Ruhi Sarikaya

The Technology Behind Personal Digital Assistants

An overview of the system architecture and key components

We have long envisioned that one day computers will understand natural language and anticipate what we need, when and where we need it, and proactively complete tasks on our behalf. As computers get smaller and more pervasive, how humans interact with them is becoming a crucial issue. Despite numerous attempts over the past 30 years to make language understanding (LU) an effective and robust natural user interface for computer interaction, success has been limited and scoped to applications that were not particularly central to everyday use. However, speech recognition and machine learning have continued to be refined, and structured data served by applications and content providers has emerged. These advances, along with increased computational power, have broadened the application of natural LU to a wide spectrum of everyday tasks that are central to a user's productivity. We believe that as computers become smaller and more ubiquitous [e.g., wearables and Internet of Things (IoT)], and the number of applications increases, both system-initiated and user-initiated task completion across various applications and web services will become indispensable for personal life management and work productivity. In this article, we give an overview of personal digital assistants (PDAs); describe the system architecture, key components, and technology behind them; and discuss their future potential to fully redefine human–computer interaction.

Digital Object Identifier 10.1109/MSP.2016.2617341
Date of publication: 11 January 2017

Introduction

We are living in the mobile Internet computing cycle. During the past decade, mobile devices have experienced unprecedented growth. According to Statista [65], there are currently more than 4.6 billion mobile phone users in the world, and the number is expected to grow even more moving forward. With this phenomenal increase in volume came technical sophistication and improved capabilities of mobile devices (both on the hardware and software sides), particularly around applications and web services where users can complete a wide array of tasks. As the need and expectation to do more grew, despite improvements, a limited natural user interface has remained as one of the major bottlenecks in interacting with these devices. PDAs (also known as *virtual assistants*) precisely target this problem and have the promise of enhancing a user's productivity by either proactively providing the information the user needs in the right context (i.e., time and place) or reactively answering a user's questions and completing tasks through natural language. Tasks can be related to device functionality, applications, or web services.

Research on PDA technology, however, started much earlier than the emergence of mobile devices. Over the last 20 years, researchers have investigated personalized virtual assistant agents targeting specific domains, including tourism, elder care, device control, and home and office applications [1]–[5]. However, attempts at bringing them to market earlier have failed because of their limited utility.

PDAs have the promise of enhancing a user's productivity by either proactively providing the information the user needs in the right context or reactively answering a user's questions and completing tasks through natural language.

Over the past five years, there has been tremendous investment in PDA technology by both small and big technology companies. Siri [17], [66], Google Now [67], Cortana [68], and Alexa [69] are the major personal assistants in the market today, and they provide proactive and/or reactive assistance to the user. *Proactive assistance* refers to the agent taking an action to assist the user without the user's explicit request. *Reactive assistance* refers to the agent responding to the user's voice or typed command to assist him or her. The number of smartphone users using PDAs increased from 30% in 2013 to 65% in 2015 [70], indicating increased adoption.

PDAs have become a key capability in most smartphones. They are now also deployed in tablets, laptops, desktop PCs, and headless devices (e.g., Amazon Echo), and some are also even integrated into operating systems. These agents are designed to be personal; they know their user's profile, whereabouts, schedules, and so forth. They can proactively start interactions with their user through notifications and system-initiated questions or reactively respond to user requests. User–PDA interactions typically take place via natural language, where the user speaks to the agent as if he or she were speaking to a real human assistant.

PDAs

What is a PDA?

A PDA is a metalayer of intelligence that sits on top of other services and applications and performs actions using these services and applications to fulfill the user's intent. A user's intent could be explicit, where the user commands the system to perform an action, or it could be inferred, where the agent notifies or makes suggestions upon evaluation of one or more triggering conditions it has been tracking. PDAs make use of some core set of technologies, such as machine learning, speech recognition, LU, question answering (QA), dialog management (DM), language generation (LG), text-to-speech (TTS) synthesis, data mining, analytics, inference, and personalization.

