This is the Project Title

Your Name

MInf Project (Part 1) Report
Master of Informatics
School of Informatics
University of Edinburgh

2018

Abstract

This is an example of `infthesis` style. The file `skeleton.tex` generates this document and can be used to get a "skeleton" for your thesis. The abstract should summarise your report and fit in the space on the first page. You may, of course, use any other software to write your report, as long as you follow the same style. That means: producing a title page as given here, and including a table of contents and bibliography.

Acknowledgements

Acknowledgements go here.

Table of Contents

Chapter 1

Introduction

The document structure should include:

- The title page in the format used above.
- An optional acknowledgements page.
- The table of contents.
- The report text divided into chapters as appropriate.
- The bibliography.

Commands for generating the title page appear in the skeleton file and are self explanatory. The file also includes commands to choose your report type (project report, thesis or dissertation) and degree. These will be placed in the appropriate place in the title page.

The default behaviour of the documentclass is to produce documents typeset in 12 point. Regardless of the formatting system you use, it is recommended that you submit your thesis printed (or copied) double sided.

The report should be printed single-spaced. It should be 30 to 60 pages long, and preferably no shorter than 20 pages. Appendices are in addition to this and you should place detail here which may be too much or not strictly necessary when reading the relevant section.

1.1 Using Sections

Divide your chapters into sub-parts as appropriate.

1.2 Citations

Note that citations can be generated using `BibTeX` or by using the `thebibliography` environment. This makes sure that the table of contents includes an entry for the bibliography. Of course you may use any other method as well.

1.3 Options

There are various documentclass options, see the documentation. Here we are using an option (`bsc` or `minf`) to choose the degree type, plus:

- `frontabs` (recommended) to put the abstract on the front page;
- `twoside` (recommended) to format for two-sided printing, with each chapter starting on a right-hand page;
- `singlespacing` (required) for single-spaced formating; and
- `parskip` (a matter of taste) which alters the paragraph formatting so that paragraphs are separated by a vertical space, and there is no indentation at the start of each paragraph.

Chapter 2

Introduction

The *Chatting about data* project aims to explore the potentials of using chatbot technologies to empower users by increasing their understanding of their own health through data. The project is split between different areas of human health: sleep, mental health, and nutrition. The latter provides an interesting variety of challenges that have been explored in academic and clinical trials, but never for the purpose of a comprehensive, general purpose consumer application. This might be an explanation of why its still not common to discuss your dietary habits with your favourite personal assistant, regardless of how ubiquitous the technology has become in the last decade. But while the trend of consumer voice assistants has been developing products as all-round helpful tools, which do not specialise in any one task, the lack of a clear winner in the space of textual digital assistants has resulted in the proliferation of programs with smaller scope. Known as chatbots, they usually specialize in accomplishing a small set of specific tasks, akin to the app model of the early smartphone era, and are similarly distributed through a centralized platform. Thus, just like food logging, a facet of the growing quantified self movement, has exploded with the smartphone, it also falls well within the scope of existing chatbots. And just as the smartphone has not completely solved food logging, with adoption still relatively low compared to tracking other vitals and abandonment being high, the relatively unexplored area of natural language assistants in the context of *m-health* still presents several unsolved problems. A nutritional assistant must contain two essential features: the storage of foods consumed by the user, and the retrieval of such information for the purposes of informing dietary choices. There is a large variation in how the two are accomplished among various existing solutions, and the aim of our project is proposing an implementation that, using a Natural Language interface and the ubiquity and capabilities of modern messaging platform, makes interactions easier and more engaging.
Our contributions to the field are a first integration of a chatbot for text and picture food logging, and a comparative experiment between the chatbot and traditional diet tracking app on a similar demographic.

Chapter 3

Background

3.1 The chatbot revolution

Much has been said about how the rapid reduction in cost of the semiconductor has, in the last 60 years, changed the world in a significant way. The rapid spread of inexpensive and energy efficient computers, networks, and storage facilities, has revolutionised how we access information, exchange goods and services, and communicate with one another. The diffusion of the smartphone, specifically, has brought forth an explosion in the amount of information generated globally, with more than 4.9 billions users, with adults in the United Kingdom spending an average of 2 hours per day interacting with their phones, browsing the web, using applications, generating tracking data, and chatting. The latter usage in particular is one of the most popular, with 42% of mobile users [18] being active social media users. The top downloaded messaging applications, as of January 2018, are Messenger and Whatsapp (1.3 billion active users each) and Wechat (980 million) [17].
Besides keeping up with friends and professional contacts, business transactions are also conducted through chat, either arranging sale and delivery of goods, or for customer assistance, the former being much more prevalent in Asia, where most medium and large size companies, as well as some smaller ones, have a WeChat presence (conversational commerce). Increasingly, many of these transactions are being automated through the deployment of chatbots (bots), an evolution of classic conversational interfaces that have become popular in the last decade for commercial and entertainment applications [28]. The most popular bot platform outside China is Facebook Messenger, which introduced the functionality to developers in April 2016[12], and it has since taken off with more than 200,000 bots on the platform as of December 2017[11]. The development of Voice Assistants such as Google Assistant, Siri, Cortana, Alexa, or open-sourced Mycroft, have also pushed the deployment of conversational interfaces, and to some extent through the chat medium, with all the mentioned agents providing some form of textual input. Besides the marketing pushes, the use of chatbots has increased thanks to advances in Natural Language Processing (NLP) and Natural Language Understanding (NLU).
While early chatbot implementations relied on simple pattern matching rules based

