HealthyLife: An Activity Recognition System with Smartphone Using Logic-Based Stream Reasoning

Thang M. Do, Seng W. Loke, and Fei Liu

Department of CSCE La Trobe University, Bundoora, VIC, 3086, Australia

Abstract. This paper introduces a prototype we named *HealthyLife* which uses Answer set programming based Stream Reasoning (ASR) in combination with Artificial Neural Network (ANN) to automatically recognize users activities. HealthyLife aims to provide statistics about user habits and provide suggestions and alerts to the user to help the user maintain a healthy lifestyle. The advantages of HealthyLife over other projects are: (i) no restriction on how to carry the phone (such as in hand bag), (ii) detect complex activities and give recommendations, (iii) deal well with ambiguity when recognizing situations, and (iv) no additional devices are required.

Keywords: health promotion, Answer Set Programming, stream reasoning, sensors, smart phone, activity recognition.

1 Introduction

Using smartphones for health support and health-associated activity recognition has been receiving much attention from researchers and end users. However, most projects have one or more of the following limitations.

The first limitation is the way users carry a smartphone. To maintain accuracy, most projects require users to carry their phones at a fixed position on their bodies or carry additional equipment. The most flexible prototype requires the phone was put in users' pants pocket [1]. This position may not be practical for many women who often keep their phone in their hand bag. The second issue is the automation; some commercial products (e.g., sports tracker of Nokia [2]) can provide useful statistics information about fitness activities but requires users to manually label start, stop times and name the activity.

The third issue is the usefulness of provided information; many projects can automatically detect only users' "basic" activities like walking, standing and running and do not use this information to give further useful suggestions, or infer higher level complex activities. Also, once activities are recognized, reasoning is needed to then provide the right suggestions to users.

To fill the above gaps, we are working on *HealthyLife*, a prototype system which focuses on using available smartphone sensors to automatically recognize user's basic, and complex activities, and give useful health-related suggestions

K. Zheng, M. Li, and H. Jiang (Eds.): MOBIQUITOUS 2012, LNICST 120, pp. 188–199, 2013.

and alerts. *HealthyLife* virtually requires no restriction on how users carry a smartphone (users can put it in pants pockets, jacket pockets, belly belt, handbag, shoulder bag, or backpack). We don't require user to wear any additional devices, and explore what is possible without such devices.

Our methodology is to use an ASR (Answer Set Programming based Stream Reasoning) logic framework (first briefly introduced in [8], and will be extended here) as the reasoning engine. ASR allows dealing with ambiguity in a logic-based framework when reasoning about situations (or *ambiguity reasoning* (§3)), decomposing complex activities into simpler ones, querying long-term data history for inferring complex activities, as well as integration with different reasoning techniques such as machine learning to process different types of sensor data (e.g., accelerometer, sound, image, etc).

The rest of this paper is organized as follows: section 2 provides a brief review of related work; section 3 introduces the concept of the ASR logic framework; section 4 discusses the design and implementation of *HealthyLife*; section 5 evaluates the prototype; and section 6 concludes.

2 Related Work

In this section we review related work which focuses on using smartphone to support users' health, more compreihensive surveys can be found in [14,13].

The applications which were deployed commercially and obtained much attention from users are the Nokia sports tracker. This application provide rich statistics of users' fitness information such as total running time, cycling time, approximate energy burned, etc. However, users still have to manually label the activity they are doing with start and stop times. Even with this limitation, the number of users are very high across more than 200 countries [2]. This number shows the huge market for this kind of application. Apple also has a similar product name Run Keeper which keeps track of user workouts.

The project in [3] uses a special fitness equipment pedometer attached to the user's waistband to detect the number of steps of a user and transmit data to the mobile phone wirelessly. The project in [4] requires users to wear a paper-sized wearable sensing device on the waist for automatically detecting user activities.

In [5], a Nike+iPod kit was put in the shoes and an iPhone in a pants pocket. This solution still has limitations as additional equipment is needed. Users in [6] are required to wear many sensors at fixed positions on their body. The prototype in [1] seems to be the most flexible in how users carry the phone but still requires putting the mobile phone in the front pants pocket.

Projects surveyed above detect simple physical activities such as walking, running, biking, lying down, or being stationary, and requires different levels of restrictions in the way users carry the smart phone and may need additional devices sometimes. Therefore, we see the necessity to have a prototype which requires no additional equipment, no limitation in carrying the phone and automatically recognize user activities (basic and complex).

