Automated
Personal Assistants

Kai A. Olsen
University of Bergen and Molde University College, Norway

Alessio Malizia
Universidad Carlos III de Madrid, Spain

Instead of just adding functions and new gadgets to current devices, we should ensure that users get the full benefits of the new technology.

You’re in the middle of a strange city. Your hotel should be nearby, but you can’t find it. There are two options. One, open an Internet connection on your smartphone, find the map service, input the city’s name, download a city map, change to a convenient map scale, type the hotel’s address, and let the GPS system lead you to your destination. Two, ask a passerby.

The easy choice is option two. That is, our smart devices can do the job, but in most cases they take too much effort. Keying comes at a cost. This is especially the case when we have to use an onscreen keyboard. But even when input is as simple as a button click, using small displays is time-consuming and irksome.

USING CONTEXT INFORMATION

Often, systems ask for input they could either find from available data or infer from contextual information. As an example, consider a situation that many of us have experienced.

You’re in a meeting that drags on. At some point, it’s clear that there’s no chance of catching the five o’clock plane back home. You’ll have to leave the meeting, get an Internet connection, log in to your airline’s website, give information such as your name and booking reference, and change the booking to a later flight. Some airlines even require you to make a phone call in this circumstance.

However, if the booking system could use contextual information, you’d be able to perform the whole operation without even interrupting the meeting. A text message to the airline saying “later flight” should suffice. The airline’s system should be able to identify you by phone number, retrieve the booking for this evening’s flight, and return a set of options for later flights, asking you to choose one. It could even book the next flight automatically, letting you change to another if this option isn’t suitable.

But this requires the airline to offer such an option. Instead of waiting for this to happen, we could use an agent running on our smartphone to change the flight based on data from the initial booking. This would be easy to do if the smartphone agent made the booking in the first place, but it should also be possible if our booking data is readily available, as in the cloud. The smartphone could implement this agent as an app, but embedding the agent in the operating system would perhaps be simpler.

A PERSONAL ASSISTANT

The idea of having a computer system that could act as a personal assistant has been around for many years. As early as 1987, Apple CEO John Scully described the “knowledge navigator,” a device that used software agents to assist the user (Odyssey: Pepsi to Apple ... A Journey of Adventure, Ideas and the Future, Harper & Row).

Several videos from Apple depicted this concept, envisioning the assistant as a bow-tied butler having human properties. The assistant had natural-sounding speech, speech recognition, and the ability to grasp the underlying semantics and actually understand what the user was saying.

Continued on page 110
The ideal interface doesn’t require any input but takes its cues from the available data.

However, these were only mock-ups; the real-world assistants were more primitive. A well-known example is the Office Assistant that came with Microsoft Office between 1998 and 2003, depicted in various ways, including as a paper clip and an Einstein caricature. This feature got a negative response from users; some went as far as to develop applications that allowed users to “shoot down” their assistant, which had the bad habit of popping up in unexpected situations, even when the function was turned off. Smithsonian Magazine called it “one of the worst software design blunders in the annals of computing” (R. Conniff, “What’s Behind a Smile?,” Aug. 2007, pp. 46-53).

The advice the Office Assistant offered was often irrelevant, downright silly, or too trivial to be of any use. The problem was that the feature took its clue from a skerrick of context, for example, just a keyword. For example, if you started by typing “Dear,” a message would appear saying, “It looks like you’re writing a letter. Would you like help?”

Today, we have more possibilities. Since we use our computers for most administrative tasks, from accessing our bank accounts to writing a shopping list, the assistant could have access to all relevant data. When it’s stored in the cloud, it becomes irrelevant if a smartphone or a PC was used to input the data. That is, the assistant could have access to phone data, text messages, e-mail, contact lists, social networks, and background information from the Web. By combining this data with the user’s location, an assistant running on a smartphone should be able to perform “intelligent” deductions in many situations.

Although speech generation and recognition might be useful in some cases, a smartphone keyboard and display can be used to communicate with the assistant. Thus, there’s no need to give the assistant human-like capabilities or for it to “understand” the user. Today, we can implement a “knowledge navigator” without the magic parts, leaving the role of the human to the user.

CONTEXTUAL DATA
What if we offer the assistant the word “hotel?” Is this a meaningful command? Of course, there are many different interpretations, including:

- I want to book a hotel.
- I need to view a booking, change it, or delete it.
- I need directions to a hotel.

However, with contextual data, things become clearer. In most cases, a simple word such as “hotel” can be given a meaningful interpretation when such data is available.

Let’s assume that the system has an overview of all your bookings, your current location, and your home address. If you’ve booked a hotel in Rome starting today and your current location is at the airport or any place in Rome, the system could retrieve the booking; offer the hotel’s name, address, and phone number; and give directions from your current location. If you have a rental car booking, the directions should be for travel by car. If you’re near the hotel, the assistant could provide directions for walking to it. If you’re farther away, it could offer suggestions for finding public transportation.

