Machine Learning, Wearable Computing, and Alzheimer's Disease

Yanrong Li
Jun Jie Ng
Chong Wee Tan
Marie Douriez
Ludovic Thea

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-91
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-91.html

May 13, 2016

Department of Electrical Engineering

and Computer Science

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Spring Semester 2016

Machine Learning, Wearable Computing and Alzheimer's Disease

Capstone Project Technical Report

Yanrong Li

yanrong.li@berkeley.edu

Apr 15 2016

Co-authors: Chong Wee Tan, chongwee@berkeley.edu

Jun Jie Ng, jng12@berkeley.edu

Marie Douriez, marie.douriez@berkeley.edu

Ludovic Thea, ludovic.thea@berkeley.edu

Professor: Prof. Alexandre Bayen, bayen@berkeley.edu

Acknowledgements

I would like to thank Prof. Alexandre Bayen for proposing the project, giving me the opportunities to do the project, and his leading to potential publications. I would like to thank Prof. John Canny for being the second project advisor. I would like to thank Dr. George Netscher for his guidance in the project. I would like to thank Dr. Lynn Huang and Dr. Alexandre Beliaev for their help on technical communications. Finally I would like to my teammates Chong Wee Tan, Jun Jie Ng, Marie Douriez and Ludovic Thea, without whose contribution the project would not be done.

Yanrong Li Apr 15, 2016

CHAPTER 1

Individual Technical Contributions

1.1 Introduction

1.1.1 Background and Previous Work

Alzheimer's disease is the sixth most frequent cause of death in the US and many of the deaths are caused by the increasing chance of anomaly behaviors as disease progresses("2015 Alzheimer's Disease Facts And Figures", 2015). To prevent the potential hazard and monitor patients' physical conditions, human caregivers are needed. But human caregiving service costs much: professional caregiving service is usually expensive, while non-professional, family caregivers will have to spend time on this. Moreover, sometimes human caregivers may misjudge the patient's conditions because of subjective understanding or occasional carelessness. In order to improve the monitoring patients with better accuracy and free people from caregiving services, we are aiming to develop a system which implements automatic Alzheimer's patients' behavior classification.

According to the proposal, the system works along a procedure of "data collection, data uploading and data analysis". First is the data collection: an Android smartwatch the patient wears collects data via Bluetooth from one

Estimote and multiple SensorTags deployed in the environment. These data include beacons sent periodically by the Estimote, and the accelerations, light intensities, pressures, etc, sent by the SensorTags. Then the data will be forwarded to an Android phone paired with the watch. Second is data uploading: the smartphone temporarily stores the data, and posts the data onto the server via backend programs at certain time of a day. Last is data analysis: Since the Estimote beacon strength provides the location information of the patient and the readings of the sensors reflect patients' influence on the environment, the algorithms implemented on the server based on heuristic design and machine learning should be able to analyze the data and classify the patient's activities.

This project had been in progress for 1 year when we took it over. The system frame, named CareEcoSystem, had been implemented by George Netscher, Julien Jacquemot and Bradley Zylstra. In their program, people can manually set up caregiver-patient pairs, associate Estimotes with patients, and initiate the system. A complete data pipeline has been implemented by George, Julien and Brad in Android to gather Estimote beacons and watch sensor data like GPS, wrap them up and upload them to the UCSF server every 12 hours together with other events. The code judging which room the patient is in has been implemented on the server in Python. Through that system, they can determine which room the patient is in with $\geq 95\%$ accuracy, and can also handle special events like "watch low battery", "patient forgets to wear watch", "patient leaves home", etc. However, it is far from enough to determine patient activities because the indoor activities only affects SensorTag readings. Hence, we were expected to integrate a new data pipeline into the CareEcoSystem which collects and uploads SensorTag data, and to implement algorithms to classify patient daily activities.

