Bot API. However, since Rasa is based on NLU, no photo or location data is handled by default. It is nevertheless necessary to deal with such information as part of the feature of this bot. The next subsection presents necessary changes made to this connector to facilitate these features.

4.1.3. Interaction with the Telegram Bot API

The built-in Telegram channels provided by Rasa consists of two classes, namely *TelegramOutput* and *TelegramInput*, controlling the handling of output and input messages to users, respectively. By default, the *TelegramInput* channel only handles text data. In order for the handling of images and locations to work, this class was modified and renamed to *TelegramPlusInput*. Additionally, some other functionalities are added, which are explained below.

User Registration

A new user needs to be registered before his or her data can be recorded. As a standard approach, this registration is done right after the user issues the */start* command to the bot. It creates a folder and two CSV files to record the user data, without which the chatbot continues to interact with the user, but no data is collected. In case there was an error in the standard registration process (e.g. the user issued the */start* command when the server is down), the *TelegramPlusInput* channel triggers a backup process in case that a user message is received, but no folder of the respective user is found.

User Location and Timezone

The bot needs to collect the timezone data from the users to schedule messages to them at the right time. The timezones are calculated based on locations sent by the users. On Telegram, a location is a JSON object consisting of the latitude and longitude. The *TelegramPlusInput* connector captures this object and converts it into a Python string that can be recognized by Rasa NLU. The format of this string is defined as follows:

```
## intent:tell_lat_lng
```

```
- "lng":0.0, "lat":0.0
- "lat":0.0, "lng":0.0
```

and the respective logic in the Telegram connector is

```
elif self._is_location(msg):
    text = '{{"lng":{0}, "lat":{1}}}'.format(
        msg.location.longitude, msg.location.latitude
    )
    # store timezone information of the user
    tf = TimezoneFinder()
    latitude, longitude = msg.location.latitude, msg.location.longitude
    timezone_str = tf.timezone_at(
        lng=longitude, lat=latitude)
    with open('./user_data/timezones.csv', 'a', newline='') as csvfile:
        writer = csv.writer(
            csvfile,
            delimiter=',',
            quotechar='|',
            quoting=csv.QUOTE_MINIMAL)
        writer.writerow(
            [datetime.now(), str(msg.chat.id), timezone_str])
```

Photos

Another important feature is to process photos sent by users to the bot. If the bot detects a user being stressed and eating at the same time, instead of asking the user to describe immediately in text what he or she is eating, it asks the user to take a photo of the food. This design choice was made under the assumption that it is relatively easier for someone who is eating to take and send a photo than to text description of the food. In addition, the photos collected were included in the survey which collects users' opinions on comfort food at the end of the user study. When a user sends a file (e.g. a photo) in a Telegram conversation, Telegram saves the photo on its server that can be accessed via the *chat_id* which identifies the chat and a unique *file_id* by calling the Telegram Bot API. With this feature, the Rasa bot does not need to save the photos locally, but only need to save the *chat_id* and *file_id*. This feature is implemented by the following code in the *TelegramPlusInput* connector.

```
if not hasattr(msg, 'photo'):
    return response.text("success")
if len(msg.photo) == 0:
    print("Not a photo")
    return response.text("success")
max_size = 0
for photo in msg.photo:
    if photo.file_size > max_size:
        max_size = photo.file_size
        file_id = photo.file_id
reference_id = ''
try:
    with open(
      './user_data/' + str(str(msg.chat.id)) + '/profiles.csv',
        newline='') as csvfile:
        reader = csv.reader(
            csvfile, delimiter=',', quotechar='|')
        for row in reader:
            reference_id = row[0]
except FileNotFoundError:
    pass
with open(
  './user_data/' + str(str(msg.chat.id)) + '/photos.csv',
  'a',
  newline='') as csvfile:
    writer = csv.writer(
        csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)
    writer.writerow([datetime.now(), str(
        msg.chat.id), file_id, reference_id])
return response.text("success")
```

Naming of the Chatbot

The history of chatbot is closely coupled with that of artificial intelligence, with the goal of mimicking humans and the challenge of passing the Turing's test (Turing 1950). Since it's earliest days, chatbot has been given human-like names, from ELIZA to PARRY (Computer History Museum 2008). In fact, these names came about to sound like humans much earlier than chatbots themselves start to get out of the rule-based zone and use more intelligent means (such as machine learning) to communicate with

humans (Shum et al. 2018). Apart from this tradition, is it widely researched that having human-like names could help a chatbot to be perceived more human by the actual human chatting with it (Xu and Lombard 2017; Araujo 2018). Therefore, it is crucial to name the chatbot. The bot in this project was named Demezys, which is a combination of *Demeter* and *Oizys*. Demeter is the Greek goddess of agriculture (Kárpáti 1993) and Oizys the goddess of misery, anxiety, grief, and depression (Wikipedia 2019). This name symbolizes the combination of eating and stress at a spiritual level.

4.1.4. Training Data

This subsection explains the training data used to train Rasa Core and Rasa NLU.

domain.yml

The declaration of actions, intents, slots and templates to each utterance are saved in *domain.yml* (see section A.1).

