Demo: Automatically Generating Interesting Events with LifeJoin

Alvin Cheung
MIT CSAIL, Cambridge, MA
akcheung@csail.mit.edu

Arvind Thiagarajan
MIT CSAIL, Cambridge, MA
arvindt@csail.mit.edu

Samuel Madden
MIT CSAIL, Cambridge, MA
madden@csail.mit.edu

Abstract

This demo will showcase LifeJoin, a system that collects raw sensor data from phones and laptop computers to generate interesting events. Given the raw sensor data, LifeJoin implements a number of activity recognition algorithms to generate higher-level events. Furthermore, it uses supervised learning techniques to learn from users' feedback to generate only events of interest. In this demo, the audience will get to interact with the LifeJoin system and be able to examine the internals of LifeJoin.

Categories and Subject Descriptors:
H.4 [Information Systems Applications]: Miscellaneous
General Terms: Human Factors, Performance
Keywords: Sensor Data, Supervised Learning

1 Introduction

Modern phones and laptop computers have an abundance of sensors. In this demo, we will show a system called LifeJoin, which uses the sensing and recording capabilities on phones and laptop computers to automatically create and generate events regarding users' activities and encounters. LifeJoin helps users discover interesting facts that they share in common with their friends by joining event streams from multiple users. LifeJoin can then report such events to social media websites like Facebook and Twitter. For instance, events such as "Jogged along the Charles River from 4-5 with @smadden" can be generated by collecting Bluetooth proximity data, joining the GPS location data from phones, and resolving GPS locations to textual names on the map.

In addition to real-time status generation, automatically generating events from data collected from phones has many uses. For instance, when a tourist arrives at a new city, an application can mine the previously generated location events that were marked as interesting by her friends to suggest tourist attractions and restaurant venues.

To summarize, the key features of LifeJoin are:

1. It employs low-overhead data collection and transmission techniques to avoid consuming excess resources (e.g., energy and bandwidth) on user's devices. It automatically adjusts the types of data collected and the collection frequency based on the amount of remaining battery on the device and the users' interests.

2. It includes a number of activity recognition algorithms (e.g., walking, running, etc) that use the sensors on smartphones, such as accelerometer, gyroscope, and GPS to generate events. We are developing libraries to collect application statuses, and adapt activity recognition algorithms from the literature [1] for this purpose.

3. It implements algorithms to infer further events given the raw data collected from the devices. This event generation process is nontrivial. For example, consider inferring the

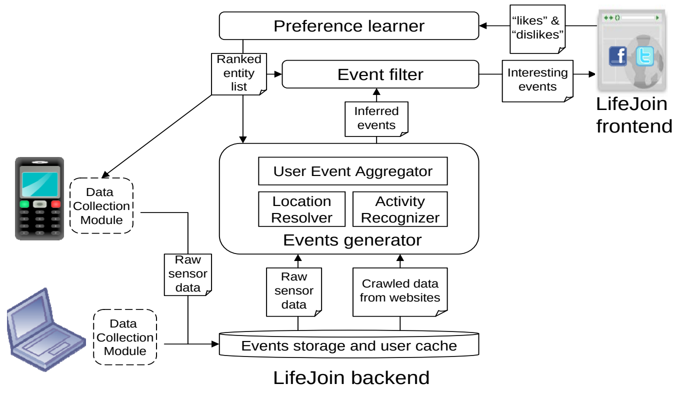


Figure 1. LifeJoin System Diagram

fact that "Arvind and his friend Alvin went to the Red Sox game at Fenway Park tonight." Assume we have collected GPS and Bluetooth data from Arvind's and Alvin's phones, to create this event LifeJoin would need to infer that:

- Arvind and Alvin were at Fenway Park, as detected from the GPS traces and a reverse-geocoding service.
- Knowing that there was a Red Sox game at Fenway Park tonight by crawling local events online.
- The two are friends from their friends' networks.
- They went to the event together, as inferred by Bluetooth proximity data.

All of the facts above require different algorithms to deduce, many of which are open research questions [2]. LifeJoin implements a number of these algorithms for event detection from the raw data collected on users' devices and websites.

4. Users typically are interested in only a subset of the generated events, for instance events about their close friends, or specific type of events such as exercises. Asking users to indicate all their interests is too cumbersome. Instead, LifeJoin implements a supervised learning algorithm to learn users' interests given their feedback (e.g., from clicks of the "like" button). Furthermore, the results from the learning algorithm are used to guide both data collection and event generation in order reduce device battery consumption. For instance, if no users are interested in reading the whereabouts of another user, LifeJoin will reduce the frequency or simply not collect any location data from that user's phone. Likewise, raw data from the devices are combined to generate only events that LifeJoin believes are of interest to users.

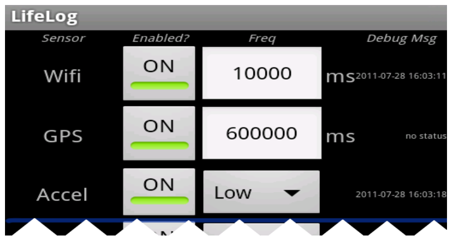
In the following, we first describe the components of LifeJoin, followed by several demonstration scenarios.

2 System Components

Fig. 1 shows the different components in LifeJoin, consisting of a low-overhead data collection component, a backend engine that includes the event generators and preference learner, and the frontend web application. We briefly describe each of these below.

Data Collection Module: To start using the LifeJoin service, the user installs the data collection application on their

SenSys'11, November 1–4, 2011, Seattle, WA, USA.
ACM 978-1-4503-0718-5/11/11



LifeLog

Sensor	Enabled?	Freq		Debug Msg
Wifi	ON	10000	ms	2011-07-28 16:03:11
GPS	ON	600000	ms	no status
Accel	ON	Low		2011-07-28 16:03:18

Figure 2. Android data collection configuration interface.