1.1.2 Work Break-down

In this project, my tasks involve all of the three parts: data collection, uploading and analysis. In data collection, I created methods to fetch SensorTag

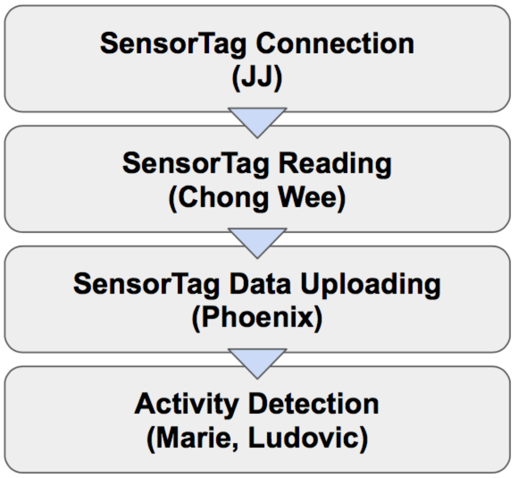


Figure 1.1: The work break-down diagram of the project

configuration information from smartphone to smartwatch to aid a more efficient SensorTag connection. In data uploading, I created methods and data structure to store and upload SensorTag readings to the server, and I transplanted the server code from UCSF private server to the DigitalOcean Droplet, which is a public cloud service where custom applications can run. In data analysis, I helped to merge the offline analysis code to the server program to make it run online, and I am expected to program classifiers of many patient activities such as showering and cooking.

Chong Wee and Jun Jie have been working on SensorTag connections and hardware trouble-shooting, while Marie and Ludovic have been working on offline data analysis and literature review. Finally we deployed the system together. A more detailed work break-down diagram is shown in Figure 1.1.

1.2 Data Pipeline

In this section I will report my contribution in the construction of data pipeline used to gather and transmit SensorTag data. It involves two major aspects: data collection and data uploading.

1.2.1 Data Collection

The major contribution of data collection is done by Chong Wee and Jun Jie. They implemented the simultaneous connection to up to seven SensorTags via Bluetooth, and fetch readings from the sensors. But we soon realized that their code was blindly reading from all sensors in a connected SensorTag, which is not very economic. As we know, a SensorTag integrates many sensors, but not all of them are working. For example, when detecting shower events in the bathroom, only the data from the humidity meter in that SensorTag is useful, and fetching readings from accelerometer and gyroscope will shorten battery life. The redundant data stored in the database is also a waste of storage, and may cause more difficulties to handle in the data analysis. A more efficient and economic way to collect data is only reading from the sensors that are used in a SensorTag, but that requires the watch to know which sensors are being used in a SensorTag when setting up connections. In order to aid this, I implemented a subsystem.

UI and Database to manage SensorTag configuration

The first part of this subsystem is the database to store SensorTag configuration information, as well as a user interface to access and manage the database. I firstly created a SQLite database on the smartphone, and a Java class $MyDatabaseHelper$ which has a series of public methods to insert, modify, delete and view the entries in the database table. Then I created a new Activity in the CareEcoSystem phone app, in which an object of $MyDatabaseHelper$ is created so that its member methods can be called. With a user interface Figure 1.2 provided, the user can input a SensorTag ID and choose a corresponding sensor type, and click the buttons to call those methods. In this way, it is very flexible for the users to input all the <SensorTag_ID, Sensor_type> pairs.

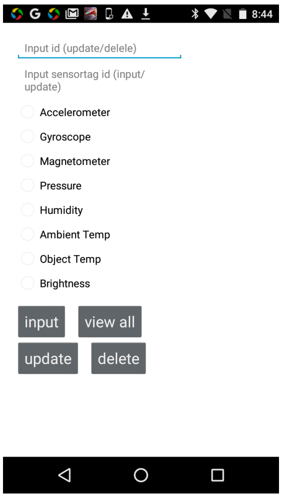


Figure 1.2: The user interface for inputting SensorTag configurations

Fetching info from phone to watch

This core work of this part is the communication between phone and watch. Since I had implemented a set of phone-watch communication code in my CS260A course, I tried to reuse the code here. However, it turned out that my code didn't work because Julien had already implemented his, and these two sets of code have the same kernel, so they are essentially conflicted and cannot run simultaneously and I had to reuse Julien's communication code. By imitating his $requestPatientInfo$ method, I implemented a $requestSensortagInfo$ method. This method is called by the watch every time the app is started, and it will trigger the following procedures:

1. Watch sends a request to phone.
2. Phone receives the request, reads all entries from the database, and wrap them into a JSON Object.
3. Phone sends back the JSON Object.
4. Watch receives the JSON Objects, parses it into two String lists. One is all SensorTag_IDs, and the other one is all Sensor_Types.