NLU

All training data regarding the intents is saved in *nlu.md* (see section A.2). It includes the food data that is used to predict the *describe_food* intent. The items are determined according to the training data from Horner. In addition, phrases such as "I didn't eat anything when I was stressful" and "Nothing" was added to address the case when a user has no food to report.

Stories

The stories are defined in *stories.md* (see section A.3). Details about the conversation flow is to be discussed in section 4.4.

4.1.5. Custom Actions

Some of the important features of Demezys was implemented in the custom actions. This subsection discusses the details of them. A full list of custom actions can be found at section A.1.

User Registration

The user registration is done via the *action_register_user* action. This action is triggered when the user issues the */start* command to Demezys. The file path *./user_data* is created,

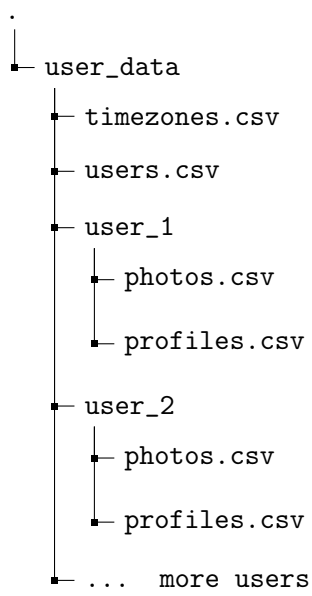

Figure 4.1.: Structure of data persistence

followed by the creation of the file *./user_data/users.csv*, which saves the *chat_id* of the user. Afterwards, it creates a path *./user_data/<chat_id>* for holding the user data, and two files *profiles.csv* and *photos.csv* under that path.

Saving Data

action_save_data is responsible for saving the stress and eating data in the *profiles.csv* files. The format of the data is illustrated in section 4.5. It first checks the latest intent of the user in order to take the corresponding actions.

If the intent was *describe_food*, it will try to get the slot values of *reference_ids* and *photo_timestamps*, which potentially link the food photo(s) being described to a previous stress entry, building the connection between stress and eating. It could be, however, that the user was not describing food according to photo(s). This situation can be broken down into two cases. One is that the user is describing food right after he or she told the bot the stress level. In this case, the *describe_food* entry is linked to the latest stress entry. The other case is that the user is describing food at the end of the day which does not refer to a specific stress entry but the day in general. In such a case, no stress entry is linked to the *describe_food* entry.

If the intent was not *describe_food*, it will directly extract the entity and save the data entry together with the intent.

Sending Image/Imanges

action_send_image extracts all the *file_id*s of the photos a user has sent to Demezys during the day and sends the photos back to the user by calling the Telegram Bot API, together with a text prompting the user to reflect food according to the photos. If no photos are found, it asks the user to reflect food directly. If the user was not detected stressed during the day, it directly sends the *utter_pieces* action to the user, which asks about how many pieces of meal/snacks the user has had during the day, bypassing the process of describing food.

Asking Location

After a user is registered, Demezys asks the user to send his or her location via Telegram. This is done by *action_ask_location*. It prompts the user with a text saying "Please send your location to me. I need this information to schedule message to you in the right time. Please use the Telegram built-in function to send location, and **DO NOT** send a text message." while sending a photo (Figure 4.2) instructing the user to do so, in case some people are not familiar with the share-location feature of Telegram.

4.2. Stress Detection with Adaptive Sampling

As discussed in section 2.2, collecting stress data relies solely on adaptive sampling. This section provides details on how the sampling works.

In this project, stress is measured with levels from 0 to 5, 0 being not stressed and 5 being extremely stressed. This straightforward one-dimension scaling has proven to be effective in measuring stress and is being widely used in psychology questionnaires (Cushway et al. 1996). In addition, users only need to spend a minimal amount of time to answer such a question, making the chatbot less intrusive.

The stress samplings are done by a scheduler written in Python. It schedules messages on a per-user basis. Upon registration, a user sends his or her location to Demezys, and the location is converted to its corresponding timezone and saved in the *timezones.csv* file together with the *chat_id*. Initially, the scheduler sends four messages per day, three of which being *detect_stress* and one being *reflect_food*. *detect_stress* triggers the bot to ask "are you feeling stressed", entering into the process of collecting the user's stress level. *reflect_food* asks the user to reflect on the food he or she has eaten when feeling stressed during the day, as well as the amount of food. The scheduling of

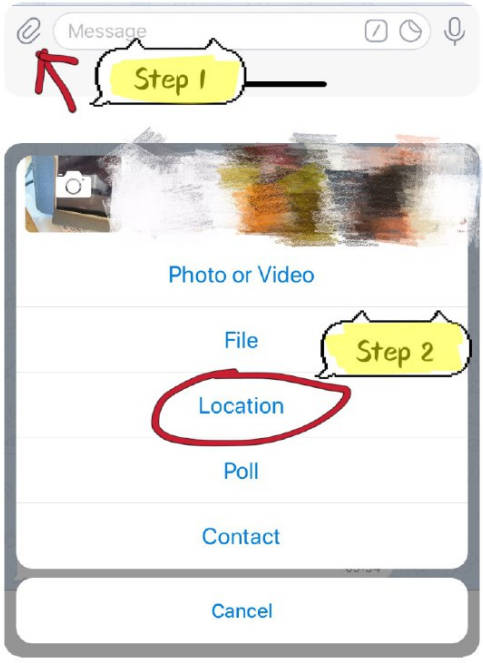

Figure 4.2.: Instruction to sharing location via Telegram

reflect_food is set to be 21:00 every day, with the assumption that this is the time between dinner and sleep for most of the people. *detect_stress* are initially scheduled 3 times per day, at 9:00, 13:00, and 19:00, which are slightly later than the times when most people are likely to have meals. The bot is designed to ask about the stress level of one hour before, in case the user says he or she is not stressed at the moment (section 4.4). The scheduling scheme is therefore assumed to hit the time when the users are eating with a higher possibility than random scheduling.