3 The ASR Reasoning Framework

This section briefly introduces the ASR Programming Framework[1] which is proposed for stream reasoning [7] tailored for activity recognition.

An ASR program is a 7-tuple: $\Pi = (\alpha, \beta, \gamma, \delta, \varepsilon, \zeta, \eta)$ where: α is a set of facts, β is a set of predicates to query external resources, γ is a set of basic activity models which may use weak constraints (discussed later), δ is a set of complex activity models, ε is a set of functional reasoners (such as an ambiguity reasoner, discussed later), ζ is a set of control commands (loop control, registration), and η are other auxiliary rules, such as converting data and performing calculations.

Among them, $\alpha, \gamma, \delta, \eta$ components are built from the ASR Core Language which is based on Answer Set Programming (ASP [9]) theory and the *dlv* solver[2]. Other components facilitate stream reasoning (or continuous reasoning). In the rest of this section, we introduce our definition of basic and complex activity models, and discuss our ambiguity reasoning feature.

Activity Complexity. A *data predicate* is one which holds sensor data values.

If a rule r defining an activity and its body $B(r)$ contains only data predicate(s) then predicates in the head $H(r)$ represent *basic activities*.

If a rule r defining an activity and its body $B(r)$ do not contain any data predicate(s) (but contain predicates representing basic or complex activities), then predicates in the head $H(r)$ represent *complex activities*.

(For well-defined semantics, at the moment, ASR assumes that the program Π has only basic and complex models of activities and doesn't have any model in which the body has a mixture of basic activities and data predicates.)

For example, we have a model defining two activities: "running while late" and "running while early":

```
1. accelerometer(3). time(17). % 17 is 5PM
2. running :- accelerometer(X),#int(X), X > 2.
3. runningWhileLate :- running, time(T), #int(T), T > 18.
4. early :- time(T), #int(T), T < 8.
5. runningWhileEarly :- running, early.
```

In this program, `accelerometer` and `time` are data predicates as they hold sensor data values (line 1). `running` and `early` are basic activities[3] as there are only data predicates in the rules' body (lines 2 and 4). `runningWhileEarly` is a complex activity as there is no data predicate in the rule body (line 5); `runningWhileLate` have both a basic activity and data predicates in the body (line 3) - which we call "middle complex" and will be considered further in future work, but middle complex rules can be easily avoided by having a rule called `late` as a basic activity.

[1] ASR language was proposed by us in another paper which is under review process.
[2] `http://www.dlvsystem.com/dlvsystem/index.php/Home`
[3] We can think of being "early" as a situation rather than an activity, as we have in mind the condition of the user getting there early.

Ambiguity Reasoning in ASR. Sensor data for activity recognition is normally ambiguous. Two activities (such as staying still and sitting on train) can have similar sensor data patterns. *Ambiguity reasoning* is applying a process (as below) in trying to find out the right activity in this situation.

Using normal rules and (strong) constraints (conditions which are not allowed to happend) gives only the right answer which is 100 percent true, according to the knowledge base of a logic program. However, using weak constraints (conditions which can happend with a violation cost), besides the the right answer, we keep potential answers (called the *best models*) with their violation degrees (or costs). The right answer will have violation cost = 0.

Each weak constraint has a weight which is defined based on how important the constraint is. In general, the more important a weak constraint is the more weight it has. The violation cost of a best model is the sum of the weight of all weak constraints which the best model violates.

ASR ambiguity reasoning observes sequences of the best models over a period of time to discover which activity seems to be the most likely answer.

For example, a program (*dlv* syntax) uses weak constraints (marked with ":~", [1:1] is [violation cost:priority level]) to recognize activities based on accelerometer data and GPS-acquired speeds as follows:

```
1. running :- acce(A), A >= 13.
2. walking :- acce(A), A < 13, A > 10.
3. :~ running, speed(S), S <= 2. [1:1] % 2m/s
4. :~ walking, speed(S), S > 2. [1:1]
5. talk :- noise(N), N >= 3.
6. quiet :- noise(N), N < 3.
```

This program processes a data window of sensor data (say, from `t1` to `t4`) and gives best models with costs as follows:

```
t1: A=13, S=3, N=3 -> Best model: {running, talking}, cost: 0
t2: A=12, S=2, N=4 -> Best model: {walking, talking}, cost: 1
t3: A=10, S=2, N=3 -> Best model: {walking, talking}, cost: 1
t4: A=11, S=2, N=2 -> Best model: {walking, quiet},   cost: 1
```

Among activities in this window, we see that running and walking (also talk and quiet) can not happen at the same time (we say they are in a *mutually exclusive* relationship) and we have to choose which activity is the more likely. The first result is *running* with zero violation but all the others are *walking* with "small" violation. This may imply that the real activity (over the period from `t1` to `t4`) can be walking.