5. Watch joins the two String lists into two Strings and set them as a public variable that can be accessed by any service or activity of the App.

After this, the watch can refer to those two Strings to determine which types of sensors it should read from when connecting to a specific SensorTag.

1.2.2 Data Uploading

My work on data uploading includes two parts: first is to transplant the server code from the private UCSF server to a public cloud service, which was proposed as Google App Engine (Google, 2015) originally. This task, seemed to be easy, actually cost a long time until I found an alternative solution, and it will be explained below. Second is to store and upload the SensorTag data after the watch collects it.

Configure DigitalOcean Droplet and transplant server code

After the SensorTag data are collected, they will be analyzed using various tools and algorithms. These computations require high performance of the device, so it is impossible to have them done directly on the smartphone or the smartwatch, and we have to use a computer to analyze data. For personal computers or laptops, we can run the computing programs, but the programs will keep occupying the computer resources and the computer has to be kept power-on.

Therefore, we choose servers to be the computing nodes (a server is essentially a remote computer). In this case, before the data is processed and analyzed on the server, they will be firstly fetched from the phone database, uploaded to the server, received by the server program (it's also called backend) and stored in the server database.

When we took over the project, the server program was deployed on a private server belonging to UCSF. However, this is not sustainable because we were aiming to independently develop the project, causing as few influences on the third-party as possible (for example, when using UCSF server to debug, some notifications may be sent to the caregivers there). What we needed was a public service because we would have complete rights to use it once we pay. Therefore, the proposal suggested we transplant the code to Google App Engine. However, the transplant is not easy: to upload the data onto server, rather than simply connecting a USB cable, we will have to follow certain communication protocols. The protocols may be different depending on the server we choose. Moreover, the backend program is merely a relay between the client (Android phone) and the database. We will have to reconstruct a database on the server to store the data. Therefore, the work for me included three major parts: modifying the APIs of both Android phone and the server code to set up communications, deploying backend program on Google App Engine, and reconstructing the database.

Google App Engine (GAE) is a cloud computing service providing environment to run backend code in multiple languages, and many methods to communicate with different clients including Android phone. It seems its flexibility and multiple functions make it user-friendly. However, it has a significant problem: the IP address of the server is hidden and flexible. In our original code, as long as we acquire the IP address and write it into the Android program, the connections will be set up, but there is no way to utilize the connection method on GAE, so we have to use the provided API. Applying new APIs is not simply the same as updating functions. Instead, a part of the program has to be redesigned because we are adding a module. Therefore, most of my time in the first half semester were spent on reading Android code, conducting trials on sample backend on GAE, reading GAE documentations, and modifying codes to use the APIs.

As the tedious documentations gave me a bad feeling about the prospect of modifying code to adapt to GAE, I decided to deploy the backend code without modification and to see how it works. To my surprise, our server code cannot run on GAE, not because of bugs but because of its high dependency on C libraries. Since the server code does data processing and analysis, some Python libraries such as NumPy, SciPy and Scikit-Learn are also adopted, and these mathematical tools are relying on C libraries. But GAE only allows pure python code and libraries to run, so I had to stop my research on GAE, and seek for substitutions.

Very soon I re-selected several cloud services which allow C-libraries to run, and after comparing them I decided to use DigitalOcean. Although it seems we have to start from scratch, actually this is not bad news. Because DigitalOcean server (also called droplet) provides fixed IP address, which means we can access it via regular HTTP protocols including the one implemented in our original code – it essentially has no difference with UCSF server.