on recognition of specific words (entity recognition) or parts-of-speech (POS), most of today's chatbot frameworks can leverage large corpora to apply machine learning algorithm, such as Intent analysis. Conversation can follow a slot or flow model: the latter is a hardcoded scripted flow diagram that guides the user through a preset conversation; the former specifies "slots" that contain some data the developer is interested in, and the chatbot will use NLP techniques to fill the slots from conversations with the user. Responses are typically pre-written and matched to an intent, but advances in deep learning are opening up possibilities for generative models, which create the answer from scratch[34]. Particularly successful can be combinations of several approaches, such as Serban, 2017's use of reinforcement learning to combine the approach of a generative deep learning model and a template-based retrieval model[50]. Critical to the success of the chatbot is a good context management system, to ensure that a multi-turn conversation isn't disjointed and that previously entered information remains available to the chatbot. All of this functionality is implemented by a growing variety of open source and commercial tools available today [36]
From a service provider's perspective, the advantage of using a chatbot instead of a human to provide customer service or present content is clear: over the long term, the cost of development is small compared to the number of salaries that would have to be paid to maintain the same amount of concurrent conversations. The centralization of services under a single interface to same extents also addresses the phenomena of "app fatigue": smartphone owners are no longer installing new apps, and when they do retention rates are abysmal [3]. Users' main motivations is that using chatbots makes them more productive compared to going through an app or long webpage to find information, as well as the possibility to customize the reply based on their own interest. [24]. This might be a symptom of increasingly shorter attention spans in younger generations[56], which also explain why a synchronous form of communication such as chat might be perceived more productive than an asynchronous medium such as email, and is reflected in users' preference on the chatbot's personality [40]. To a lesser extent, people also use chatbots for entertainment value and because they benefit from the social aspect, but some of the interest might only be attributed to the novelty value.
Given the need of chatbots to be used productively, user needs will cause significant consequences for the field of Human-Computer Interaction (HCI): new paradigms of interface design will have to be invented, and novel approaches to combine different types of outputs will be possible [32]. Undoubtedly, it provides a great advantage to the less tech-savy, who might have trouble understanding the user interfaces of many bespoke applications, but will already be familiar to the "unifying" chat window, to the point that, if the chatbot application is aware of enough information about its users, it might be able to create personalised interaction taylored to their preferences, and by doing that alleviate the growing digital divide that some segments of the population experience because of the implicit biases of some tech workers [**?**].

3.2 Advances in mHealth

Since the earliest days of chatbots, such as Eliza, which was modelled after a Rogerian psychotherapist [55], it's been clear that conversational agent can have significant impact on health. Its application however remained limited, for once because of the restriction at the time of having to use a teletype, but mostly because the still primitive state of the technology made freeform chatting, and crucially the ability to understand anything about your mental health, impossible. More recently, the development of cheap sensors and widespread connectivity through smartphones has spurned a growing sector of m-health applications. From the 2000s, mobile phone use made patient - doctor communication quicker and more frequent, as well as enabling some initial forms of monitoring and providing a hub for Body Area Network including different medical sensors [45]. The 2010s have seen further developments along these uses, as well as the advance of "Artificial Intelligent" applications, provided by vast quantities of data collected through mobile devices, as well as advances of other Computer Science and Engineering disciplines such as image recognition, virtual reality, robotics, drones, 3D-printing, and the Internet of Things (IoT) being applied to the medical field [46]. The vast quantities of data being collected have helped to advance the state of the art on several medicinal application - but they also provide a valuable monetisable resource, both from a perspective of Big Data analysis, and for the number of for-profit advertising companies that will sell medical information to marketers and insurance companies [53], as well as hackers for resale on the black market and for purposes of identity theft [57]. The sensitive nature of this kind of information makes it difficult to operate without breaking patient confidentiality, since even if anonymised medical records can be re-identified by correlating with outside sources [52], and even large experienced medical institution and data collection companies can breach existing regulations [15].
Smartphone and wearable devices users have also been collecting their own personal information, giving birth to the idea of the Quantified Self, or Personal Informatics., Much of the Quantified Self movement is based on the idea that the automation of data gathering will lead into greater insight and improve our own health and behaviours by reflecting and evaluating our past experiences. Rivera and Pelayo, 2012 [49] propose a framework necessary for self tracking app, based on the three activities of tracking cues, triggering and recall. Tracking can be done through software logging or hardware sensors. Triggering can be active, when the user is prompted to reflect in a suitable context, or passive, where the information is simply displayed in a location the user can observe and notice significant changes. Recalling can be aided through different techniques, which usually involve a considerable amount of post-processing and enhanced by access to large datasets. There is still much work that could be done in the area of contextualization, by associating the collected data with other sources, and data fusion, comparing your own data with independent self or peer reporting, as well as better data visualization in attractive and intuitive ways.
One example where the quantified self phenomenon can have a real impact is preventive medicine, promoting healthy lifestyle to alleviate future medical issues. Diet, in particular, has been shown to benefit from open ended food-logging more than other methodologies [23]. While Turner-McGrievy, 2013 [**?**] found little advantage

in replacing a paper diet tracking system with a mobile application, Personal Activity trackers in their study did receive an advantage; this leads to speculation that the lack of improvement in the first group might have been caused by the diet tracker's UX; and in fact, using the My Meal Mate app over a 6-month trial, Carter 2013, [**?**] reported increased adherence, usage, convenience, social usability, and overall satisfaction compared to traditional diet tracking.
Good examples of currently active commercial quantified self apps for fitness and nutrition are *My Fitness Pal*[10] and Google Fit[**?**], whose designs have been shown [51] to prioritize continuance intention (the willingness to continue using the app), usability qualities such as directness, informativeness, learnability, efficiency and simplicity; user value features such as satisfaction, customer need, attachment, pleasure and sociability. The usability of these interfaces, which include both a textual input and a barcode scanner for commercial products, has increased the number of DIY food loggers[21]; however, there is still some friction to a seamless logging experience[**?**], which might be bridged by the use of a chatbot interface.
Chatbots have been speculated to provide a useful tool as a behavioural intervention technology, used to complement human practitioners in reaching a larger number of users and automate personalised messages [33]. Experimental and clinical trials using simpler informational chatbots have been made in various medical fields, such as counselling [25], mental health intervention [31] and sexual health information [**?**], generally providing positive results, or at least giving indication that the medium might be used to address the specific issues. Among the more successful experiments in Behavioural Health Intervention, the MobileCoach open source platform [13] was originally designed as a text messaging based system, which did some parsing on the backend but mostly relied on practitioner's interventions. It has now been redesigned as a full fledged online chat platform, which was perceived positively in clinical trials where participants treated for obesity interacted with a chatbot that exhibited a distinct personality [38].
One successful attempt at making a dietary tracking chatbot was Forksy [9], but there are no statistics on its usage, and development seems to have stopped. Forksy is very effective on getting users to log their diet, but seems to have no "smart" features.