We use a weight function to calculate how "right" a best model is in comparison to all others. In other words, the bigger cost the less value weight function has. For example, a weight function f_w can be defined as: $f_w = (MinTop - Cost)$, where $MinTop$ is the smallest number which is greater than any cost that may appear in a program Π. in the example above, we can set $MinTop = 3$ and $f_{w_1} = 3 - 0 = 3$, and $f_{w_2} = 3 - 1 = 2$ and so on, where f_{w_i} is the weight function value of best model at time t_i.

The violation cost and weight function value (or *weight*) of a best model is also the cost and weight of each activity in that model. In the above example, the violation cost of the best model at time t_1 is 0 and the weight is 3, and so, each activity *running* and *talking* in that model has cost 0 and a weight of 3.

We then use a membership function to estimate which activity has the best chance of being the right answer. For a given activity, the membership function f_m calculates the division of (i) the sum of the weights of the activity in every best model, and (ii) the sum of the weights of all activities in every best model. The activity having the highest membership value is chosen as the answer. For example, $f_m(running) = 3/(3+2+2+2) = 0.33$, $f_m(walking) = (2+2+2)/(3+2+2+2) = 0.67$. So *walking* is the more likely activity over the time period from `t1` to `t4`. In similar way, *talking* is the more likely over *quiet*.

4 *HealthyLife*: Design and Implementation

HealthyLife automatically detect users' daily activities for recording, making statistics and providing useful suggestions. *HealthyLife* aims to support office workers who normally work indoors and need to keep their life balance between being stationary and physically active, between working and relaxing, between spending time indoors and outdoors, and between being isolated and social.

Design. The architecture of *HealthyLife* is illustrated in Figure 1 and has four main components: (i) Client, which is installed on users' mobile phones, (ii) Web Server, which manages data transmission between the mobile phone and server, (iii) Data stream Manager System (DSMS), which stores and manages collected data, and (iv) Reasoning Server, which performs logical inference.

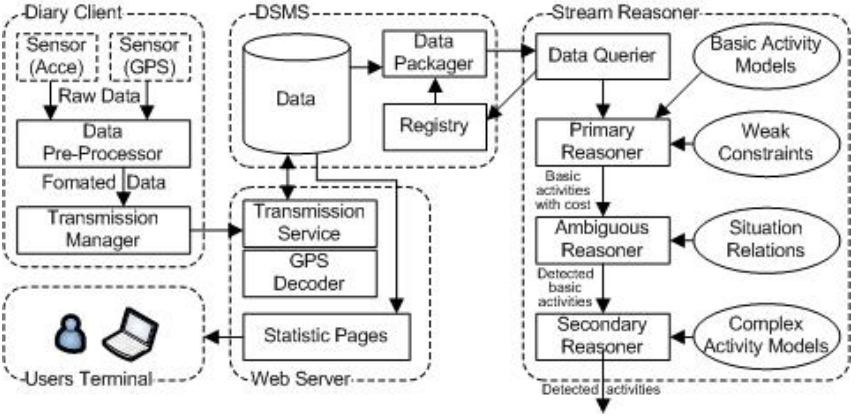

Fig. 1. Components Diagram

(i) Client: collects 3D accelerometer data, GPS data and performs pre-processing to reduce the amount of data transmitted to the server over the mobile network (and wifi in the next version). The pre-processor is a trained Artificial Neural Network (ANN) to recognize basic activities such as walking, running, driving, and staying still (when the phone is stationary).

(ii) WebServer: has a servlet for receiving data from the client and does reverse geocoding with the Google API to find out the place and address where the user is. After that, all the data is passed to the DSMS.

(iii) DSMS: stores data coming from the client through the WebServer and prepares data as requested by the reasoner. DSMS also stores streams of results from the reasoner.