If you’re away from your hometown and don’t have a hotel reservation, the system should offer a selection of nearby hotels that have a room available and are within the price range you normally select. In some circumstances you might want to translate the word “hotel” into another language. The system could offer this as a secondary option, or you might have to add more keywords.

In the early days of computing, an interface was an empty line on a teletype or a blinking “>” on a display. Nowadays, this has been replaced by apps and their form-based input. The drawback is that the user must choose the right app for the function, fill out forms, provide codes, and so on. Perhaps we should reintroduce the command line interface, but this time let an assistant parse the data.

AUTOMATIC ASSISTANCE
Some time ago, we stayed at a large airport hotel. There were six floors and as many elevators, all in the same area. When we went from the reception area to the elevators, the door of one elevator was open so that we could enter and press the button for our floor without having to wait. When we went down for dinner, we also found an elevator waiting. With a simple addition to the control program that made an elevator available at each floor, this hotel was able to offer guests a convenient service.

With some extension, this idea also can be used when elevators are busy or when there are fewer elevators than floors. A “here” command for bringing an elevator to the reception floor could be executed automatically when the receptionist hands a key-card to a guest. The same action could be performed when guests retrieve the key-card as they leave their room. Thus, a better service can be offered just by using contextual data.

This is the ideal interface—one that doesn’t require any input but takes its cues from the available data.

As another example, assume that you’re on your way to the bus stop. With data including the time of day,
your location, the bus stop location, and the bus route, the assistant could automatically present the bus schedule on the phone display, or better still, it could count down the minutes until your bus arrives. You could explicitly enter the route data, or the assistant might be able to deduce it from your commuting history. The data could be presented on an “I feel lucky” display on the phone. That is, you could just take a look at this display to get all the information you need. But if you pass the bus stop on your way to the grocery, the phone would clear the data. When you enter the grocery, the system would, of course, present your shopping list.

In yet another example, imagine yourself driving to the airport. The weather is bad, traffic is dense, and the cars ahead are moving along slowly. You wonder if you’ll catch the flight. Hopefully, there might be other delays such as a change in the flight schedule because of the bad weather. With a context-sensitive interface, you would only need to look at the smartphone’s “I feel lucky” screen. The system should already have deduced what information you might need, such as an updated departure time for your flight or directions that offer a way around the traffic congestion.

**DATA STANDARDS**

Storing data in the cloud offers the opportunity to have access to all relevant information. This is necessary to understand the user’s context. The assistant will interpret a “later flight” or “hotel” command incorrectly if it doesn’t have access to all bookings. Today, however, when we use computers for nearly everything, this data is available. But access to data isn’t enough. It must also be presented in a useful form.

This is the great challenge. It’s possible to extract the necessary information from an e-mail confirmation of a booking, such as the hotel name, arrival date, departure date, reference number, and so on using a Web service such as TripIT (www.tripit.com). However, storing the booking information in a standardized format in the cloud would be more convenient.

Thus, to get the full advantage of having an assistant, we need a more formalized infrastructure than we have today. This will require agreements on standards (S. Ortiz Jr., “The Problem with Cloud-Computing Standardization,” *Computer*, July 2011, pp. 13-16), but standardization isn’t easy to achieve.

Although technical, economic, and political issues can hamper the work, there are several success stories. In Norway, for example, banks have developed a common ID system for online banking. Combining this with a national interbank system has kept transaction costs very low, and it’s also convenient for users with accounts in several banks.

The assistant’s interpretation of the context might not always be correct because it doesn’t have the most recent data or it makes the wrong deductions. However, since the system is serving as the user’s personal assistant, even if these errors are annoying, they’ll usually be recognized. The fact that most advice is on current events, especially what’s happening right now, will also be of use in detecting erroneous deductions.

Initially, like its predecessors, the smartphone evolution focused on new functions. Now, there’s a need to consider the user interface. This time, the focus isn’t on organizing the input or developing convenient forms or menu systems, but on finding ways to avoid input. A personal assistant can do this for us. As we’ve seen, a human-like assistant isn’t necessary. In many cases, having background data from the cloud, as well as the time of day and the user’s location, should be sufficient.

As computing professionals, perhaps our job in the next decade isn’t just to add functions and new gadgets to current devices, but to ensure that users get the full benefits of the new technology. Ideally, we should seek “free” solutions so that the user can get valuable information without having to pay based on button clicks.}

**Kai A. Olsen** is a professor at the University of Bergen and Molde University College in Norway as well as an adjunct professor at the School of Information Services, University of Pittsburgh. Contact him at kai.olsen@himolde.no.

**Alessio Malizia** is an associate professor in the Computer Science Department at Universidad Carlos III de Madrid, Spain. Contact him at alessio.malizia@uc3m.es.

---

**Editor:** Neville Holmes, School of Computing and Information Systems, University of Tasmania; neville.holmes@utas.edu.au

---

**Selected CS articles and columns are available for free at**