3.3 Making a smart chatbot application

As researchers in many other fields in Computer Science are realising in the last decade, the collection of large quantities of data can have other uses besides record keeping, by leveraging the booming fields of machine learning and data science, and could eventually lead to make a truly useful chatbot to replace or complement professional dietitians in a larger measure.
Ever since Richards, 1902 [48], attempts have been made to algorithmically use food composition values to maximise food value per money spent. However, as Richards herself notes, "we know too little of the effect on digestibility, of cooking, and of the combination of two or more foods in one dish, or at one meal, to permit of very close calculation".
Even a century later, nutritional science still struggles to establish criteria to categorise

any one food as “good” or “bad”, because of the large number of nutrients that make up each food [54]
To achieve a truly smart dietary assistant, we should be able, given a vast amount of information about our users, their habits and goals, to recommend an effective strategy to achieve the latter by analysing their choices in the former. As Gregori, 2017 [34] describes, the architecture of a chatbot requires four components: a frontend, a knowledge base, a backend and a corpus. While there are many tools that can be used for chatbot frontend and backend, finding an appropriate corpus and generating a knowledge base are domain specific tasks, with far less options available.
Even for a restricted dietary task such as reducing fats consumption, expert systems will be based off a set of handcrafted rules [47]. While the medical community has made efforts to solidify their field into knowledge bases, there are no prevailing standards to read and interpret them, and although some efforts have been made to use knowledge graph representation to power a symptoms identifier chatbot [**?**], there doesn’t seem to be a canonical dietary knowledge base. Current commercial apps use a combined approach of total calorie counts and macro/micro nutrient percentages, but this approach is often insufficient to initiate healthy behaviours [**?**]. Despite the criticism for the occasional sensationalism, the emerging field of dietary epidemiology advocates a holistic approach to nutrition studies, by taking into account genetic, lifestyle and metabolic information as much as dietary records, making the mere tracking in sufficient to draw anything but the most casual inferences on the users’ health [**?**]. But until this branch of the field develops enough to provide us with effective personalised nutrition (some recent startups would like us to believe that’s already the case [16]), it’s possible to use a more restricted approach based on recognising unbalanced diets from the lack or excess of certain key nutrients, abstracting the mechanics of quantifying exact measures from the users by providing more immediate advice through food recommendation. Data analysis techniques on food composition can be used to draw networks of complementary foods (foods that together fulfil nutritional needs) [37], which could be used to give suggestions based on what users have already eaten. There are plenty of choices for nutritional value composition datasets[], and free or commercial APIs []
Success in activity tracking is influenced by demographics, with older and lower income subjects having lower rates of initial activation and retention [44]. This problem may be caused by the bespoke user interface each fitness tracker comes with, a problem which might be alleviated by using a universal chat interface.
Popular fitness tracking apps often providing social networking functionalities, which have helped participants achieve their fitness goal through a combination of competition with their peers and social accountability [26]. Gamification has also proven useful [42], and so have financial involvement, but only when profiled as a loss and not for modest gains [41]. One company who successfully integrated diet tracking with monetary incentives and social accountability was Gym-Pact (later Pact app), which rewarded users for tracking their calories, eating enough fruits and vegetables, and exercising, but took money from them if they didn’t. The app reached a sufficiently large number of participants [39] to sustain itself for several years, and a high percentage of users was frequently able to achieve their goals (but their business model wasn’t profitable enough, and they closed in Summer 2017).

3.4 Image recognition

Most messaging apps today come with media functionality integrated; in particular, it's easy to take pictures and send them as a message from within the app itself. A diet tracking chatbot might benefit from the users' ability to take pictures and instantaneously receive feedback on their nutritional value. While this as to the best of our knowledge never been attempted within a chatbot interface, photographic diet diary have complemented food logging for many years, both in a traditional paper form to aid recollection[], and more recently electronically. Classically, portion size estimation required placing a fiducial marker, an object with a distinctively recognisable pattern, in the frame of the image, as to be able to fit a geometrical model on the entire picture [19]. A slight twist on this has been spun by Smartplate, a startup that uses a distinctively shaped plate to implement image based food tracking []
A different approach was used by Google research with the Im2Calories Android app [43]. Besides using a convolutional network based off a newly collected MultiLabel dataset to classify what the food in the picture is, different CNNs are also used to segment images and to estimate their 3D volumetry. This allows the app to assign calorie counts to images that contain different foods in the same plate, and to have more precise estimation of size. Unfortunately, neither the app nor the datasets have been publicly released. More recently, small startups like Calorie Mama[5] and Bitesnap[4], as well as Samsung's digital assistant Bixby[**?**] have also implemented similar functionality, although it's still not clear how their models were trained, or how effective they are.
Among social media users, especially on the Instagram photo sharing platform, it's common to photograph images of aesthetically pleasing food. While this does by no mean provide an exhaustive nutritional history, it can be used as a further automation to save users from having to manually log their meals and extract nutritional information, as well as potentially another avenue to establish social accountability to log healthy food [**?**]. We will still have to use computer vision algorithms on this data, because Instagram tags are unreliable in identifying the content of the picture due to the large number of slang-related false positives [**?**].

Chapter 4

Implementation

4.1 Design

As a dialogue agent, a chatbot needs to be designed with conversation between agent and user in mind, with each turn in the exchange serving a function of supplying or retrieving information from the system. To determine what kind of interactions the users and the chatbot should have, we started our design phase using the Botmock tool to draft some conversation graphs. One consideration when determining the tonality of our conversation sketches was determining what personality the chatbot should exhibit. Since chatbots users have shown to appreciate the fun aspect of designed chatbot personalities [40], we tried to imbue our conversations with some colloquial aspects, like using slang, calling the user names, and using emojis.
To establish a level of familiarity, rather than fetching a user's name from the client's account information, we initiate our first conversation by asking for it. The rest of the introductory messages just describe the chatbot's functionalities. Besides asking for help, users have three main available actions they can take: tell the chatbot what they are eating, send a picture of food, and ask the chatbot to recap what they had on a specific date. If the user doesn't specify a quantity or measurement to their meal, the chatbot will ask for details. To avoid bogging down the user with measuring their portions, and since we mostly want them to think about their diet in general terms, we encourage to use relative rather than absolute quantities (more, less or same as usual). We hope that using these measurements we will be able to maintain longer term user engagement, without sacrificing understanding of how much food is being consumed. Once we have this information, we send it through the Nutritionix API, to find out the nutritional content of that meal.
Sending a picture to the chatbot triggers a call to the *clarifai* vision API, to understand what the contents of the image is, using their food model. *clarifai* returns a list of guesses for a picture, and their confidence value. If only one guess has confidence above the arbitrary threshold of 97%, we save the food in the picture in our database; in any other case, we present the user with a list of the top three guesses as interactive buttons, so we can store the correct value in the database.