(iv) Stream Reasoner: is an ASR program (§3) which has four main components as follows:

- Data Querier β: gathers four possible basic activities detected by the pre-processor and the GPS data from the DSMS stored in the form of data predicates.
- Primary Reasoner γ: uses basic activity models and weak constraints to detect all basic activities with violation cost, if any.
- Ambiguous Reasoners ε: perform ambiguity reasoning (§3) to choose the most likely user activity, periodically, over predefined periods.
- Secondary Reasoner δ: uses detected basic activities with other data (GPS at the moment) to recognize complex activities which normally happen over a longer period of time such as "user is wondering around". The secondary reasoner also gives predictions and suggestions to the user such as "user needs to exercise".

At the moment, since there is no standard ontology for activities, the decision regarding which activities are basic and which are complex is application-dependent. Also, in the set of rules used in the reasoner, we leave up to the modeller to recognize which predicates are to model basic activities and which are to model complex activities, and which predicates do not define activities.

Another component of the *HealthyLife* is the user terminal which users use to review their lifestyle with a web browser. The user terminal can be a desktop, laptop, tablet, or the smartphone itself.

Ambiguity Reasoning. We implement ambiguity reasoning in a Java package named *reasoner* with main parameters as follows:
$|W| = 5$ (basic activities window size), and window slide is 2, this mean we observe five consecutive best models and repeat ambiguity reasoning after having every two new best models. $f_w = |W| - C$ is the weight function, where C is the violation cost,
f_m is the membership function, mentioned earlier, and more generally, it is:

$$f_m(a) = \frac{\sum_{P \in g(a)} P}{\sum_{(q,r) \in F} (q * r)}$$

Where: $BM^{t'} = (\{a_i^{t'} | 1 \leq i \leq k_{t'}\}, f_w^{t'}) = (A^{t'}, f_w^{t'})$ is the best model at time t', a is an activity, $BM = \{(\{a_i^t | 1 \leq i \leq k_t\}, f_w^t) | 1 \leq t \leq |W|\}$, $F = \{(f_w^t, |A^t|) | BM^t = (A^t, f_w^t), 1 \leq t \leq |W|\}$, $g(a) = \{f_w^t | a \in A^t, BM^t = (A^t, f_w^t), 1 \leq t \leq |W|\}$.

ANN. Because developing an ANN network is not a research goal of this paper, we used sample structure in Encog[4] as follows.

Input data: The features extracted from 3D accelerometer data is the magnitude A [10] of the force vector of three values according to three directions acceX, acceY, and acceZ: $A = \sqrt{acceX^2 + acceY^2 + acceZ^2}$. For every window of five A, we calculate $Min(A)$ and $Max(A) - Min(A)$ to feed into the ANN.

ANN structure: The ANN we use has four layers which are input, output and two hidden layers. The input layer has two inputs: $Min(A)$ and $Max(A) - Min(A)$. The output layer has two bits which encode four activities: 00 - walking, 01 - running, 10 - driving, 11 - staying still. Each hidden layer has seven neurons. The activation function for hidden and output layers is the TANH, and the input layer is the Linear function.

Implementation. We developed the Client on a Samsung Galaxy S2 running Android 2.3 and the Client Repository for buffering transmitted data is SQLite. We use a HTTP client for Android to communicate with the WebServer. The database server is MySQL 5.1 DBMS. The reasoner uses the ASR logic framework. WebServer is the Apache Tomcat 7 and the reasoning server, WebServer, and Database Server are installed in Ubuntu 11.10 running as a guest OS in Virtual Box 4.1.6 (as software compatibility and quick experiment) on a Windows 7 x64 Desktop PC with Quad CPU Q9550 2.83 GHz and 4GB RAM.

5 Experimentation and Evaluation

In this section we describe how *HealthyLife* was used, and activities detected, and discuss the accuracy, system performance and the feasibility of the prototype. After being started, the Client component runs on the user's mobile phone silently and doesn't have obvious side effects on the usage of the phone. Every time the user wants to check their lifestyle activities inferred they just need to use any device having a web browser to view statistics (figure 2) and suggestions about their life style (though it is quite feasible to give suggestions and alerts on the users' mobile phone automatically).