Deploying the code on DigitalOcean droplet requires a pre-configuration of the environment. It is time-consuming but not difficult, basically just installing the softwares and libraries needed such as Sci-py, Scikit-learn and Theano. After I configured the DigitalOcean Droplet, I copied the database of UCSF server and duplicated it on my Droplet. Since the original database has real data from people such as GPS record, and using them without carefulness may cause a private issue, I just duplicated the framework of the database, i.e., I copied the database name, table name and attribute name, but all the tables are empty except for the tables storing necessary configuration information. The lack of documentation causes some trouble, but it didn't prevent me from resolving all problems by deducing from the error message. At last the server program ran successfully on the DigitalOcear Droplet.

The final step was to set up the connection between Android phone and the server. I added the SSH credential of my Droplet in the Android App, but the App still failed to connect to the server. With the help of Bradley, we found that the FQDN in the hosts file was not properly set: it should have been set as the IP address of the Droplet instead of a localhost address, which made the server program listen to a wrong address.

In summary, this part of work took a long time basically because we went into a wrong direction at the beginning, and none of us are familiar with constructing

...	sensortag_id	type	reading_all	reading_x	reading_y	reading_z
...	68:C9:0B:05:0C:06	Gyroscope	0	1.0356	0.0105	0.0068
...	68:C9:0B:05:0C:06	Humidity	25.6835	0	0	0

Table 1.1: The relation to store SensorTag Readings. The leftmost "..." represents three columns "patient_id", "timestamp" and "is_committeed", which are general attributes for all the relations. Due to the limit of space we don't show them all here

servers. Plus, the server code we are using is still not completed and contains lots of bugs. During the transplant of server code I also helped Julien and Brad fix many problems, which are too detailed to be stated in the report.

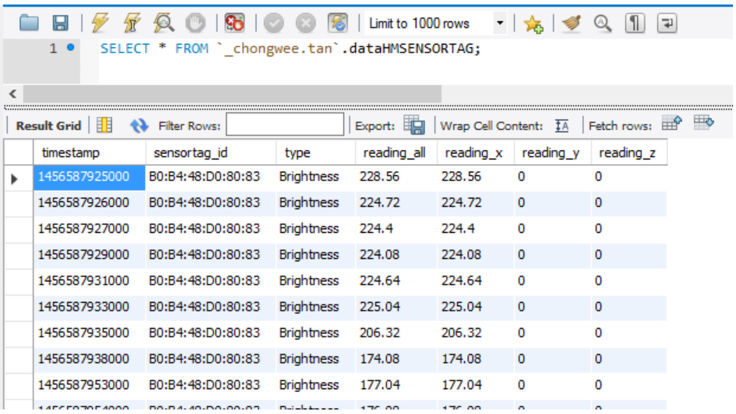
Create methods and tables to handle SensorTag data

Chong Wee and Jun Jie's work had given us access to the SensorTag readings, but these data are stored in .csv files instead of uploaded to the server database, so what I needed to do is to refer to the methods and tables handling Estimote data to create another set of code to handle SensorTag data.

First is to create a table to store SensorTag data in the SQLite database on phone and watch. In general, the sensors can be separated into two types. One type is 1-dimension reading sensor, such as humidity meter, luxometer and thermometer. The other type is 3-dimension reading sensor, such as accelerometer and magnetometer which contain readings in three components x, y and z. In order to minimalize the complexity of the tables and generalize both of them, I designed a relation as shown in Table 1.1.

The 0s are useless data here. When we analyze the data entry, we check the "type" of each entry to see if it is a 3-dimensional reading or a 1-dimensional reading. If it is 1-dimensional, then we just read attribute "reading_all". While if it is 3-dimensional, then we read attributes "reading_x", "reading_y" and "reading_z".

Then, I created a method to parse the entry of SensorTag table for uploading, and did some other minor modifications to make sure the SensorTag table is always processed together with other tables such as Estimote table, GroundTruth



```
SELECT * FROM `_chongwee.tan`.dataHMSENSORTAG;
```

timestamp	sensortag_id	type	reading_all	reading_x	reading_y	reading_z
1456587925000	B0:B4:48:D0:80:83	Brightness	228.56	228.56	0	0
1456587926000	B0:B4:48:D0:80:83	Brightness	224.72	224.72	0	0
1456587927000	B0:B4:48:D0:80:83	Brightness	224.4	224.4	0	0
1456587929000	B0:B4:48:D0:80:83	Brightness	224.08	224.08	0	0
1456587931000	B0:B4:48:D0:80:83	Brightness	224.64	224.64	0	0
1456587933000	B0:B4:48:D0:80:83	Brightness	225.04	225.04	0	0
1456587935000	B0:B4:48:D0:80:83	Brightness	206.32	206.32	0	0
1456587938000	B0:B4:48:D0:80:83	Brightness	174.08	174.08	0	0
1456587953000	B0:B4:48:D0:80:83	Brightness	177.04	177.04	0	0