4.1.1 Nutritional analysis

Once the user starts logging details about their food consumption, we will need to start analysing what they are eating to give them advice. Lacking comprehensive nutritional knowledge, the best we can do is crafting some elementary heuristics.
The food retrieval intent queries the chatbot's database for all foods stored on a certain date. To aid user's reflection, we provide some basic data analysis, specifically on the food's quantity: if there are more than 10 foods logged, with at least one having been consumed in large quantities and either less than two thirds consumed in small quantities or more than two thirds consumed in large quantities, we warn the user they might be overeating. Conversely, if they have logged less than 3 items of food, or if more than $2/3$ of their meals come in a small quantity, we warn the user about undereating. If they are asking about the current date, they might just not have had time to log all their food; in that case, we only send a warning if the current time is past 10 PM.
One example of food category that is often praised by nutritionists for its high nutritional value [] is leafy green vegetables. To demonstrate the capabilities of chatbots as nutritional advisers, we handcrafted a list of leafy greens, which we check every meal for. If we don't observe the user eating any of these foods in 2 days, we prod the user with a reminder. Prodding in a chatbot context requires finding a very delicate balance: it can be too infrequent, making it less useful for users who actually want to be reminded; but if too frequent, it might soon become annoying. For example, the Forksy nutritional chatbot [] sends a reminder to log a meal for every single one (at least 3 a day). This causes the bot to feel overbearing, and it might actually drive the user away (Forksy actually implements an interesting solution by asking how long before the next reminder, but it still gives no option to deactivate reminders completely). Our solution is staggering the no usage messages at increasing intervals, first after day one, then on day two, four, seven and fifteen, to maximise our chances of getting a forgetful user to start messaging again, without being too annoying if they choose not to.

Architecture

Our chatbot's architecture is composed of a natural language interface in Google Dialogflow, hooking up to an Express.js server running a Heroku instance, storing data on a hosted MongoDB, and retrieving other information from external APIs. The agent itself is presented through a Facebook Messenger bot, which we chose for its popularity.

Dialogflow concepts

Today there are several Natural Language Understanding platform specifically designed to create chatbots. Our choice fell on *Google Dialogflow* (at the time *API.AI*), because of its ease of integrations with most popular messaging platform, ease of development and solid NLP functionality. A Dialogflow agent is set up with a library of patterns, *intents*, that use example sentences or templates to parse inputs to the chatbot. Templates include parameters whose types are called *entities*, some of which are already defined by Dialogflow, but new ones can also be added by the programmer. One of the standout capabilities of Dialogflow is that a parameter can be marked as required by the intent, so if the initial utterance does not contain it, the chatbot will prompt the user to specify the new parameter. To maintain the flow of conversation, an intent can be followed by another, based on what the user replies, with a context object which stores all parameters passed from the original intent down to all its followups. Each intent can trigger an immediate response, as defined on the Dialogflow console, or it can trigger an *Action* to be fulfilled by the *Webhook*. *Actions* are functions held on the server that can access the parameters of the current intent, execute their own logic, call external APIs etc. They usually are triggered as a POST request to the webhook's address, with the request JSON object as its body, and the HTTP response determines the behaviour of the chatbot, either by replying with more text, sending media or rich text, or triggering an *event*, which executes an intent just as one of the triggering sentences had been sent by the user. Small talk intents are also defined as part of Dialogflow to take care of common sentences unrelated to the chatbot's domain, and Dialogflow's machine learning engine can be interacted with through a Training console (still in Beta at the time of writing), which can be used to correct intent and parameter recognition (and indirectly, to gain insight into how the engine's model classifies new text).

Our chatbot defines the following intents:

- *First interaction* is the initial intent, triggered by the Facebook Messenger ”Get Started” button or with the ”Greetings” keyword for debugging purposes. This asks the user what their name is, and is followed up by

  Name save which waits for a name to be given, and saves it to the database

  Name confirmed replies with a welcome explanation message

- *Food log - text* is triggered when the user adds a meal to the log. It takes required parameters of food (as a list), quantities, and optional parameters of meal name and date

- *Send food pic* waits for a Facebook Messenger Media object (a picture in our case) to trigger an action which runs image recognition. If no unique match is found,

  Send food pic - no is triggered by the webhook whenever there is no clear candidate for classification of a picture

  Clarify food pic recognises user's sending a clarification for what the food was through a Facebook Messenger button. To make sure the button's message matches this intent and not the generic food logging, a messenger_button token is appended to the message payload

  Send food pic - yes is launched whenever a picture is classified correctly, and just triggers the Analyse food pic intent

  Analyse food pic takes the food content found in a picture received from the user, analyses and stores the result into a database

- *Help* matches a request for help with a few reminders of the chatbot's functionalities
- *Date retrieval* is used to query for past food logs, taking just a date as a parameter. For logging purposes, with removal functionality to be implemented in the future,

  Date retrieval - false will recognise when the user declares the food log for that day to be incorrect
- *What's my name* is mainly a debugging intent, to check if the chatbot has managed to successfully save the name for the current user in the database
- *Sinkhole* is used for training purposes to redirect all the intents that were misclassified

Our chatbot defines the following entities:

- *meal* enumerates four different meals: breakfast, lunch, dinner and snacks
- *quantity* contains a list of all the different measurement units for food
- *meal-quantities* combines quantities with quantifiers (a, some, integers etc.)
- *approximate-quantifier*
- *date-ext* extends the @sys.date object with today and tomorrow

Backend

By default, the Dialogflow interface includes a small inline editor to implement some simple webhook logic. While the web interface is limiting for creating a backend of the complexity required, it's easy to export this example code to Google Cloud Functions [6], Google's serverless cloud function service, and their own database, storage and analytics tool, Firebase. The sample code consists of an Express.js [8] web server, which listens to POST requests to the */webhook* route to the server, which will be sent from Dialogflow, and provides helper functions to craft the appropriate response as a JSON object, which will trigger a reply through the chat client. The largest portion of the code is the action-handlers dictionary, which associates a different function to any of the Dialogflow intents.

We started developing our webhook from this starting example in GCF, but we soon realised that a core requirement of our design, the ability to call up external APIs, was not possible under the free tier of Google Cloud. Thus, it was necessary to find a replacement. Some options that were considered were Apache OpenWhisk [2], Captain Duck Duck [22], Amazon Web Service Lambda or Elastic Beanstalk [**?**], Dokku [7], 1backend [1]. In the end, Heroku was selected as a solution because of its many productivity advantages. Among the many alternative the popular Platform as a Service (PaaS) solutions offer, we took into consideration the mature tooling, the easy to use deployment infrastructure, which consist of simply pushing the code to a version controlled repository, the automatic inclusion of a free domain name, and simple (but barebone) scheduling functionality. Heroku offers both a free and a paid tier; for our purposes of creating a prototype, the free tier offers all required functionality; however, it would not be sufficient to power the chatbot infrastructure in a production setting, as there are limits to free users' capabilities, most notably a temporary suspension (and prolonged wake up times) after an hour of inactivity. Having moved to a full PaaS implementation, we were able to expand our program from a single file to several modules, which was necessary to avoid a large unwieldy single file, and allows us to compartmentalize between different types of functionality.