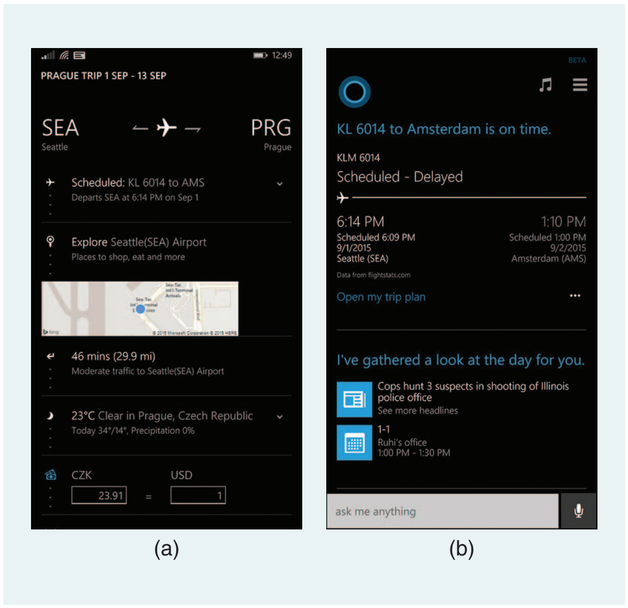
Why do we need PDAs?

PDAs are built to help the user get things done (e.g., setting up an alarm/reminder/meeting, taking notes, creating lists) and provide easy access to personal/external structured data, web services, and applications (e.g., finding the user's documents, locating a place, making reservations, playing music). They also assist the user in his or her daily schedule and routine by serving notifications and alerts based on contextual information, such as time, user location, and feeds/information produced by various web services, given the user's interests (e.g., commute alerts to/from work, meeting reminders, concert suggestions). Collectively, these functionalities are expected to make the user more productive in managing his or her work and personal life.

For example, airline travel is a commonly supported scenario by most PDAs. If the user has booked a flight and received a confirmation e-mail along with an itinerary, the PDA scans the e-mail, extracts the flight information, and stores it on the service. On the day of travel, the PDA computes the user's current location using the global positioning system (GPS) on the device, checks the traffic conditions to the airport, and tells the user when to leave for the airport. It also checks the flight status and updates the user if there is a delay, using a flight card as shown in Figure 1. Additionally, it provides weather forecasts for the destination as well as currency conversion rates. Typically, a user has to use multiple applications to go through each of these steps to find out the needed information that is listed on the cards for the travel. None of these atomic steps is significant in isolation, but stitching them together can potentially mark a breakthrough in usefulness to the user. This is the key promise of PDAs.

What is personal about PDAs?

PDAs are expected to be personal. Ideally, the PDA is expected to know who its user is, what its user does, its user's interests, what its user needs, and when and where its user needs it. Despite numerous efforts over the past 20 years to make


(a)

PRAGUE TRIP 1 SEP - 13 SEP

SEA Seattle → PRG Prague

Scheduled: KL 6014 to AMS
Departs SEA at 6:14 PM on Sep 1

Explore Seattle(SEA) Airport
Places to shop, eat and more

46 mins (29.9 mi)
Moderate traffic to Seattle(SEA) Airport

23°C Clear in Prague, Czech Republic
Today 34°/14°, Precipitation 0%

CZK 23.91 = USD 1

(b)

KL 6014 to Amsterdam is on time.

KLM 6014
Scheduled - Delayed

6:14 PM
Scheduled 6:09 PM
9/1/2015
Seattle (SEA)

1:10 PM
Scheduled 1:00 PM
9/2/2015
Amsterdam (AMS)

Data from flightstats.com

Open my trip plan

I've gathered a look at the day for you.

Cops hunt 3 suspects in shooting of Illinois police office
See more headlines

1-1
Ruhi's office
1:00 PM - 1:30 PM

ask me anything

FIGURE 1. The proactive flight cards. (a) Summary and suggestions for the trip. (b) Flight details for the first leg.

human–computer interaction personal (particularly around web search), personalization has remained largely broken until recently, not only for PDAs but also for general human–computer interaction due to the four main gaps.

1) *Data*: The system had a limited amount of data to properly model the user and his or her interests. It understands the user based on the experience it provides and the feedback loop it uses.
2) *Computing*: The limited computing power and machine learning were not adequate for modeling the complexity of user behavior.
3) *Interest*: There has been conflict of interest between the user, the platform, and those who pay for the user's attention. The system has not been necessarily prioritizing the user's interests over these other actors.
4) *Content/action*: The system does not support the actions the user wants to perform or does not have the content to serve the user's interests.