Figure 1.3: The luxometer data uploaded to the server database.

table, etc.

For the server-end code which receives the uploaded data, I didn't need to modify anything because it had been implemented with strong generality: it automatically creates tables to store SensorTag data as long as the uploaded file is following a correct format. Finally I merged my code with Chong Wee and Jun Jie's code, and we formally combined our work to form a complete data pipeline collecting and uploading SensorTag data. Figure 1.3 shows that the SensorTag readings now can be smoothly transmitted to the server database.

1.3 Data Analysis

Data Analysis is actually the core topic of this project. However, during the last several months, I spent most of the time on building the systems and did no significant contributions on this part. The major work on data analysis has been conducted offline by Ludovic and Marie. They were able to manually load raw data and run algorithms locally, while our aim is to merge the algorithms with the server code frame to let them run online. In the following paragraphs, I will describe what I am doing and plan to do before my leaving.

1.3.1 Merge Offline Analysis Codes with Server Program

Currently I am merging the offline analysis code with the server program. The principle is simple: the main thread of the server program calls $AnalysisThreadHandler$ every 12 hours. The $AnalysisThreadHandler$ basically calls all the analyzing methods to analyze the new data, and output the results to the result table. Originally there was only Estimote RSSI data in the database, so there was only *RoomLocationing* classifier implemented. I created several new classes to classify showering, cooking, flushing toilet, etc., and added them to the $AnalysisThreadHandler$. Now I am debugging the code and hopefully it can be done by my graduation.

1.3.2 Improve algorithms to classify activities

The algorithms may not be as scalable as wished because we are not only collecting data from 1-2 persons' homes, so the improvement on generalization is probably needed. In order to test the generalizability of the algorithms, we will need to deploy the systems in various indoor structures. Now we have reached several volunteers on experiments and will guide them deploy the systems as soon as we finish the polishing of the system code.

1.4 Other Contributions

All of us shared some contributions on team events, including team paper, presentations, Leadership course assignments, documenting codes, etc. The most memorable activity we finished together was joining a presentation in UCSF on Feb 2, 2016. During the presentation Jun Jie reported the temporary achievements of our project perfectly.

1.5 Future Plan

There is only one month left before our graduation and the end of this project. However, for me there are still at least three months remaining. Considering that we have taken too long time on system implementation and we are a little bit behind the schedule on data analysis, I proposed a volunteer on this project during the summer semester before I start working in company in July. I wish during this period I can achieve significant progress on classifying patients' activities and detecting anomalies online. We have also planned to lead the work into potential publications.

1.6 Conclusion of Individual Contributions

In this project, I contributed significantly on data pipeline to collect and transmit SensorTag data. I am also integrating data analysis code with the server program now and hopefully will get it done by May. During the following summer I plan to dig into the project deeper and try to achieve some publications. The project gives me opportunities to practice coding in Android and Python, to apply leadership theories, to make friends with super nice professor and students, and to do good to people and families suffering from illness.

CHAPTER 2

Engineering Leadership

2.1 Introduction

Given the prohibitive cost of nursing facilities and dedicated care for Alzheimer's Disease patients (Michael Hurd & Langa, 2013), our project aims to design and develop an affordable (under $1,000) system for tracking and monitoring patients in their own homes. Using inexpensive and commonly available sensors, wearable devices and smartphones, the system constantly gathers information about the patient's daily routine and relays it to a centralized server. This is where machine learning algorithms are used to build up knowledge of the patient's behavior over time, before applying analysis to flag any potential anomalies. In doing so, this system will allow patient's to stay safer in their own homes and provide medical practitioners with updates on the progression of their patients' condition. In the following sections, we shall analyze the industry that our project operates within, discuss the technology strategy that we adopted and highlight the intellectual property concerns we have.