- *index.js* is the main function for the Express server, it runs in a loop waiting to receive any requests, and serves a response for any predefined URLs. On a POST request to */webhook*, it will read the request body to find the action's name and parameters, and calls a function as defined in the *actionHandlers* dictionary. Most intents match one-to-one to an inline function in this object, although some are defined in an external module
- *messenger.js* contains output functions to send a reply to the user, either going through the Dialogflow API, or directly through the user's Facebook Messenger account (necessary for sending a message on a predetermined schedule, without a user initiating the conversation)
- *picture.js* handles all intents related to pictures, from querying the *clarifai* to handling incorrect or imprecise guesses.
- *mongo.js* abstracts some of the low level database, like connecting to the database and querying the username
- *analysis.js* deals with all food related activity, like querying the Nutritionix API and analyse users' dietary records
- *strings.js* contains all messages the chatbot will send to the user, all collected in one file for ease of editing
- *worker.js* defines a set of functions to run periodically, which will verify some information about each of the chatbot users, and send an appropriate message

The latter module, unlike all the others, doesn't export any of its functions, but it's run every day at 8 AM GMT by the Heroku scheduler add-on.

4.1.1.1 Database

We store important data from the chatbot on a MongoDB database free instance on mlab.com. The database consist of a single collection, users, with a unique document for each user. Besides containing identifying information such as a unique Dialogflow ID, a session ID to initiate messenger conversations and a name, we also store a list of meals, objects containing a list of food items, their quantities, a date and an optional meal name (lunch, dinner, breakfast or snacks). Finally, we have a counter object, which stores values for the number of days since the user has logged any food, the number of days since the user has had any green leaf vegetables, and the total number of days. These are used to determine whether a reminder should be sent to that specific user about any of those issues (the total count reminder is used to ask for feedback about our experiment after three days, to collect some data while participants are still engaged with the chatbot).

4.2 Failures

While we have developed and delivered a fully functional chatbot, there are many features that we would have liked to have but could not implement for lack of time. Other features were started, but couldn't be completed.

Instagram failure

A unique feature of our chatbot design was its integration with Instagram. If a user had an account on the social network, we would have asked them during the onboarding procedure to log into their account to generate a unique access key, which we would later have used to crawl their image history for food pictures. We developed the onboarding dialogue sequence, giving the user a choice on whether to add an account or not, and we registered an Instagram developer account and an application to generate unique login URL. If a user logged in through this link, they would have been redirected to a custom address on our server where the access token would have been saved to their account.
We completed our implementation of this first phase, and successfully tested the token generation on our own account. But as we were developing the crawler implementation, our application was suspended for Terms of Service violation. Speaking with a Facebook inc. engineer, they speculated that the reason for the ban was that the picture retrieval API is only intended for building alternative clients, making our idea infeasible. An alternative with a more permissive API is Flickr, but we didn't explore the possibility of using this smaller social network.

4.2.1 Defining a food entity

The Dialogflow predefined entities do not include one for food. This is in fact a non-trivial problem, because enumerating all possible food requires knowing about both "raw" ingredients, and commercial food which might be referred to as their brand name or with a specific product denomination. We therefore tried to handcraft our own food corpus, in the hope it would provide sufficient coverage. To collect food entities, we first used the Open Food Facts database [27]. Besides being freely accessible, this option was selected because of the large number of entries, 374259, the presence of generic food identifiers associated with commercial product for 59383 of the entries, and a nutritional approved health rating on a A to F scale. Moreover, besides a raw data export, the service provides an experimental JSON API For online queries. The raw database was exported as a MongoDB [14] object, in bson format. After having created an empty *openfooddata* table, using the *mongoimport* command we copy the contents of this object in the new database. Then, through the mongo console, we run command

```
var cursor = db.products.find(
    {$and: [
      {$or: [
            {‘‘generic_name’’ : {$ne: ””, ’$exists’:true
               ↪ }},
            {‘‘generic_name_en’’ : {$ne: ””, ’$exists’:
               ↪ true}}
         ]},
      {”countries” : {$regex: ‘‘en|UK|United States|
         ↪ Canada’’}
    ]}
)

var cursor = db.products.find({$and: [{$or: [{”” : {$ne:
   ↪ ”, ’$exists’:true}},{”generic_name_en”: {$ne: ”, ’
   ↪ $exists’:true}}]}, {”countries” : {$regex: ‘‘en|UK|
   ↪ United States|Canada’’}}]})
while (myCursor.hasNext()) {
    printjson(myCursor.next());
}
```

Unfortunately, this produced only 795 food item which were indexed as being listed in English, and had a proper name, and manual testing resulted in several misses. Subsequently, we found an online corpus of British raw food at LanguaL.org [](1316 items), but it wasn’t sufficient for common queries either.
In the end, we decided to compromise, deeming missing on a less common food being logged more harmful than false positives. Our final approach was marking the food parameters as a sys.any entity, which is equivalent to a wildcard match. This is fine for a more structured intent, where the user prefaces tier logged food with an action verb (“I ate”, “I had” etc), but these aren’t sufficient to capture the full range of inputs, as some user will also just say the name of the food. To catch those cases, we had to add an intent case which just matches sys.any, but that obviously has the unwanted consequence of also catching utterances that neither the intent nor the smalltalk module catch.

Classifying food

Just like our handcrafted leafy green vegetable set, we would have liked the capability of automatically recognise whenever food belonged to a certain food category (meat, fish, nuts etc), which we would have used to create a set of simple heuristics to detect missing categories from the user's diet.
The apparently simple task of associating the name of a food to a category is deciptively complex; even humans assign multiple categories to the same food (e.g. fruit or snack for Apples), and thinking about a certain food as belonging to a certain category will influence their beliefs in regard to its nutritional properties [35]. Our naive attempt to classify food according to its category was to cluster it based on its nutritional values: ideally, similar kind of foods would have ended up being classified in the same clusters ("high in sugar", "high in protein", "low in vitamins" etc) and manual inspection of classified data could have been used to assign an intuitive category to each cluster.
The k-means clustering is used to group points into n-dimensional space into a predetermined k clusters, by iteratively computing the cluster each point belongs to based on a distance metric, until cluster membership becomes stable. While our vector space was 250-dimensional, the number of distinct nutritional values identified by the USDA nutritional database, it's not trivial to determine the value of *k*. If we had had a distinctive number of food groups we wanted to obtain (like, for instance, the 10 categories identified by the Australian National Nutrition Survey of 1995 [**?**]), we could have used that as *k*, but obviously any kind of foods that belonged to a cluster we haven't considered would have been incorrectly classified. Napoleon, 2011 [**?**, **?**] describes an algorithm to both select a value *k*, and to reduce the dimensionality of our data set, which allows us to reduce computation by eliminating nutrients that don't contribute significantly to classification. This was necessary, as the size of our training data was too small compared to the dimension of each data point, so any unsupervised learning method would fail to give us any interesting result []
Calculated clusterings for the training set, any subsequent food the user logged would have been classified based on its distance from the calculated cluster centres. We built a training dataset by fetching from the Nutritionix API the nutritional values of the 1316 foods in the LanguaL.org dataset, minus the 213 foods Nutritionix had no nutritional value on record for. The Nutritionix API returns a list of values of type (ID, quantity), where the ID Corresponds to nutritional values as identified by USDA. We passed the data to a custom node script, using the *mljs* library [], to expand each food's value into a 250-dimensional vector, and perform dimensionality reduction using PCA[29], find good starting cluster centres, and executes k-means clustering on the entire dataset using the *clusterfck* library [**?**]. The cluster centres are then stored to a file, with the intention of them being read by the application for classifying new food.
All attempts at classification, however, where disastrous. All food seem to cluster around a small number of highly centralised centres, with most clusters receiving less than a dozen foods, a couple in the low hundreds, and the remaining thousands grouping around the same cluster. This result is corroborated by Kim, 2015 [37]'s finding that foods of similar origins do not necessary cluster together when grouping based on raw nutritional values. Their proposed network classification and metrics of nutri-