To better understand when participants are likely to feel stressed on an individual basis, an adaptive scheduling model was developed. This model not only adapts the scheduled timestamps based on historical values but also the number of stress samplings per day. This reduces the number of times when the chatbot asks about stress for those who feel stressed less frequently than others (Figure 2.1), ensuring that Demezys is not "spamming" its users.

The adaptive sampling is achieved by using k-means clustering on previous timestamps of stress entries. First, all previous stress entries are collected, with the cor-

responding timestamps and stress levels. All timestamps were collected in the 'Europe/Berlin' timezone. Then, the timestamps are converted to the local time of the users according to the timezone saved in *timezones.csv* Afterwards, the difference between each timestamp and the timestamp of the start of its day is calculated and saved. For example, for timestamp `2020-04-07 13:00:50` is transformed to 46850 seconds, which is the difference between itself and the start of the day (`2020-04-07 00:00:00`). Each saved time difference is a data point in integer form. Finally, each data point is copied *n-1* times where *n* is the stress level corresponding to that data point. This is to make sure that timestamps associated with higher levels of stress get more weight in the clustering training so that the training result can better predict the time when the user is likely to feel more stressed. This is based on the intuitive assumption that immense stress does not come and go abruptly so that a person is likely to feel some level of stress shortly before and after the timestamp when he or she is feeling a high level of stress.

The centroid of each cluster is converted back to a timestamp from its integer form. The `hours` of the centroids will be the hours when Demezys schedules the new *detect_stress* message the coming day. The k-means clustering algorithm is applied to all 18 participants whose *chat_id*s are even numbers, among the total of 33 participants. Participants with odd *chat_id*s receive reminders at fixed times per day, 3 times a day as the controlled group. The participants are not informed of this setting.

4.3. Collection and Processing of Eating Behavior Data

There are three types of eating behavior data collected, namely food descriptions, amount compared to the non-stress state, and the number of pieces.

4.3.1. Food Descriptions

Food descriptions are collected whenever a user tells Demezys that he or she was stressed one hour ago and was eating then. In addition, if there is at least one record with a positive stress level during the day, the user will be prompted to give food descriptions either according to the photo(s) he or she has sent during the day, or plainly if there is no photo. To collect food descriptions, Demezys asks the user either of the following questions

- Hi, time to reflect what you've eaten when you were stressed. Please describe the food on the photo(s).
- Please describe the food you've eaten when you felt stressed today.

according to the situation. Users' answers to these questions are saved into *profiles.csv* as food descriptions.

Towards the end of this study, the food items from the descriptions were extracted manually and sent back to the users for comfort-food labeling. The binary label is added back to each food record for training the prediction models.

4.3.2. Food Amount Compared to the Non-Stress State

Right after food descriptions, the food amount compared to the non-stress state is asked at the end of each day in case the user has reported at least once a positive stress level. This is done by the following action

```
utter_more_or_less:
- text: "Was it more or less than the amount you would normally eat
  when you're not stressed?"
  buttons:
  - title: More
    payload: /tell_more_or_less{"more_or_less":"more"}
  - title: Less
    payload: /tell_more_or_less{"more_or_less":"less"}
  - title: Same
    payload: /tell_more_or_less{"more_or_less":"same"}
```

where three buttons are presented to the user. The value of the slot *more_or_less* is recorded.

4.3.3. Number of Pieces

While the *more_or_less* values are relative to a non-stressed state, there is another piece of information collected which is the number of *pieces*. A piece can be either a meal, an afternoon tea, an extra dessert, or a snack/some snacks a person eats at once. It is an absolute value that tells how frequently the user eats. Like *more_or_less*, *pieces* are also reported by the users at the end of the day, after Demezys prompts the users with the action:

```
utter_pieces:
- text: "How many pieces did you eat in total today?
  A piece can be either a meal, an afternoon tea, an extra dessert,
  or a snack/some snacks you eat at once."
  buttons:
  - title: 0
```

```
  payload: /tell_pieces{"pieces":"0"}
- title: 1
  payload: /tell_pieces{"pieces":"1"}
- title: 2
  payload: /tell_pieces{"pieces":"2"}
- title: 3
  payload: /tell_pieces{"pieces":"3"}
- title: 4
  payload: /tell_pieces{"pieces":"4"}
- title: 5
  payload: /tell_pieces{"pieces":"5"}
- title: 6
  payload: /tell_pieces{"pieces":"6"}
- title: 6+
  payload: /tell_pieces{"pieces":"N"}
```

4.4. Conversation Flow

The choices made by the chatbot follow a certain conversation flow, meaning that it determines its actions based on previous conversations (user intents and chatbot actions). The major features within the conversation flow are onboarding, detecting stress, recording food, reflecting food, and a number of auxiliary conversation paths. Figure 4.3 summarizes the conversation flow, part of which has been discussed in previous sections.