Data Set and Training Data. Data from the 3D accelerometer, in the mobile phone, was taken at the sampling rate of 5Hz (this rate was chosen based on work in [11,12]), and GPS data is refreshed at the rate of 0.2Hz. Data for training and evaluating system was taken from 8 users with age from 6 to 67 (the app aims at office workers but we still use a variety of users' age, gender, and job to check the precision and generality of our approach), and the phone was carried in different

[4] `http://www.heatonresearch.com/encog`

Fig. 2. Statistics information, x-axis is activities, y-axis is time in minutes

ways such as in tight/loose pants pocket, jacket high/low pocket, hand-held bag, and shoulder bag. When the data was collected, each user was encouraged to do different activities in different styles like walk with/without shoes, run slowly/quickly, run with high/low steps. We have 4 long-term users who used the prototype for 1 to 3 weeks and others used it for 20 minutes. Data collected from each user is marked with a unique id and we call them *original data*. For long term analysis where only short term data was available, the original data from all users were combined to create new data which simulates virtual long-term users. We used both original and simulated data to evaluate our prototype. When data was collected, users were required to take note in their diaries for labelling the data sets.

We selected, as ANN training data, a set of 6,850 "typical" accelerometer data samples which was collected when users continuously performed four basic activities walking, running, driving and putting the phone still on a table. The accelerometer data are packed in windows of size 5 samples to be fed into the trained ANN. The output of the ANN, which is recognized basic activities, is further processed by a window sliding technique with a size of 7 and slides of step size 5. The activity which appears most often in the window will be the final result of the recognition phase using the ANN. This process has a recognition rate of 0.2 Hz and was used in [15]. These window sizes are chosen based on the assumption that: A user is said to perform a basic activity if it lasts for more than 5 seconds; this time hurdle can vary with specific applications. We also define the *Activity time* is the sampling time of the last sample of the data window used for activity recognition.

In practice, our prototype can transmit all this data through a mobile network. But because mobile phone signals may drop at some times, in some areas, and optimizing transmission is not our goal here, to evaluate our prototype we store data in files and stream them to our system for reasoning. In next version, we will use compressed data such as in the approach in [15].

Recognized Activities and Accuracy. Basically, the human daily activities which can be detected by using a single accelerometer are: walking, running, driving (or in transportation), and staying still; so, we call them basic activities.

HealthyLife detects these basic activities and combine these with GPS data to infer the users' place (where users are) and to infer complex activities. To evaluate the accuracy, we compare the activities detected by *HealthyLife* with the diary which user noted manually. Then for each activity, we divide the total time *HealthyLife* detects correctly over the total time the real activities happen.

Basic Activities. Because one of the major design feature is to let users be free in the way they carry their phone, this is a source of ambiguity in activity recognition. For example, we say the user is *softwalking* when the user walks barefoot or puts the phone in their suitcase or handbag. Figure 3 shows that softwalking and driving have very similar sensor data patterns. To deal with this ambiguity, we combine accelerometer data and GPS data via weak constraints as follows and use ambiguity reasoning as implemented in section 4.

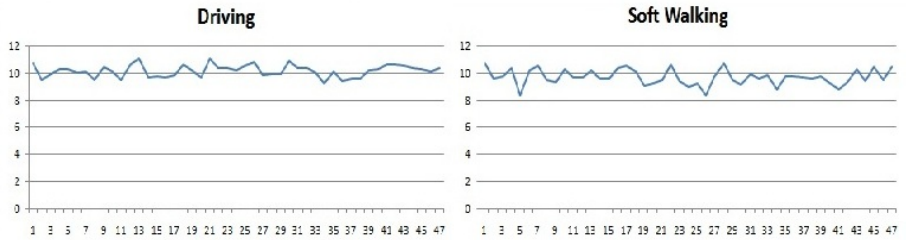

Fig. 3. Accelerometer patterns, x-axis is time, y-axis is A: force vector magnitude

```
:~ basicActivity(ActName), gps(GpsAccuracy, GpsSpeed),
   ActName = driving, GpsSpeed <= 2. [4:1]
:~ basicActivity(underCover), basicAct(driving). [4:1]
:~ basicActivity(stayingStill), gps(GpsAccuracy, GpsSpeed),
   #int(GpsSpeed), GpsSpeed > 1. [4:1]
```

In this code fragment, *underCover* means the user is staying indoor, or surrounded by high trees or buildings. [4 : 1] is the [*violation cost* : *priority level*] of the weak constraints which shouldn't happend.