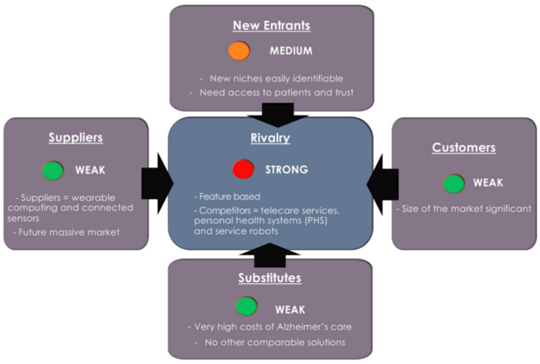


Figure 2.1: Porter's Five Forces on Technology-Integrated Home-based Care Industry

2.2 Industry Analysis

For industry analysis, we reference Porter's Five Forces (Porter, 1979) and performed an analysis on each of the forces with regard to the industry for technology-integrated home-based care solutions for Alzheimer's Disease patients. Porter's framework allows us to compare the relative strengths and weaknesses of the following forces: Threat of New Entrants, Bargaining Power of Customers, Bargaining Power of Suppliers, Threat of Substitutes and Intensity of Rivalry. In doing so, we gain clarity in the profitability of the industry in general and how the stable state might appear in the long term.

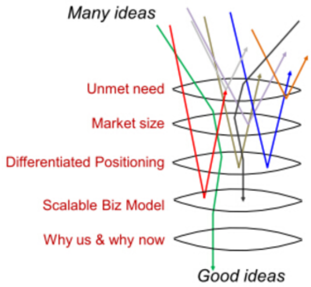
For the threat of substitutes, we assess that the force asserted is extremely weak. According to a paper published by the New England Journal of Medicine, the average annual financial cost of Alzheimer's care in 2010 was $28,501 (Michael Hurd & Langa, 2013). Out of that, 48.6% was attributed to payments made to nursing home care (Michael Hurd & Langa, 2013). By allowing Alzheimer's patients to live at home independently as long as possible, they will be able to save a substantial amount of the financial cost associated with Alzheimer's care. Moreover, nearly 18 billion hours of unpaid care was provided to Americans with Alzheimer's Disease in 2014 ("2015 Alzheimer's Disease Facts And Figures", 2015). Through the use of technology to replace dedicated caregiving, our solution provides a game-changing alternative to Alzheimer's Disease patients. Specifically, our solution is significantly more cost efficient (under $1,000 compared to $28,501) and reduces the human intervention needed by existing solutions by doing away with a dedicated caregiver. Hence, we believe that the threat of substitutes to the industry is weak.

In terms of the bargaining power of customers, we assess this to be weak. There are around 47.8 million people in the US aged 65 or older, representing 15% of the total US population (IBISWorld, 2014). Since Alzheimer's Disease and other dementias are highly correlated with age, nearly 5.1 million senior Americans are expected to be afflicted by Alzheimer's Disease ("2015 Alzheimer's Disease Facts And Figures", 2015). Given the size of the market, it is unlikely that any one single group of customers can assert significant pressure on the industry. Furthermore, the high cost of substitutes mentioned previously, such as nursing homes and dedicated caregivers, further erodes the bargaining power of customers.

Next, we argue that the bargaining power of suppliers is weak as well. Since the industry depends heavily on off-the-shelf wearable computing and connected sensors (a subset of the Internet of Things category), we shall examine these suppliers in detail. Firstly, it is predicted that 33% of adults in the US shall own a wearable device by 2017, and the smartwatch will lead this growth by taking an increasingly larger share of the smart devices (Insider, 2016). Secondly, it is also predicted that the worldwide market for IoT solutions will grow from $1.9 trillion in 2013 to $7.1 trillion in 2020, and will be the world's most massive device market, equivalent to double the size of all of the other smart devices market in 2019 (*Internet of Things - HorizonWatch 2015 Trend Report*, 2016). These trends are immensely beneficial to our industry since the prevalence of smartwatches and connected sensors represents substantial competition between suppliers, lowering the bargaining power that individual suppliers may possess over our industry.