4.5. Data Persistence

In the previous sections, the design of the chatbot system was presented comprehensively. This last section of the chapter discusses how data collected via the chatbot is persisted for later analysis.

As introduced, the data of each user is saved into two files, namely *profiles.csv* and *photos.csv*. The scheme of each file is listed below.

- **profiles.csv:** [EntryID, Timestamp, ChatID, Message, Intent, Entity, PhotoTimestamp, ReferenceID]
- **photos.csv:** [Timestamp, ChatID, FileID, StressRecordID]

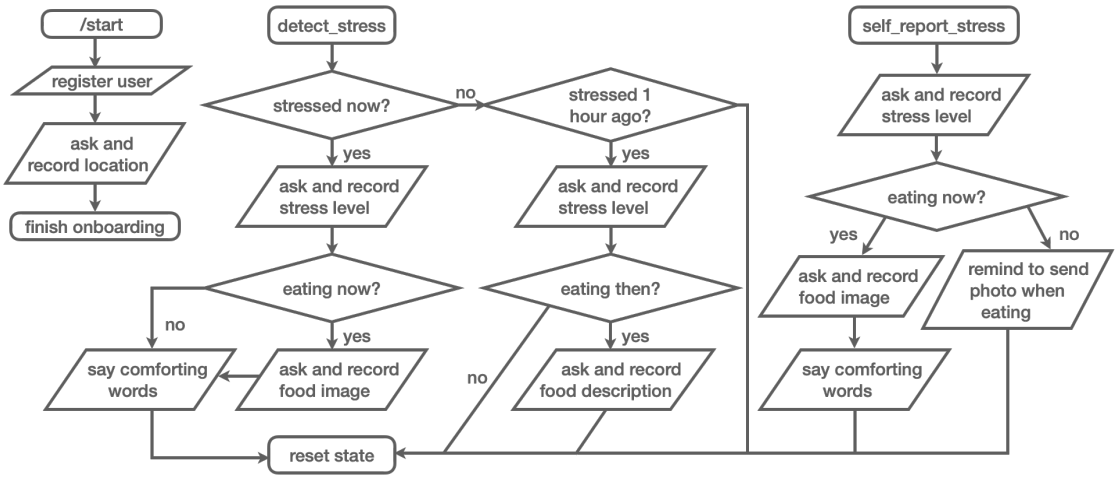

Figure 4.3.: Conversation Flow

The choice that the data was saved in CSV files instead of a database, is out of the fact that the relations between entries are rather simple. Specifically, only food descriptions need to relate to their image counterparts (if there is any) and ultimately the corresponding stress entry when the eating behavior happened. This is done by the *ReferenceID* entry in *profiles.csv* and *StressRecordID* in *photos.csv*. Whenever an image of the food is retrieved from the Telegram server, its *StressRecordID* is retrieved and saved in a Rasa slot, which will be linked to the entry in *profiles.csv* whose *EntryID* matches it. Another benefit of saving the data in CSV files is that it is straightforward to bulk process the data for analysis based on machine learning tools.

The timestamps are kept regarding the timezone of the server where the chatbot is run. When user-specific timestamps are required, they are obtained by converting the timezone to that of the user saved in *timezones.csv*. *Message* saves the original user message to the bot, while the *Intent* field is self-explanatory. *Entity* saves the useful information extracted from the message and the intent, e.g. "less" for *tell_more_or_less* or "4" for *tell_stress_level*. In case a food description refers to two or more photos, multiple photo timestamps and reference IDs are saved in the respective fields in the form of formatted lists. Table 4.1 shows the *profiles.csv* of one participant with data collected within a day (*ChatID* is omitted due to privacy concern).

Table 4.1.: Part of a profile.csv file generated for a participant

EntryID	Timestamp	Message	Intent	Entity	PhotoTimestamp	ReferenceID
45	2020-04-12 03:09:51	/tell_stress_level{level:2}	tell_stress_level	2		
46	2020-04-12 08:00:19	/tell_stress_level{level:3}	tell_stress_level	3		
47	2020-04-12 13:01:22	/tell_stress_level{level:2}	tell_stress_level	2		
48	2020-04-12 15:03:41	Fried chicken wings	describe_food		2020-04-12 13:01:39	47
49	2020-04-12 15:03:49	/tell_more_or_less{more_or_less:same}	tell_more_or_less	same		
50	2020-04-12 15:03:54	/tell_pieces{pieces:6}	tell_pieces	6		