During the past seven years, two primary changes have occurred that allowed PDAs to be personal: 1) the increased number of device sensors on mobile phones and 2) the quality and quantity of the user data and digital artifacts (on the device and/or in the cloud) coming from the web services and applications the user accesses. As a result of these changes, it is now possible to represent a user along four axes:

1) *user profile*: user's name, age, gender, parental status, profession, employer, home, work, people in his or her close circle (people graph), favorite places, files, documents, music, photos, and interests explicitly provided by the user
2) *digital activity*: digital artifacts (e.g., calendar, e-mail, social media activity, web searches) on applications and web services
3) *space*: physical location of the user
4) *time*: time at which a specific digital or physical activity takes place.

These four dimensions, when considered together, blend the physical with the digital world and open up new possibilities for powerful inferences and deep user understanding. Inevitably, managing and protecting privacy and security of the user data and information is a major concern, and what has been done in that space is critical, but it is outside the scope of this article.

Mobile device sensors

The computational power and capabilities of mobile phones are increasing every year. The number of built-in sensors on smartphones (e.g., Samsung Galaxy) more than tripled during the past five years [71]. Smartphone sensors measure motion/orientation, GPS coordinates, and many other user and environmental conditions. For example, a device's gravity sensor provides data to infer complex user gestures and motions, such as shake, swing, or rotation. The rich high-precision data coming out of these sensors are made available through application programming interfaces (APIs) and are used in numerous applications and scenarios. The information is sensitive, as it is personal and contextual. Making it available opens up new research areas like fitness and health applications or opens up new solutions to already existing problems [6]. User experiences that currently exist can also be enhanced by the available data. For example, using activity detection, the PDA can hold the incoming call or send a short message service (SMS) text message to the caller if the user is biking or it can turn up the volume if the user is climbing stairs/walking.

The three main mobile platforms (Android, iPhone operating system, Windows) support four broad categories of sensors on mobile devices.

1) *Motion*: This sensor set includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors. They measure acceleration and rotational forces along three axes. They measure movement and orientation of the device.
2) *Environmental*: These sensors measure various environmental conditions, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, magnetometers, and thermometers.
3) *Position and location*: These sensors measure the physical position and location of a device. This category includes orientation sensors and magnetometers. The magnetometer can determine the rotation of the device relative to magnetic north. It can also detect magnetic fields around the device. GPS and Wi-Fi (not really a sensor in the traditional sense) determine the location of the device.
4) *Proximity*: This sensor detects whether the phone is brought near the face during a phone call. This functionality disables the touch screen, preventing inadvertent input to the phone from the user's face and can also save battery power.

System architecture

The scenarios that the PDAs support can be divided into two main categories: 1) proactive and 2) reactive assistance. The conceptual agent architecture designed to support these two modes of assistance is shown in Figure 2. The system