tional fitness do offer a potential method to recommend complementary meals for the chatbot, but deriving the full network from the provided dataset would have required a significant amount of time.

Ignoring nutrient counts, alternative classification methods could be usage of word embeddings in recipes https://github.com/altosaar/food2vec While this might be a good way to find if two foods are culinarily related, it does not necessarily satisfy the property of nutritional equivalence, which is what we'd ultimately like the chatbot to do. Eftimov, 2017 [30] developed an automatic classification method for European standard food classification system FoodEx2, which doesn't only list composition value, but also chemical contaminants, food consumption and pesticide residuals. Their algorithm goes through a classified learning phase and a probabilistic natural language extraction phase for description, which achieves a good accuracy of 89%, but the resulting classification categories are too broad to be useful.

Chapter 5

Evaluation

As with all software artefacts that have a user facing component, testing can be lead both on a technical level and on a usability level. Since the chatbot infrastructure contains many different components, it will be necessary to conduct some testing on the robustness of each.

5.1 Testing

Throughout the whole implementation, particular care was taken to follow software engineering best practices. Because JavaScript is a dynamically typed language, we lack a compiler's check for code correctness. As a consequence, we have to run code to find out if it works. To avoid deploying a broken commit to the server, we wrote a git pre-commit hook to test if the program doesn't crash immediately, and to pass a linter to catch any syntax error in the code:

```
npm start > /dev/null &
sleep 5
if ``eslint *.js`` && [[ -n `pidof -k node` ]] ; then
    echo "Pass linter and npm doesn't crash"
    exit 0
else
    pkill node
    exit 1
fi
```

While this was a good tool to statically catch errors, some bugs would only emerge through dynamic testing. While the testmybot unit test library looked like a promising solution for establishing a routine of test-driven development, it soon was evident that the Facebook hooks are still not mature enough to be used in production, so we had to resort to manually testing each new feature, by messaging the chatbot from a personal Facebook account, repeating the same script and adjusting the code until the issue was fixed. Most debugging information was printed to the Heroku several logs through the *console.log* JavaScript function.

5.2 Evaluation

For our evaluation, we ran an experiment giving out the chatbot to 11 university students, all within ages of 20 to 25 and at least moderately physically active, to use for a week. As a control group, another 9 university students were prescribed to use the MyFitnessPal app for the same duration. All users were recruited through Facebook chat or in person, and all were given the OK to start the evaluation on the same day. In retrospect, having a more gradual rollout might have helped with spotting the first bugs initially and giving us a chance to fix the underlying issues without compromising the platform for every other user.

5.2.1 Record keeping

Since this was our first usage of the chatbot outside our own testing, we expected to encounter a variety of bugs and phrasings that it had never encounter from us. We set up a detailed logging function for all error case, printing the user ID as well so as to be able to reconstruct the causes at a later stage. We could also access a complete record of all communication through the Facebook app console, as well as having a list of intents identified and how the parameters were matched from the Dialogflow agent. While having this much access gave us some great insight into what might be affecting faulty behaviours, it was also concerning how we could read the conversations in their entirety, and while Dialogflow allows to deactivate the logging, there was no way of doing that through Facebook. And even if there was, it would be trivially easy to still log everything through the server.

5.2.2 Experiment description - survey, in person interview

To initiate the experiment, the chatbot users' Facebook profiles were added as testers through the Facebook developer console. They were then sent a link to the chatbot's Facebook page, where they could press a clearly visible button to start chatting. This would open a chat window, where they had the option of pressing a button to get started before being taken through their first conversation. Users were given no indication on how to procede, except for the chatbot's introductory message. Over the course of the evaluation, users sent us some questions (never through the chatbot) on what they could do with it. The MyFitnessPal testers were asked to give feedback a week after the evaluation started; the bot testers were sent feedback forms after 9 days.

Chapter 6

Discussion

6.1 Chatbot performance

The nature of chatbot development is one of predicting what a user will say; this is a hard job for any large software development effort, and even harder for a single student. Testing on our own account, using phrasings we knew the chatbot would recognise, was very different from deploying it in the wild, and experiencing all the different ways participants in the study tried to express the same concepts.

* sys.any didn't work * circular conversations * reminder failure * add testers * size recognition doesnt work * multiple foods broken down by sizes and messages * food api not working well (Long island ice tea)

During user evaluation, features were pointed out: clear record if incorrect food, add food on different date; users wanted more frequent reminders; didn't realise pictures capability, or lookback on food; reminders didn't work for the first 6 days, but when they were turned on 100% of users went back in the same morning, and most sent a message after as well; bad interaction with follow ups

Features we want: recurring meals, GIFS, challenges

Through the implementation, it became evident how little privacy a chatbot user can expect. No e2e encryption, data shared among chat provider, nlp tools, databases, cloud hosting and analytics

Chapter 7

Conclusion

Security of chatbots is bleak but important: medical information need to be GDPR and HIPAA compliant, but no major chatbot provider bot offers a chatbot API and e2e encryption [20]

Our hackish heuristic should be tested out over a longer period, or they should receive a more systematic approach

Bibliography

[1] 1backend.

[2] Apache OpenWhisk. URL: https://openwhisk.apache.org/.

[3] App Fatigue Allowed Chatbots To Emerge for Customer Service Engagement. URL: https://chatbotsmagazine.com/app-fatigue-allowed-chatbots-to-emerge-for-customer-service-engagement-e8e

[4] Bitesnap - Photo Food Journal. URL: https://getbitesnap.com/.