5. Experiment

6. Data Analysis

7. Result

8. Limitations

9. Future Work

10. Conclusion

A. Rasa Training Data

A.1. domain.yml

```
%YAML 1.1
---
actions:
- action_register_user
- action_save_data
- action_send_image
- utter_are_you_eating
- utter_ask_food
- utter_ask_for_image
- action_ask_location
- utter_ask_stress_level
- utter_cheers
- utter_describe_food
- utter_did_you_eat
- utter_goodbye
- utter_good_night
- utter_greet
- utter_happy
- utter_iamabot
- utter_intro
- utter_more_or_less
- utter_pieces
- utter_remind_to_take_photo
- utter_see_you
- utter_stress
- utter_stress_delayed
- utter_success
- utter_unstressed
entities:
```

```
- level
- more_or_less
- pieces
intents:
- affirm
- bot_challenge
- deny
- describe_food
- detect_stress
- eating_now
- goodbye
- not_eating_now
- now_no_stress
- now_stress
- photo_sent
- polite
- previous_no_stress
- previous_stress
- reflect_food
- register
- start
- stressful_yes
- stressful_no
- tell_lat_lng
- tell_more_or_less
- tell_pieces
- tell_stress_level
- thank
slots:
  level:
    type: text
  more_or_less:
    type: text
  photo_timestamps:
    type: list
  pieces:
    type: text
  reference_ids:
    type: list
```

```
templates:
  utter_are_you_eating:
  - text: Are you eating right now?
    buttons:
    - title: Yes
      payload: /eating_now
    - title: No
      payload: /not_eating_now
  utter_ask_food:
  - text: "What did you eat?"
  utter_ask_for_image:
  - text: "Please take a photo of what you're eating and sent it to me
    now. When the photo is sent, press 'Done'."
    buttons:
    - title: Done
      payload: /photo_sent
  utter_ask_stress_level:
  - text: "Can you tell me the intensity of your stress within the scale
    between 1 (slightly stressful) and 5 (extremely stressful)?"
    buttons:
    - title: 1
      payload: /tell_stress_level{"level":"1"}
    - title: 2
      payload: /tell_stress_level{"level":"2"}
    - title: 3
      payload: /tell_stress_level{"level":"3"}
    - title: 4
      payload: /tell_stress_level{"level":"4"}
    - title: 5
      payload: /tell_stress_level{"level":"5"}
  utter_cheers:
  - text: "Cheers"
  utter_describe_food:
  - text: "Hi, time to reflect what you've eaten when you were stressed.
    Please describe the food on the photo(s)."
  utter_did_that_help:
  - text: "Did that help you?"
  utter_did_you_eat:
  - text: "Did you eat anything whilst you were stressed?"
```

```
  buttons:
  - title: Yes
    payload: /affirm
  - title: No
    payload: /deny
utter_goodbye:
- text: Bye
utter_good_night:
- text: Got it. Good night
utter_greet:
- text: Hey! How are you?
utter_happy:
- text: "Great, carry on!"
utter_iamabot:
- text: "I am a bot, powered by Rasa."
utter_intro:
- text: "Welcome on board. Demeter is the Greek goddess of agriculture,
  and Oizys the goddess of misery, anxiety, grief, and depression.
  This is how I got my name. Eating is a highly emotional behaviour,
  and it's the first step to relate stress with eating that will
  eventually help you to eat mindfully.\n\nPlease press the button
  below to continue"
  buttons:
  - title: Start the Journey
    payload: /register
utter_more_or_less:
- text: "Was it more or less than the amount you would normally eat
  when you're *not* stressed?"
  buttons:
  - title: More
    payload: /tell_more_or_less{"more_or_less":"more"}
  - title: Less
    payload: /tell_more_or_less{"more_or_less":"less"}
  - title: Same
    payload: /tell_more_or_less{"more_or_less":"same"}
utter_pieces:
- text: "How many *pieces* did you eat in total today? A *piece*
  can be either a meal, an afternoon tea, an extra dessert, or a
  snack/some snacks you eat at once."
```

```
  buttons:
  - title: 0
    payload: /tell_pieces{"pieces":"0"}
  - title: 1
    payload: /tell_pieces{"pieces":"1"}
  - title: 2
    payload: /tell_pieces{"pieces":"2"}
  - title: 3
    payload: /tell_pieces{"pieces":"3"}
  - title: 4
    payload: /tell_pieces{"pieces":"4"}
  - title: 5
    payload: /tell_pieces{"pieces":"5"}
  - title: 6
    payload: /tell_pieces{"pieces":"6"}
  - title: 6+
    payload: /tell_pieces{"pieces":"N"}
utter_remind_to_take_photo:
- text: "If you eat in the coming **2 hours**, please **take a
  photo of the food** and **send it to me**."
utter_see_you:
- text: "Give yourself some space."
- text: "Take a deep breath."
- text: "A hug from me and everything will be fine. Bye."
- text: "Try to break big problem down to small tasks, and it
  might help you turn stress into productivity. Bye."
- text: "Make yourself a cup of tea or coffee."
- text: "Many people are dealing with stress just like you.
  Talk to someone close to you and you can help each other."
- text: "Reflect on how you dealt with stress last time and
  that might help. Bye."
- text: "It's said that focusing on your body and finding
  tense muscles can help calming down. Give it a try."
- text: "Remember, storms don't last forever. Bye."
- text: "Find yourself a comfortable place. This is the first
  step dealing with stress. Bye."
utter_stress:
- text: "Are you feeling stressed?"
  buttons:
```

```
    - title: Yes
      payload: /now_stress
    - title: No
      payload: /now_no_stress
  utter_stress_delayed:
  - text: "What about one hour ago? Were you stressed then?"
    buttons:
    - title: Yes
      payload: /previous_stress
    - title: No
      payload: /previous_no_stress
  utter_success:
  - text: "You're successfully registered."
  utter_unstressed:
  - text: "Great to know!"
```