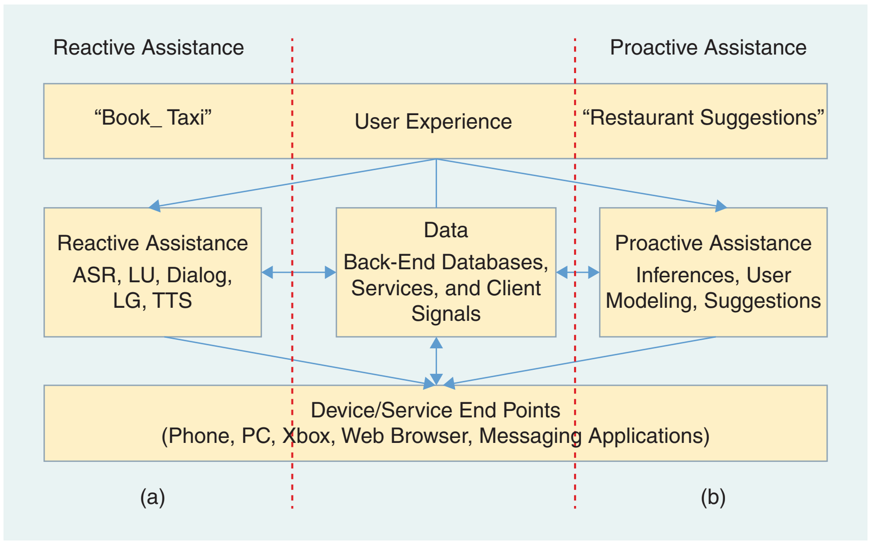

Reactive Assistance
Proactive Assistance
"Book_ Taxi"
User Experience
"Restaurant Suggestions"
Reactive Assistance
ASR, LU, Dialog, LG, TTS
Data
Back-End Databases, Services, and Client Signals
Proactive Assistance
Inferences, User Modeling, Suggestions
Device/Service End Points
(Phone, PC, Xbox, Web Browser, Messaging Applications)
(a)
(b)

FIGURE 2. The personal digital agent architecture for (a) reactive assistance and (b) anticipatory computing.

architecture depicts proactive and reactive user experiences, data, and service end points. Reactive assistance is shown in Figure 2(a), where the user issues an explicit natural language command (e.g., "book me a taxi") to the agent. The user request is handled through a set of reactive assistance components, such as speech recognition, LU, and DM. The data coming from various back ends, and applications are served to the user according to the constraints specified in the natural language query. The experience (reactive and/or proactive) can be served in one or more of the different device or service end points.

Proactive assistance [Figure 2(b)] involves anticipatory computing, where the personal digital agent does things in a contextual manner (i.e., at the right time and place) that it expects is valuable to the user without an explicit user request. Proactive assistance makes use of inference, user modeling, and ranking to power experiences. Back-end data, device, applications, and web services signals are leveraged for proactive inference and triggering.

Even though proactive and reactive parts of the current PDA architectures are built in isolation, in principle they can use a single architecture to enable both types of experiences. In fact, most proactive scenarios have reactive extensions and vice versa. For example, if the user makes a restaurant reservation (reactively), the agent may (proactively) suggest a movie after the dinner or may offer to book a cab to take the user to the restaurant. Data and context are shared between the two assistance modes. Next, we focus on the proactive system architecture and the components that power proactive scenarios.

Even though proactive and reactive parts of the current PDA architectures are built in isolation, in principle they can use a single architecture to enable both types of experiences.

Proactive assistance

Proactive assistance is based on the theory of proactivity that describes user desires and a model of helpfulness [7]. The goal is to provide assistance to automate tasks or further the user's interests for things he or she cares about, all within context, without explicit user request [8]. To achieve that, the agent is designed to possess a set of attributes; it should be valuable in that it advances the user's interests and tasks, while not interfering with the user's own activities or attention unless it has the user's explicit approval. It should be unimposing. The agent should be transparent in what it knows about the user. It should be anticipatory and know the future needs of the user and bring opportunities to the surface. The agent should also continuously learn and refine its decisions from the feedback signals it receives regarding the actions it takes. These principles put the user at the center, and the agent's actions are considered valuable only if they ultimately add value for the user. Proactive assistance operates on the proactivity continuum [31], which ranges from zero to full automation, allowing for the following scenarios:

- do it yourself (no help from the agent)
- user tells the agent what to pay attention to (notifications/alerts)
- agent infers user's habits/patterns and makes suggestions (inference/suggestions)
- agent makes decisions and takes actions (full autonomy on task decisions/executions).

Most of the currently supported proactive scenarios are notifications/alerts and suggestions. Even though there is some preliminary work, none of the agents in production supports autonomous decision making and action taking on behalf of the user without confirmation.

The proactive agent system architecture is shown in Figure 3. Signals coming from web services, device sensors, and the user's profile are processed, where processing includes parsing, enriching, and filtering to merge device and service data. The next step is aggregation, which joins the processed data streams through time and space (i.e., location) about the user's whereabouts and actions/tasks done at specific times and places. This step blends the physical and digital worlds and allows for powerful inferences that capture repetitive behavior and events in both worlds. The signals are used to make inferences and train machine-learned models for modeling the user and his or her interests. The same set of signals is also used to set rules for notifications and alerts the user wants the agent to serve. The models and rule recipes are deployed to a run time environment. Once proactive scenarios are deployed in production, capturing and feeding back user behavior signals regarding notifications, alerts, and