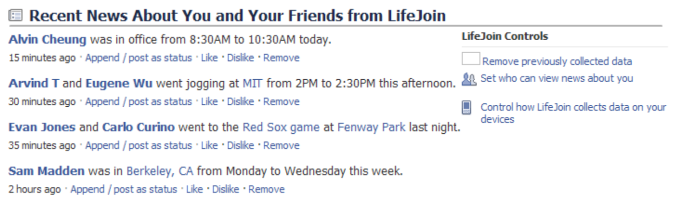
devices, such as phones and computers. The application collects different types of data that are available on the devices such as GPS geographical location data, accelerometer and gyro sensor readings, output from activity recognition algorithms, Bluetooth proximity data, web contents the user has accessed, etc. To protect privacy, the user can change the type of data and frequency of data collection on the module's interface. The collected data are summarized and sent to the LifeJoin backend. A screenshot of our interface for configuring data collection is shown in Figure 2.

Event Generators: The LifeJoin event generators process the raw data from the devices by integrating them with other available data. Each event generator maintains a cache for each user that stores the most recent events received (e.g., events received within the past week), and periodically crawls specific websites to detect local events. Our prototype currently includes a number of event generators. The location resolver takes in GPS readings collected from the devices and translate them into location events about known public locations. The activity recognizer uses accelerometer readings and location data to detect users' activity (such as walking, running, etc). Finally, the user aggregator groups together events about individual users that are common (e.g., events about two users who performed the task at the approximately the same time), and generates events of the form "<user name(s)> <action name(s)> at <location(s)> from <time> to <time>," which are then passed to the event filter. The event filter uses the results from the preference learner to decide which events the frontend should display.

Frontend: The frontend is the interface that LifeJoin users interact with. It is a mixed-initiative interface that displays a list of events about the user and her friends, along with a "like / dislike" option below each event for the user to express her preferences. The preferences are sent back to the preference learner to drive data and event generation.

To protect the user's privacy, the frontend allows the user to control when and what events should be generated from the raw data collected on her devices, remove previously collected data, and manage which of her friends can view the generated events published by LifeJoin.

Preference Learner: The preference learner receives the preferences from the frontend. Since the user does not explicitly inform the system about why she likes or dislikes a particular event, the learner needs to infer that information. For example, a user "like" the event "Sam went jogging at MIT from 2PM-3PM today" because she is interested in events about Sam, events about Sam jogging, or simply because she has not read events about Sam before. In order to infer that information, the learner first generates all possible pairs of entities for each event displayed. For instance, in the example event above the entities generated will be {Sam},



Recent News About You and Your Friends from LifeJoin

Alvin Cheung was in office from 8:30AM to 10:30AM today.
15 minutes ago · Append / post as status · Like · Dislike · Remove

Arvind T and Eugene Wu went jogging at MIT from 2PM to 2:30PM this afternoon.
30 minutes ago · Append / post as status · Like · Dislike · Remove

Evan Jones and Carlo Curino went to the Red Sox game at Fenway Park last night.
35 minutes ago · Append / post as status · Like · Dislike · Remove

Sam Madden was in Berkeley, CA from Monday to Wednesday this week.
2 hours ago · Append / post as status · Like · Dislike · Remove

LifeJoin Controls

Remove previously collected data

Set who can view news about you

Control how LifeJoin collects data on your devices

Figure 3. LifeJoin Frontend Screenshot

{jogging}, {MIT}, {Sam, jogging}, {Sam, MIT}, {Sam, 2PM-3PM}, etc. We imagine an event is interesting to a user either because she likes to see events about an entity (e.g., events about jogging), or it is an usual event for the user (e.g., a friend visiting an exotic place). In that respect, for each pair of user u and entity e, the learner maintains two scores: an "interest" score and a "novelty" score, where:

$$
\begin{aligned}
\text{interest}(u,e) &= [\text{likes}(u,e) - \text{dislikes}(u,e)]/\text{events}(u,e) \\
\text{novelty}(u,e) &= 1 - [\text{events}(u,e)/\text{events}(u,*)]
\end{aligned}
$$

here likes(u,e) returns the number of events containing e for which u has clicked "like" (vice versa for dislikes), and events(u,e) returns the number of events that were displayed to user u containing entity e. Finally, events$(u,*)$ returns the number of events that were ever displayed to user u. The score for an entity is a weighted sum of the interest and novelty scores, and the overall score for an event is the normalized sum of all the entity scores embedded in the event.

These scores are updated as more events are displayed to the user. The learner periodically ranks the scores for all entities and produces a ranked list, which is used to drive the operations of the other system components. For instance, the event generators use the list to decide which events to generate given the raw data. And, if the event filter receives too many events from the event generators at a given time, then it uses the ranked list to determine which events to send to the frontend and their relative ordering.

3 Demonstration Scenario

We will release a prototype version of the LifeJoin system before the conference and invite users to install them on their devices for data collection. Our demo team will run this application for several weeks as well. During the demo, we will replay the data that was collected by our demo team and show how the LifeJoin interface evolved during this period on two screens:

- One of the screens will show the LifeJoin frontend interface (Fig. 3). The audience will be able to view the incoming events as published by LifeJoin on the frontend, and be able to provide feedback to the system to see how their feedback changes the event generators.
- The other screen will look into the LifeJoin backend by showing the events being collected and streamed into the system from the devices, the event generators combining the collected data to create events, and the preference learner periodically producing the ranked list of entities. It will also show how the system makes use of the ranked list for optimization.

4 References

[1] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2), February 2010.

[2] A. Thiagarajan, J. P. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using gps-enabled smart-phones. In *Sensys*, 2010.