A.2. nlu.md

```
## intent:affirm
- yes
- indeed
- of course
- that sounds good
- correct
- a little
- a little bit
- a bit
- ja

## intent:describe_food
- an apple
- Apfelstrudel
- Aprikose
- aubergine
- avocado
- a bread
- a bread and a cup of coffee
```

```
- a chocolate bar
- a burger
- a coffee
- bacon
- baguette
- a baguette
- bami
- banana
- banana and chocolate
- barbecue
- BBQ
- beef
- berries
- berry
- biscuits
- blackberries
- blueberries
- Bonbon
- breakfast
- brezel
- burger
- burrito
- butter
- cake
- cakes
- candy
- candies
- carotte
- cereal
- cheese
- cheese burger
- cherries
- chicken
- chicken nuggets
- chicken wings
- chili con carne
- chips
- chocolate
- cookies
```

```
- corn
- crab
- crepe
- crepes
- croissant
- a croissant
- cucumber
- curry
- Currywurst
- dessert
- dim sum
- dinner
- doner
- Donerkebab
- donut
- doughnut
- duck
- dumplings
- egg
- eggplant
- eggs
- falafel
- fish
- fish and chips
- flour
- fondue
- food
- fried
- fried rice
- grapes
- grape fruit
- ham
- hamburger
- Haxe
- hazel nuts
- honey
- hotpot
- hot dog
- icecream
```

```
- instant noodles
- jam
- juice
- just candies
- kebab
- ketchup
- KFC
- kimchi
- Knodel
- lamb
- lasagne
- Lebkuchen
- lemon
- liver
- lunch
- mango
- marmelade
- marone
- maroni
- marshmallow
- mayo
- McDonald's
- meat
- meat ball
- meatball
- melon
- milk tea
- mint
- miso
- mousse
- Mozzarella
- mushrooms
- musli
- m&m
- M&M
- m & m
- noodles
- Nutella
- octopus
```

```
- olive
- omelette
- onion
- orange
- paprika
- parmesan
- pasta
- peach
- peanut
- pear
- peperoni
- pizza
- pommes
- popcorn
- pork
- potato
- pretzel
- pudding
- raisin
- ramen
- ravioli
- ribs
- rice
- rice noodles
- risotto
- Ritter Sport
- rose
- salad
- salami
- salmon
- sandwich
- sashimi
- sausage
- seafood
- Schnitzel
- Schweinshaxe
- shrimps
- some snacks
- soup
```

```
- spaghetti
- spring roll
- steak
- strawberry
- strawberries
- supper
- sushi
- sweet
- sweet melon
- taco
- tapas
- tart
- tatar
- tea
- tiramisu
- tofu
- tomato
- tortilla
- turkey
- udon
- vegetables
- wasabi
- wheat
- wok
- wrap
- wurst
- yogurt
- I didn't eat anything
- no food
- I didn't eat anything when I was stressful
- I don't eat when I'm stressful
- nothing
- I did not eat anything

## intent:detect_stress
- detect stress
- hello
- Guten Tag
- hey there
```

```
- hi
- hey
- Hi
- Hallo Foodbot.
- Servus
- Hi
- Hallo
- good morning
- good evening

## intent:goodbye
- bye
- goodbye
- Tschuess
- Goodbye
- see you around
- see you later

## intent:polite
- good night
- same to you
- sweet dream
- see you tomorrow
- see you

## intent:previous_stress
- previous stress

## intent:reflect_food
- reflect food

## intent:register
- register me

## intent:stressful_yes
- I am feeling stressful
- I'm worried
- I feel tense
- I am nervous
```

```
- stressful
- tense
- worried
- nervous

## intent:stressful_no
- I am calm
- I feel relaxed
- calm
- everything's ok
- relaxed

## intent:tell_lat_lng
- "lng":0.0, "lat":0.0
- "lat":0.0, "lng":0.0

## intent:thank
- thank you
- thanks
- cheers

## intent:bot_challenge
- are you a bot?
- are you a human?
- am I talking to a bot?
- am I talking to a human?
```