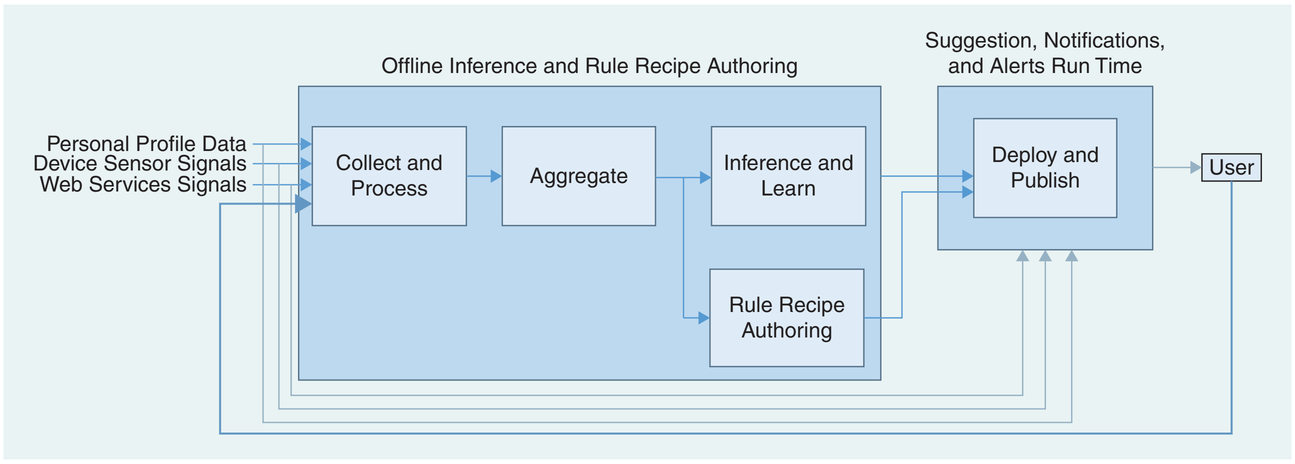


FIGURE 3. The proactive assistance system architecture.

suggestions are essential for the proactive agent to learn and adapt to the user.

Notifications and alerts

In the notifications/alerts category, the system allows the user to set rules to define the triggers for certain actions. If the triggering condition evaluates to TRUE, the action is executed. The rules are defined over a set of signals. These signals are produced by an information channel that can be evaluated by the proactive agent. These channels represent many types of information, such as date/time and location, as well as constantly updated data feeds generated by various web services, which include weather, sports, news, finance, and entertainment. For example, one can create rules to obtain an alert when the Seattle Seahawks score a touchdown. A user can set a rule to be reminded of his or her mother's birthday. It is also possible to combine these signals to formulate more complex triggering rules. For example, a specific flight departure time, a user's physical location, and a commute time to the airport can all be used to trigger an alert that reminds the user that it is time to leave for the airport. Once the trigger rule is set, the agent monitors the signals from the corresponding information channels to evaluate the rule. If the rule evaluates to TRUE, the agent takes an action. The actions are communicated to the user in a target device-specific manner, which could be a proactive entity card, SMS, or even a phone call. This type of proactive agent programming falls under the if-then recipes [60], in which simple rules allow users to control many aspects of their digital life.

Inference and suggestions

In the suggestions category, the agent infers the user's habits and routines by reasoning over his or her past behavior and makes a personalized recommendation to the user with the goal of furthering the user's interest. For example, knowing that the user watched comedy movies featuring a particular actor in the past, the agent may suggest a new comedy movie featuring that same actor in the future. The agent can also suggest new experiences based on the logical sequencing of different yet related (through time or location) events. For instance, if the user made a restaurant reservation in a metropolitan city downtown, the agent may suggest nearby parking places. Through inference, the agent can learn certain facts about the user, by reasoning over the user's whereabouts and movement patterns through time and location. For example, the user's home and work location could be inferred by joining GPS data with time over several weeks. If the user is spending most or all of his or her time between 9 a.m. and 5 p.m. during weekdays at a specific location over several weeks, that is likely to be the user's work location. Likewise, the user's commute hours between home and work could also be inferred from combining home and work location with the GPS data during the likely morning and evening commute hours over several weeks. This inference is used to proactively show the traffic commute cards around the time the user typically commutes to/from work (or home).