[5] Calorie Mama AI - Food Image Recognition and Calorie Counter using Deep Learning. URL: http://www.caloriemama.ai/.

[6] Cloud Functions - Event-driven Serverless Computing — Google Cloud Platform. URL: https://cloud.google.com/functions/.

[7] Dokku. URL: https://github.com/dokku/dokku.

[8] Express - Node.js web application framework. URL: https://expressjs.com/.

[9] Forksy a robot-nutritionist that helps increase your energy. URL: https://getforksy.com/.

[10] Free Calorie Counter, Diet & Exercise Journal — MyFitnessPal.com. URL: https://www.myfitnesspal.com/.

[11] Messenger Platform 2017 — The Evolution of Why We Message. URL: https://messenger.fb.com/blog/a-look-back-on-messenger-platform-in-2017.

[12] Messenger Platform at F8 — Facebook Newsroom. URL: https://newsroom.fb.com/news/2016/04/messenger-platform-at-f8/.

[13] MobileCoach. URL: https://www.mobile-coach.eu/.

[14] MongoDB. URL: https://www.mongodb.com/.

[15] NHS patients' data was illegally transferred to Google DeepMind. URL: https://news.sky.com/story/nhs-patient-data-given-to-google-illegally-10935315.

[16] Personalized Nutrition Designed for Better Health & Weight Loss Habit. URL: https://habit.com/.

[17] Most popular messaging apps 2018 — Statista. URL: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.

[18] Social media: worldwide penetration rate 2018 — Statistic. URL: https://www.statista.com/statistics/269615/social-network-penetration-by-region/.

[19] Ziad Ahmad, Marc Bosch, Nitin Khanna, Deborah A. Kerr, Carol J. Boushey, Fengqing Zhu, and Edward J. Delp. A Mobile Food Record For Integrated Dietary Assessment. *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management - MADiMa '16*, pages 53–62, 2016. URL: http://dl.acm.org/citation.cfm?doid=2986035.2986038, doi:10.1145/2986035.2986038.

[20] A Alesanco, J Sancho, Y Gilaberte, E Abarca, and J Garc. EMBEC & NBC 2017. 65:185–188, 2018. URL: http://link.springer.com/10.1007/978-981-10-5122-7, doi:10.1007/978-981-10-5122-7.

[21] Emily Alonso. How This Winner-Took-All by Helping People Lose. 2015.

[22] Kasra Bigdeli. Captain Duck Duck.

[23] S. A. Bingham, C. Gill, A. Welch, K. Day, A. Cassidy, K. T. Khaw, M. J. Sneyd, T. J. A. Key, L. Roe, and N. E. Day. Comparison of dietary assessment methods in nutritional epidemiology: weighed records v. 24 h recalls, food-frequency questionnaires and estimated-diet records. *British Journal of Nutrition*, 72(04):619, oct 1994. URL: http://www.journals.cambridge.org/abstract{_}S0007114594001650, doi:10.1079/BJN19940064.

[24] Petter Bae Brandtzaeg and Asbjørn Følstad. Why People Use Chatbots. In Ioannis Kompatsiaris, Jonathan Cave, Anna Satsiou, Georg Carle, Antonella Passani, Efstratios Kontopoulos, Sotiris Diplaris, and Donald McMillan, editors, *Internet Science*, pages 377–392, Cham, 2017. Springer International Publishing.

[25] Gillian Cameron, David Cameron, Gavin Megaw, Raymond Bond, Maurice Mulvenna, Siobhan O ' Neill, Cherie Armour, and Michael Mctear. Towards a chatbot for digital counselling. *Mental Health Artificial Intelligence*, pages 1–7. URL: http://hci2017.bcs.org/wp-content/uploads/BHCI{_}2017{_}paper{_}110.pdf.

[26] Yu Chen and Yu Chen. Exploring social accountability for pervasive fitness apps Exploring Social Accountability for Pervasive Fitness Apps. (January), 2014.

[27] Collaborative. Open Food Facts, 2015. URL: https://world.openfoodfacts.org/.

[28] Robert Dale. The return of the chatbots. *Natural Language Engineering*, 22(5):811–817, 2016. arXiv:0910.0834, doi:10.1017/S1351324916000243.

[29] C Ding and X He. K-means clustering via principal component analysis. *Proceedings of the twenty-first international conference on Machine learning*, Cl(2000):29, 2004. URL: `papers://b6c7d293-c492-48a4-91d5-8fae456be1fa/Paper/p2738{\%}5Cnfile:///C:/Users/Serguei/OneDrive/Documents/Papers/K-meansclusteringviaprincipalcomponent.pdf`, arXiv:arXiv:0711.0189v1, doi:10.1145/1015330.1015408.

[30] Tome Eftimov, Peter Korošec, and Barbara Koroušić Seljak. Standfood: Standardization of foods using a semi-automatic system for classifying and describing foods according to FoodEx2. *Nutrients*, 9(6), 2017. doi:10.3390/nu9060542.

[31] Danielle Elmasri and Anthony Maeder. Brain Informatics. 7670:243–251, 2012. URL: `http://link.springer.com/10.1007/978-3-642-35139-6`, doi:10.1007/978-3-642-35139-6.

[32] Asbjørn Følstad and Petter Bae Brandtzæg. Chatbots and the new world of HCI. *Interactions*, 24(4):38–42, 2017. URL: `http://dl.acm.org/citation.cfm?doid=3115390.3085558`, doi:10.1145/3085558.

[33] Silvia Gabrielli, F B K Create-net, and Silvia Gabrielli. Addressing Challenges in Promoting Healthy Lifestyles : The AI-Chatbot Approach Addressing Challenges in Promoting Healthy Lifestyles : The AI-Chatbot Approach. (July), 2017.

[34] Eric Gregori. Evaluation of Modern Tools for an OMSCS Advisor Chatbot. *School of Computer Science Graduate Student Publications, Georgia Tech*, 2017. URL: `https://smartech.gatech.edu/bitstream/handle/1853/58516/evaluation{\_}of{\_}modern{\_}tools{\_}for{\_}an{\_}omscs{\_}advisor{\_}chatbot{\%}281{\%}29.pdf?sequence=1{\&}isAllowed=y`.

[35] Brett K. Hayes, Hendy Kurniawan, and Ben R. Newell. Rich in vitamin C or just a convenient snack? Multiple-category reasoning with cross-classified foods. *Memory and Cognition*, 39(1):92–106, 2011. doi:10.3758/s13421-010-0022-7.