A.3. stories.md

```
<!-- User registration -->
## user register
* register
  - action_register_user
  - action_ask_location
* tell_lat_lng
  - utter_success
  - action_restart
```

```
<!-- Reacting to Stress -->

## rightly detect stress - eating
* detect_stress
  - utter_stress
* now_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_are_you_eating
* eating_now
  - utter_ask_for_image
* photo_sent
  - utter_see_you

## rightly detect stress - eating - repeat
* detect_stress
  - utter_stress
* now_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_are_you_eating
* eating_now
  - utter_ask_for_image
* photo_sent
  - utter_see_you

## rightly detect stress - not eating
* detect_stress
  - utter_stress
* now_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
```

```
  - utter_are_you_eating
* not_eating_now
  - utter_see_you

## wrongly detect stress
* detect_stress
  - utter_stress
* now_no_stress
  - slot{"level":"0"}
  - action_save_data
  - utter_stress_delayed

## wrongly detect stress - previously stress - eaten
* previous_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_did_you_eat
* affirm
  - utter_ask_food
* describe_food
  - action_save_data

## wrongly detect stress - previously stress - haven't eaten
* previous_stress
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_did_you_eat
* deny
  - utter_see_you

## wrongly detect stress - previously no stress
* previous_no_stress
  - utter_unstressed

## self-report stress - eating
```

```
* stressful_yes
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_are_you_eating
* eating_now
  - utter_ask_for_image
* photo_sent
  - utter_see_you

## self-report stress - not eating
* stressful_yes
  - utter_ask_stress_level
* tell_stress_level{"level":"1"}
  - slot{"level":"1"}
  - action_save_data
  - utter_are_you_eating
* not_eating_now
  - utter_remind_to_take_photo

<!-- Reflection on Food -->

## reflect food
* reflect_food
  - action_send_image
* describe_food
  - action_save_data
  - utter_more_or_less

## tell more or less
* tell_more_or_less{"more_or_less":"more"}
  - slot{"more_or_less":"more"}
  - action_save_data
  - utter_pieces

## tell pieces
* tell_pieces{"pieces":"1"}
  - slot{"pieces":"1"}
```

```
  - action_save_data
  - utter_good_night

<!-- Auxiliary -->

## start
* start
  - utter_intro

## gratitude
* thank
  - utter_cheers

## bot challenge
* bot_challenge
  - utter_iamabot

## say goodbye
* goodbye
  - action_restart

## being polite
* polite
  - action_restart

## being polite 2
* reflect_food
  - action_send_image
* polite
  - action_restart

## save data whenever user describes food
* describe_food
  - action_save_data
```

List of Figures

List of Tables

Bibliography

Alkhaldi, G., F. L. Hamilton, R. Lau, R. Webster, S. Michie, and E. Murray (Jan. 2016). "The Effectiveness of Prompts to Promote Engagement With Digital Interventions: A Systematic Review." In: *J Med Internet Res* 18.1, e6. ISSN: 1438-8871. DOI: 10.2196/jmir.4790.

Araujo, T. (2018). "Living up to the chatbot hype: The influence of anthropomorphic design cues and communicative agency framing on conversational agent and company perceptions." In: *Computers in Human Behavior* 85, pp. 183–189. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2018.03.051.

Barak, A., L. Hen, M. Boniel-Nissim, and N. Shapira (2008). "A Comprehensive Review and a Meta-Analysis of the Effectiveness of Internet-Based Psychotherapeutic Interventions." In: *Journal of Technology in Human Services* 26.2-4, pp. 109–160. DOI: 10.1080/15228830802094429. eprint: https://doi.org/10.1080/15228830802094429.

Braun, D., A. Hernandez Mendez, F. Matthes, and M. Langen (Aug. 2017). "Evaluating Natural Language Understanding Services for Conversational Question Answering Systems." In: DOI: 10.18653/v1/W17-5522.

Ciman, M., K. Wac, and O. Gaggi (2015). "iSensestress: Assessing stress through human-smartphone interaction analysis." In: *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, pp. 84–91.

Ciman, M. and K. Wac (July 2016). "Individuals' Stress Assessment Using Human-Smartphone Interaction Analysis." In: *IEEE Transactions on Affective Computing* PP, pp. 1–1. DOI: 10.1109/TAFFC.2016.2592504.

Computer History Museum (2008). *Computer History Museum - Exhibits - Internet History - 1970's*. URL: https://web.archive.org/web/20080221093646/http://www.computerhistory.org/internet_history/internet_history_70s.shtml (visited on 04/26/2020).

Cushway, D., P. A. Tyler, and P. Nolan (1996). "Development of a stress scale for mental health professionals." In: *British Journal of Clinical Psychology* 35.2, pp. 279–295. DOI: 10.1111/j.2044-8260.1996.tb01182.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2044-8260.1996.tb01182.x.

Du, J., J. Huang, Y. An, and W. Xu (Jan. 2018). "The Relationship between stress and negative emotion: The Mediating role of rumination." In: *Clinical Research and Trials* 4. DOI: 10.15761/CRT.1000208.

Fitzpatrick, K. K., A. Darcy, and M. Vierhile (June 2017). "Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial." In: *JMIR Ment Health* 4.2, e19. ISSN: 2368-7959. DOI: 10.2196/mental.7785.

Gardner, M. P., B. Wansink, J. Kim, and S.-B. Park (2014). "Better moods for better eating?: How mood influences food choice." In: *Journal of Consumer Psychology* 24.3, pp. 320–335. DOI: 10.1016/j.jcps.2014.01.002. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1016/j.jcps.2014.01.002.

Gold, A. E., K. M. MacLeod, B. M. Frier, and I. J. Deary (1995). "Changes in mood during acute hypoglycemia in healthy participants." In: *Journal of Personality and Social Psychology* 68.3, pp. 498–504. DOI: 10.1037/0022-3514.68.3.498.