The key questions here are determining the type of suggestion and when to do it, because there is an associated cost with the suggestion (if the action, relevance, or timing is wrong). To get around the cold start problem (if the agent does not have access to the user's past activity through a feedback loop or the user is accessing the PDA for the first time), the user is also given the ability to teach the agent his or her interests from a precompiled list of topics, including news, sports, finance, technology, dining, and entertainment. The decision for taking a proactive action is driven by a machine-learned model, given the costs and benefits as constraints. The machine-learned model combines a set of information in the user's profile, demographic and content-based profiles, and online user behavior signals (such as click through, dwell time, and dismissal), along with the user's recent relevant activity (e.g., similar content searches), which are captured in the history variable h in (1). This is used to model whether a specific user u will like the specific suggested entity e. Standard machine-learning techniques, such as maximum entropy models [35], gradient boosted decision trees [55], and deep learning techniques [48], are used to incorporate both user-specific online and offline signals to estimate the probability that the user is expected to like the suggested entity

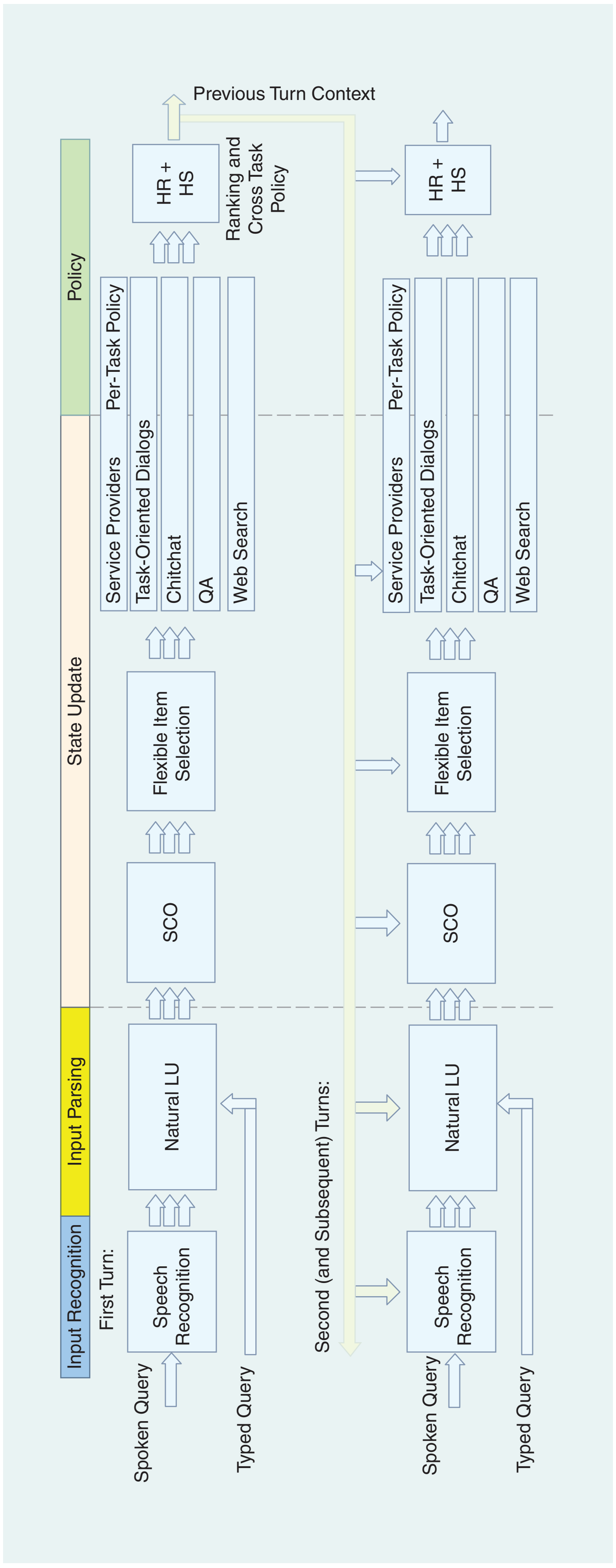


FIGURE 4. The reactive assistance architecture: conversational understanding system architecture.