[36] Javier Couto. Building a Chatbot: analysis & limitations of modern platforms - Tryolabs Blog, 2017. URL: `https://tryolabs.com/blog/2017/01/25/building-a-chatbot-analysis--limitations-of-modern-platforms/`.

[37] Seunghyeon Kim, Jaeyun Sung, Mathias Foo, Yong Su Jin, and Pan Jun Kim. Uncovering the nutritional landscape of food. *PLoS ONE*, 10(3):1–17, 2015. doi:10.1371/journal.pone.0118697.

[38] Tobias Kowatsch, Dirk Volland, Iris Shih, R Dominik, K Florian, Filipe Barata, Andreas Filler, B Dirk, Katrin Heldt, and Pauline Gindrat. Designing the Digital Transformation. 10243:485–489, 2017. URL: `http://link.springer.com/10.1007/978-3-319-59144-5`, doi:10.1007/978-3-319-59144-5.

[39] Kim Ly, Min Zhao, and Dilip Soman. Nudging 15. 2013.

[40] Indrani Medhi Thies, Nandita Menon, Sneha Magapu, Manisha Subramony, and Jacki O'Neill. How Do You Want Your Chatbot? An Exploratory Wizard-of-Oz Study with Young, Urban Indians. In Regina Bernhaupt, Girish Dalvi,

Anirudha Joshi, Devanuj K. Balkrishan, Jacki O'Neill, and Marco Winckler, editors, *Human-Computer Interaction - INTERACT 2017*, pages 441–459, Cham, 2017. Springer International Publishing.

[41] Patel MS, Asch DA, Rosin R, and Et al. Framing financial incentives to increase physical activity among overweight and obese adults: A randomized, controlled trial. *Annals of Internal Medicine*, 164(6):385–394, 2016. URL: `+http://dx.doi.org/10.7326/M15-1635`, doi:10.7326/M15-1635.

[42] Patel MS, Benjamin EJ, Volpp KG, and Et al. Effect of a game-based intervention designed to enhance social incentives to increase physical activity among families: The be fit randomized clinical trial. *JAMA Internal Medicine*, 177(11):1586–1593, 2017. URL: `+http://dx.doi.org/10.1001/jamainternmed.2017.3458`, doi:10.1001/jamainternmed.2017.3458.

[43] Austin Myers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban, Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, and Kevin Murphy. Im2Calories: Towards an automated mobile vision food diary. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter(December):1233–1241, 2015. doi:10.1109/ICCV.2015.146.

[44] Mitesh S. Patel, Luca Foschini, Gregory W. Kurtzman, Jingsan Zhu, Wenli Wang, Charles A.L. Rareshide, and Susan M. Zbikowski. Using wearable devices and smartphones to track physical activity: Initial activation, sustained use, and step counts across sociodemographic characteristics in a national sample. *Annals of Internal Medicine*, 167(10):755–757, 2017. doi:10.7326/M17-1495.

[45] Patrick, K., Griswold, G. W., Raab, and F. Health and the mobile phone. *American Journal of Preventive Medicine*, 35(2):177–181, 2008. URL: `http://apps.isiknowledge.com/full{\_}record.do?product=CABI{\&}search{\_}mode=Refine{\&}qid=49{\&}SID=3CEcJmNAPbH4nkMjbpO{\&}page=3{\&}doc=107`, doi:10.1016/j.amepre.2008.05.001.Health.

[46] Calie Pistorius. Developments in emerging digital health technologies. (1), 2017.

[47] James O. Prochaska, Wayne F. Velicer, Colleen Redding, Joseph S. Rossi, Michael Goldstein, Judith DePue, Geoffrey W. Greene, Susan R. Rossi, Xiaowu Sun, Joseph L. Fava, Robert Laforge, William Rakowski, and Brett A. Plummer. Stage-based expert systems to guide a population of primary care patients to quit smoking, eat healthier, prevent skin cancer, and receive regular mammograms. *Preventive Medicine*, 41(2):406–416, 2005. doi:10.1016/j.ypmed.2004.09.050.

[48] Ellen Henrietta (Swallow) Richards and Louise Harding. Williams. The dietary computer. Explanatory pamphlet; the pamphlet containing tables of food composition, lists of prices, weights, and measures, selected recipes for the slips, directions for using the same., 1902. URL: `file://catalog.hathitrust.org/Record/005812628http://hdl.handle.net/2027/uc2.ark:/13960/t46q1vm37http://hdl.handle.net/2027/uc1.{\$}b102933http://hdl.handle.net/2027/hvd.32044080799737http://hdl.handle.net/2027/hvd.hc35c9http://hdl.handle.net/2027/hvd.h`.

[49] Verónica Rivera-Pelayo, Valentin Zacharias, Lars Müller, and Simone Braun. Applying quantified self approaches to support reflective learning. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK '12*, page 111, 2012. URL: `http://dl.acm.org/citation.cfm?doid=2330601.2330631`, doi:10.1145/2330601.2330631.

[50] Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeshwar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. A Deep Reinforcement Learning Chatbot. (Nips):1–9, 2017. URL: `http://arxiv.org/abs/1709.02349`, arXiv:1709.02349.

[51] Amalia Suzianti, Rizky Puti Minanga, and Felisa Fitriani. Analysis of User Experience (UX) on Health-Tracker Mobile Apps. *International Journal of Computer Theory and Engineering*, 9(4):262–267, 2017. URL: `http://www.ijcte.org/index.php?m=content{\&}c=index{\&}a=show{\&}catid=87{\&}id=1402`, doi:10.7763/IJCTE.2017.V9.1148.

[52] Latanya Sweeney. Computational disclosure control: A Primer on Data Privacy Protection. *Computer Science*, Ph.D.:216, 2001.

[53] Adam Tanner. For Sale: Your Medical Records. *Scientific American*, 314(2):26–27, jan 2016. URL: `http://www.nature.com/doifinder/10.1038/scientificamerican0216-26`, doi:10.1038/scientificamerican0216-26.

[54] USDA Food and Nutrition Service. Implications of Restricting the Use of Food Stamp Benefits: Summary. 2007. URL: `http://www.fns.usda.gov/sites/default/files/arra/FSPFoodRestrictions.pdf`.

[55] Joseph Weizenbaum. ELIZAa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[56] Henry H. Wilmer, Lauren E. Sherman, and Jason M. Chein. Smartphones and cognition: A review of research exploring the links between mobile technology habits and cognitive functioning. *Frontiers in Psychology*, 8(APR):1–16, 2017. doi:10.3389/fpsyg.2017.00605.

[57] Ryan Witt. Healthcare data: A hacker's jackpot — Healthcare IT News. URL: `http://www.healthcareitnews.com/blog/healthcare-data-hackers-jackpot`.