Gültepe, O. (2019). "Notification Timing for a Proactive Virtual Dietary Advisor." unpublished bachelor's thesis supervised by PD Dr. Georg Groh, advised by Monika Wintergerst. Technical University of Munich, Munich, Germany.

Horner, L. (2019). "Dialog System-Based Dietary Assessment for a Virtual Dietary Advisor." unpublished bachelor's thesis supervised by PD Dr. Georg Groh, advised by Monika Wintergerst. Technical University of Munich, Munich, Germany.

Juodytė, M. (2019). "Dialog System-Based Assessment of Stress Eating." unpublished bachelor's thesis supervised by PD Dr. Georg Groh, advised by Monika Wintergerst and Martin Lurz. Technical University of Munich, Munich, Germany.

Kárpáti, J. (1993). "Amaterasu and Demeter: About a Japanese – Greek Mythological Analogy." In: *International Journal of Musicology* 2, pp. 9–21. ISSN: 09419535.

Leipold, N., M. Madenach, H. Schäfer, M. Lurz, N. Terzimehic, G. Groh, M. Böhm, K. Gedrich, and H. Krcmar (2018). "Nutrilize a Personalized Nutrition Recommender System: an Enable Study." In: *HealthRecSys@RecSys*.

Leiva, L., M. Böhmer, S. Gehring, and A. Krüger (2012). "Back to the App: The Costs of Mobile Application Interruptions." In: *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services*. MobileHCI '12. San Francisco, California, USA: Association for Computing Machinery, pp. 291–294. ISBN: 9781450311052. DOI: 10.1145/2371574.2371617.

Locher, J. L., W. C. Yoels, D. Maurer, and J. van Ells (2005). "Comfort Foods: An Exploratory Journey Into The Social and Emotional Significance of Food." In: *Food and Foodways* 13.4, pp. 273–297. DOI: 10.1080/07409710500334509. eprint: https://doi.org/10.1080/07409710500334509.

Meinel, M. (2019). "Development of a chatbot for supporting healthy dietary choices." unpublished guided research supervised by PD Dr. Georg Groh, advised by Monika Wintergerst. Technical University of Munich, Munich, Germany.

Melillo, P., M. Bracale, and L. Pecchia (2011). "Nonlinear Heart Rate Variability features for real-life stress detection. Case study: students under stress due to university examination." In: *BioMed Eng OnLine* 10.96. DOI: 10.1186/1475-925X-10-96.

Mental Health Foundation (2018). *Mental health statistics: stress*. URL: https://www.mentalhealth.org.uk/statistics/mental-health-statistics-stress (visited on 04/09/2020).

Ozier, A. D., O. W. Kendrick, L. L. Knol, J. D. Leeper, M. Perko, and J. Burnham (2007). "The Eating and Appraisal Due to Emotions and Stress (EADES) Questionnaire: Development and Validation." In: *Journal of the American Dietetic Asociation* 107.4, pp. 619–628. DOI: 10.1016/j.jada.2007.01.004.

Rasa (2020). *Rasa: Open source conversational AI*. URL: https://rasa.com/ (visited on 04/15/2020).

Shum, H., X. He, and D. Li (2018). "From Eliza to XiaoIce: challenges and opportunities with social chatbots." In: *Frontiers Inf Technol Electronic Eng* 19, pp. 10–26. DOI: 10.1631/FITEE.1700826.

Spence, C. (2017). "Comfort food: A review." In: *International Journal of Gastronomy and Food Science* 9, pp. 105–109. ISSN: 1878-450X. DOI: https://doi.org/10.1016/j.ijgfs.2017.07.001.

Sun, F.-T., C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, and M. Griss (2012). "Activity-Aware Mental Stress Detection Using Physiological Sensors." In: *Mobile Computing, Applications, and Services*. Ed. by M. Gris and G. Yang. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 282–301. ISBN: 978-3-642-29336-8.

Telegram (2020). *Telegram Bot API*. URL: https://core.telegram.org/bots/api (visited on 04/15/2020).

Tomiyama, A. J., M. F. Dallman, and E. S. Epel (2011). "Comfort food is comforting to those most stressed: Evidence of the chronic stress response network in high stress women." In: *Psychoneuroendocrinology* 36.10, pp. 1513–1519. ISSN: 0306-4530. DOI: https://doi.org/10.1016/j.psyneuen.2011.04.005.

Torres, S. J. and C. A. Nowson (2007). "Relationship between stress, eating behavior, and obesity." In: *Nutrition* 23.11, pp. 887–894. ISSN: 0899-9007. DOI: https://doi.org/10.1016/j.nut.2007.08.008.

Turing, A. M. (1950). "Computing Machinery and Intelligence." In: *Mind* 59.236, pp. 433–460. ISSN: 00264423, 14602113.

Wikipedia (2019). *Oizys*. URL: https://en.wikipedia.org/wiki/Oizys (visited on 04/26/2020).

Xu, K. and M. Lombard (2017). "Persuasive computing: Feeling peer pressure from multiple computer agents." In: *Computers in Human Behavior* 74, pp. 152–162. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2017.04.043.

Zhai, J. and A. Barreto (2006). "Stress Detection in Computer Users Based on Digital Signal Processing of Noninvasive Physiological Variables." In: *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1355–1358.