As for intensity of rivalry within the industry, our team anticipates strong competition from the different players, namely telecare services, personal health systems (Codagnone, 2009) and service robots. According to Pols (Pols, 2012), 'Telecare' is an "umbrella term referring to the technical devices and professional practices applied in 'care at a distance', care that supports chronically ill people living at home". It is a passive, unsophisticated technology that requires the patient to initiate a call to the telecare center in times of need. A second form of technology integrated home-based care solutions can be generally termed as Personal Health Systems (Codagnone, 2009). This refers to any home-based ICT system that collects, monitors and manages health-care related information (Codagnone, 2009) (Doris Schartinger & et al., 2013) (Fabienne Abadie & Maghiros, 2010). In contrast to telecare, PHS is an active technology comprised of motion sensors, on-furniture sensors (e.g. chairs, beds, doors) and video cameras (activated only during emergencies for privacy) to monitor patients in their own homes. At the forefront of technology-integrated patient care, there have been attempts to introduce service robots to the patient care activity chain. Compagna and Kohlbacher (Compagna & Kohlbacher, 2015) investigated instances of service robots in patient care facilities, identifying possible applications in logistical activities to reduce caregiver workload and therapy enhancement through multimedia entertainment.. In summary, given the myriad solutions offered in this industry, we foresee that the rivalry would be strong and features-based, where companies innovate to provide more timely and accurate monitoring for both patients and caregivers.

Lastly, the threat of new entrants is the most uncertain force to this industry. While intellectual property could form significant barriers to entry, the diversity of solutions available means that a new entrant could easily identify a new niche to operate within. Nevertheless, we believe that access to patients and trust formed between patients and companies could be a key barrier formed by the incumbents. Once an incumbent has cornered a sizeable portion of the market, the reputation and trust it has built up would be its key weapon against any new entrant. Thus, we feel that there is a medium threat in the form of new



Criteria	Score
Unmet Need	A
Market Size	A
Differentiated Positioning	A-
Scalable Business Model	A
Why us	A-
Why now	B+

Figure 2.2: Applying Zafar's Product Idea Evaluation Framework to our project

entrants.

2.3 Technology Strategy

Based on Naeem Zafar's framework for determining the quality of a product idea (Zafar, 2009), we assessed our project's suitability for product creation. While this framework is meant to evaluate a product idea at its incubation phase before proceeding to development, we have applied it retrospectively to determine its suitability for transition from an academic project to a full fledged commercial product.

To begin with, the framework requires us to consider whether an unmet need exists for the product. Clearly, based on the high costs associated with nursing homes and dedicated care-giving services, Alzheimer's Disease patients have a strong desire to stay at home independently (Score for Unmet Need: A). With an estimated 5.1 million Americans above 65 with Alzheimer's Disease and a further 200,000 below 65 ("2015 Alzheimer's Disease Facts And Figures", 2015), we expect the market size to be relatively large, thereby bringing with it substantial opportunities for commercialization and growth (Score for Market Size: A).

In terms of differentiated positioning, we have investigated potential competitors in terms of telecare systems, PHS and service robots. Our proposed solution can be considered as a form of PHS, considering that it uses sensors to track patient activity. However, our solution has a significant edge over contemporary implementations which are prone to false alarms due to the usage of predetermined thresholds (Neven, 2015). In contrast, our proposed solution makes use of machine learning to recognize patient behavior over time, hence reducing the likelihood of false alarms. By maintaining machine learning as our competitive edge over our competitors, our proposed solution should emerge as the forerunner in this field (Score for Differentiated Positioning: A-).

Next, the framework has prompted us to consider the scalability of the business model. Our proposed solution exploits prevalent off-the-shelf technologies such as wearable computing, IoT and next generation communication infrastructure. Since these components and infrastructure are readily available to our customers, our scalability will not be constrained by them. Furthermore, as our added value comes largely from the machine learning algorithm and server-based processing, we are able to easily scale up our operations by adding server capacity (Score for Scalable Business Model: A).