The result of recognition without ambiguity reasoning is shown in tables 1, with ambiguity reasoning in table 2. Table 1 shows that, without ambiguity reasoning, when user was (soft) walking Healthy Life tends to recognize that user was driving. This is understandable as shown in figure 3. With ambiguity reasoning, the accuracy increased significantly from $\simeq 50\%$ to $\simeq 73\%$ for detecting (soft) walking. These two tables also show that ambiguity reasoning doesn't effect the accuracy of detecting activities which are not ambiguous (or don't have similar sensor data pattern) such as running and staying still.

Users' Current Place and Complex Activities. We detect users' location (their current place) and then combine that with detected basic activities to infer complex activities. To detect user's place (in a meaningful way as opposed to numerical coordinates), we use the Google API to find places (or Points of Interest) around the user and compare their addresses with the closest address from the user's position. The place which has a matching address is where the user is.

Table 1. Result without ambiguity reasoning

		Recognized			
		walk	**run**	**drive**	**stay still**
Actual	**walk**	50%	0%	50%	0%
	run	3%	97%	0%	0%
	drive	23%	0%	70%	7%
	stay still	0%	0%	0%	100%

Table 2. Result with ambiguity reasoning

		Recognized			
		walk	**run**	**drive**	**stay still**
Actual	**walk**	73%	0%	27%	0%
	run	2%	98%	0%	0%
	drive	7%	0%	88%	5%
	stay still	0%	0%	0%	100%

We use GPS data with a precision of $\leq 30m$ for detecting user places; a detected precision of > 30 implies that user is undercover like indoors or surrounded by tall buildings. A set of places which can be detected in *HealthyLife* is found in table 3. With big places (park, shopping mall) or well defined places (home, office), Healthy Life can recognize with high accuracy. For small area (bank, gym), the accuracy reduce quickly as there may be many other small places around. Note that with such knowledge, one can estimate how often the user goes to the gym as health related information, and also, when the user is at the gym, how active s/he was over that time s/he was there. We asked users to provide the address of their home and office to detect when the user is at home and at work and we realized the ability to detect these addresses automatically. The rules to detect user places have form as follows:

```
userAt(Place) :- closestAddress(Adress1,UserPosition),
place(Place, Address1), Address1 = Address2.
userAt(underCover,Time) :- gpsData(accuracy,Time), accuracy > 30.
```

Table 3. Detected User Places

Places	**Accuracy**
User at Home	100%
User at Work	100%
User at Gym	67%
User at Shopping mall	90% 90%
User at Park	100%
User at bank	51%

Table 4. Complex Activities & User Preference

Complex Activities	**Users Preference**
User active at work	Like stay indoor
User working late	Like stay at home
User working hard	Like shopping
User may be tired	Like nature
Need more workout	Like fitness
At risky activity	Driving much
Running at park	Seems to be fit

We can predict users' status based on a previous week's data and users' preferences (indicated in table 4) with rules of the form:

```
userWorkingHard :- userAtWork(Hours),avgWorkHours(AH),Hours > AH + 5.
userMayBeTired :- avgWorkHours(AH),userAtWork(Hours),Hours > AH + 10.
userWorkingLate :- userAtPlace(office,T),#int(T),lateWorkTime(T).
lateWorkTime(T) :- time(T), #int(T), T > 17. % 5pm.
runningAtPark :- userAtPlace(park), basicAct(running).
```

Transmission Feasibility. Optimizing bandwidth utilization was not our goal, but we examined the feasibility (not to compare with other algorithms) of transmitting *HealthyLife* data from the users' smart phone to our server. Every data record includes 2 characters of basic activity code, 10 of time stamp, 19x2 of latitude and longitude, 4 for accuracy, 3 for speed, 10 for GPS time, and 6 separators. To sum up, we need 73 characters/bytes to represent a data record.

If the system runs 24/7 at the data rate of 0.2Hz (one data record every 5 seconds) , everyday, *HealthyLife* needs to transmit 73 bytes x 0.2 Hz x 86,400 s/day = 1.2 MB/day or about 36 MB/month. Transmitting this amount of data is possible for a mobile phone network in Australia. For example, a $29 VirginMobile package has a quota of 250MB/month. If this amount of data becomes an issue, we could use the algorithm in [15] to optimize the transmission.

System Speed. After getting data from the user's smart phone (with the rate 0.2 Hz), our single ASR reasoner, as noted in 4, have a reasoning speed of 1Hz and so can service 5 users at the same time (this is easily scaled by having more reasoners and more machines).