Lastly, the framework challenges the team to answer why we should be the one to bring such a product to market and why this is the most opportune time to do so. We believe that we are in an excellent position to do so given our relationship with the Memory and Aging Center at University of California San Francisco. This provides us with access to a panel of neurologists, patients, caregivers and healthcare regulators, allowing us to refine the product to best fit the needs of Alzheimer's Disease patients (Score for Why Us: A-). In addition, we believe the timing is appropriate since we are approaching a phase in technology where wearable computing, connected sensors and high bandwidth data communications are becoming ubiquitous, allowing us to deploy our solution at scale in a cost efficient manner. Moreover, the field of machine learning is advancing rapidly and this technology has been proven to produce results when large amounts of data is available (Score for Why Now: B+).

2.4 Intellectual Property

For this project, we are in an unique position with regard to topics on intellectual property. Since our work brings significant benefit to patients, publishing our algorithms and source code in the public domain would greatly benefit the Alzheimer's Disease community or even the greater dementia community. This can be achieved by lowering barriers of entry for companies to join the industry, thereby bringing down the cost of development and spurring further innovations in this area.

On the other hand, given the size of the target market and viability of the product, there are enormous opportunities for commercialization. By retaining the intellectual property and building a commercial product around it, we would be able to introduce a truly differentiated offering to the market, with the potential of achieving sizeable returns to our research and development. Alternatively, should we decide not to bear the risks of bringing the product to market, there are also potential gains to be reaped by licensing the intellectual property to interested parties.

Despite the lure of potential gains, our team firmly believes that such innovation should be propagated to benefit the millions of patients afflicted by Alzheimer's Disease. Through open-sourcing the intellectual property, we hope that like-minded researchers can also contribute towards our effort and allow patients to stay safer in their own homes.

References

2015 alzheimer's disease facts and figures. (2015). *Alzheimer's& dementia: the journal of the Alzheimer's Association*, *11*(3), 332.

Codagnone, C. (2009). *Reconstructing the whole: present and future of personal health systems* (Tech. Rep.). PHS.

Compagna, D., & Kohlbacher, F. (2015). The limits of participatory technology development: The case of service robots in care facilities for older people. *Technological Forecasting And Social Change*, *93*, 19-31.

Doris Schartinger, S. G., & et al., B. H.-S. (2013). *Personal health systems: State of the art* (Tech. Rep.). AIT Austrian Institute of Technology.

Fabienne Abadie, M. v. L. C. P. P. B. A. H. J.-A. V., Cristiano Codagnone, & Maghiros, I. (2010). *Strategic intelligence monitor on personal health systems (simphs) - market structure and innovation dynamics* (Tech. Rep.). Institute for Prospective and Technological Studies, Joint Research Centre.

Google. (2015). *App engine: A powerful platform to build web and mobile apps that scale automatically.* Retrieved 2015-11-28, from `https://cloud.google.com/appengine/`

IBISWorld. (2014). *Number of adults aged 65 and older* (Tech. Rep.). Business Environment Profiles, IBISWorld.

Insider, B. (2016). *The wearables report: Growth trends, consumer attitudes, and why smartwatches will dominate.* Retrieved 2016-03-05, from `http://www.businessinsider.com/the-wearable-computing-market-report-2014-10`

Internet of things - horizonwatch 2015 trend report. (2016). Retrieved 2016-03-05, from http://www.slideshare.net/HorizonWatching/internet-of-things-horizon-watch-2015-trend-report-client-version-28jan2015

Michael Hurd, A. D. K. M., Paco Martorell, & Langa, K. (2013). Monetary costs of dementia in the united states. *New England Journal of Medicine*, *368*(14), 1326–1334.

Neven, L. (2015). By any means? questioning the link between gerontechnological innovation and older people's wish to live at home. *Technological Forecasting and Social Change*, *93*, 32–43.

Pols, J. (2012). *Jeannette pols.* Amsterdam University Process.

Porter, M. (1979). *How competitive forces shape strategy.*

Zafar, N. (2009). *7-steps to a successful startup.* Five Mountain Press.