Potential Applications. *HealthyLife* has potential for many applications for single users and groups of users - to infer group or collective activities from single users' activities. Single users can have statistics about their fitness status and daily activities automatically to organize their time and lifestyle better. This prototype also has the ability to connect to other smart systems such as smart house, smart office or a social network. If more users share their data (without any id information), people can become aware of community events (e.g., a community marathon) and friends' activities. For example, a user who wants to find a quiet area can avoid the park where there is a public event. *HealthyLife* can provides statistics data for health promotion bodies, a city council and businesses to service citizens and customers better. For example, local governments can invest more on sports infrastructure if there are more people doing exercise. More generally, companies can send advertisement messages to users, if allowed to, when users are stationary (say in a bus or train) where they are more likely to give attention.

6 Conclusion

HealthyLife differs from other work by automatically detecting users' basic and complex activities and also does ambiguity reasoning. We combine accelerometer data, GPS data (reverse geocoded into meaningful places) and weak constraints to perform ambiguity reasoning, and showed how this can improve real-life activity recognition performance. Besides detecting basic activities, *HealthyLife* is able to detect complex activities, which can be tracked for statistics for health-related purposes and rules can be used to map inferred activities and activity histories to suggestions for users, all within a logic-based rule framework.

References

1. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. In: SIGKDD Explor. Newsl., vol. 12(2), pp. 74–82. ACM, New York (2010)
2. Sport-Tracker, http://www.sports-tracker.com/blog/about/
3. Mattila, E., Parkka, J., Hermersdorf, M., Kaasinen, J., Vainio, J., Samposalo, K., Merilahti, J., Kolari, J., Kulju, M., Lappalainen, R., Korhonen, I.: Mobile Diary for Wellness Management–Results on Usage and Usability in Two User Studies. IEEE Transactions on Information Technology in Biomedicine 12(4), 501–512 (2008)
4. Sunny, C., David, W.M., Tammy, T., Mike, Y.C., Jon, F., Beverly, H., Predrag, K., Anthony, L., Louis, L., Ryan, L., Ian, S., James, A.L.: Activity sensing in the wild: a field trial of ubifit garden. In: Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (CHI 2008), pp. 1797–1806. ACM, New York (2008)
5. Washington Education, https://dada.cs.washington.edu/research/projects/aiweb/main/media/papers/UW-CSE-08-04-02.pdf
6. Alhamid, M.F., Saboune, J., Alamri, A., El Saddik, A.: Hamon: An activity recognition framework for health monitoring support at home. In: 2011 IEEE Instrumentation and Measurement Technology Conference (I2MTC), pp. 1–5 (May 2011)
7. Della Valle, E., Ceri, S., Barbieri, D.F., Braga, D., Campi, A.: A first step towards stream reasoning. In: Domingue, J., Fensel, D., Traverso, P. (eds.) FIS 2008. LNCS, vol. 5468, pp. 72–81. Springer, Heidelberg (2009)
8. Do, T.M., Loke, S.W., Liu, F.: Answer set programming for stream reasoning. In: Butz, C., Lingras, P. (eds.) Canadian AI 2011. LNCS, vol. 6657, pp. 104–109. Springer, Heidelberg (2011)
9. Michael, G., Vladimir, L.: The stable model semantics for logic programming. In: Kowalski, R., Bowen, K. (eds.) Proceedings of International Logic Programming Conference and Symposium, pp. 1070–1080 (1988)
10. Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Using mobile phones to determine transportation modes. ACM Trans. Sen. Netw. 6, 13:1–13:27 (2010)
11. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
12. Huynh, T., Schiele, B.: Analyzing features for activity recognition. In: Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies, pp. 159–163. ACM (2005)
13. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing 8, 36–66 (2012)
14. Predrag, K., Wanda, P.: Healthcare in the pocket: Mapping the space of mobile-phone health interventions. Journal of Biomedical Informatics 45(1), 184–198 (2012)
15. Jayaraman, P.P., Sinha, A., Sherchan, W., Krishnaswamy, S., Zaslavsky, A., Haghighi, P.D., Loke, S., Do, M.T.: Here-n-Now: A Framework for Context-Aware Mobile Crowdsensing. In: Proceedings of the Tenth International Conference on Pervasive Computing, UK (June 2012)