(i.e., content). Within the maximum entropy modeling framework, the probability is computed using

$$P(o|\, e, u, h) = \frac{e^{\sum_i \lambda_i f_i(o,e,u,h)}}{\sum_{o'} e^{\sum_{j \in J} \lambda_j f_j(o',e,u,h)}}. \qquad (1)$$

Here, o denotes outcome (e.g., like or dislike). Notice that the denominator includes a sum over all possible outcomes, o', which is essentially a normalization factor for probabilities to sum to 1. The functions f_i are usually referred to as *feature functions*, or simply *features*. These binary feature functions are given as

$$f_i(o,e,u,h) = \begin{cases} 1, & \text{if } o = o_i \text{ and } q_i(e,u,h) = 1 \\ 0, & \text{otherwise} \end{cases}, \qquad (2)$$

where o_i is the outcome associated with feature f_i and $q_i(e,u,h)$ is an indicator function on the user, suggestion, and history. The model parameters λ_i are learned on labeled data, which capture the user's response (e.g., like or dislike) for the suggested content in the past.

The specific features of the suggested entity (e) include the estimated value of suggestion type, value of suggestion instance, timing of the suggestion, cost of mistake, cost of interruption, and urgency (time sensitivity) of the suggestion. Learned thresholds on the model outputs govern the number of suggestions of each type that can be displayed concurrently, the maximum frequency for suggestions of each type, and the permitted or prohibited modalities of each suggestion type. The thresholds for acting, asking, suggesting, or doing nothing are established from a range of default values according to user-stated advice and elicited initial preferences from the configuration wizard.

Reactive assistance

Reactive assistance is traditionally known as the *conversational understanding system*. The conversational understanding system for PDAs spans a wide spectrum of domains, including goal-/task-oriented dialogs [53], [16], chitchat, QA [37], and classical web search answers. The conversational understanding system also handles additional input modalities besides speech, such as typing and/or touch. Each of these domains is commonly referred to as an *answer* (A). Some of the domains involve device functionality (e.g., alarm, SMS, calling, note, reminder), while others may involve web services and applications (e.g., directions to a particular location, movie hours at a theater, factoids, stock prices, weather).

The reactive assistance system architecture is shown in Figure 4. The user submits a request to the PDA to perform a task or seek information using one of the modalities, and the agent interprets the request and generates a response. For voice queries, the first step is to recognize the spoken words [9], [10]. The LU component takes the speech transcription (or the text input if the user types) and performs a semantic analysis to determine the underlying user intent [11]–[14], [17]. The user's intent could be related to information search, QA, chitchat, or task-oriented specialist dialogs. Because PDAs support multidomain and multiturn interactions, multiple alternate semantic analyses (typically at least one for each domain) are generated in parallel for late binding on the user's intent [15]. These semantic analyses are sent to the dialog state update component, which includes slot carryover (SCO) [38], flexible item selection from a list [39], knowledge fetch from the service providers, and dialog hypothesis generation [15], [16]. Note that in this framework, we consider chitchat, QA, and web search as an additional set of LU domains. All the dialog hypotheses are ranked by the hypothesis ranking (HR) module. The top hypothesis is selected by the hypothesis selection (HS) module by taking the provider responses (i.e., knowledge results) into account [18]. The top dialog hypothesis (along with the ranked dialog hypothesis distribution) is the input to the dialog policy component, which determines the system response based on the scenario and business logic constraints. Typically, for voice input, the agent speaks the natural language response via the TTS synthesis engine [19].

The reactive assistance behavior is governed by (3). The goal of the reactive agent is to provide the best system response $\hat{R}$ to a given user query, Q. The system response, R, consists of a dialog act, which includes system action (e.g., information to be displayed, question to be asked, or action to be executed), natural language prompt, and a card in which the response is displayed

$$\hat{R} = \underset{R}{\operatorname{argmax}} \{P(R \mid Q, B_{A_1}, \ldots, B_{A_N}, \bar{B})\}, \tag{3}$$

where B_{A_1} denotes the current belief about the dialog state of the answer A_1 (e.g., weather, alarm, places, reminder, sports, etc.) after processing query Q, and $\bar{B}$ shows the system's belief about the state of the interaction across all answers for the current session. In practice, it is hard to solve (3). Instead, a suboptimal solution can be achieved with the assumption that, given the query Q and the beliefs for the dialog states of the individual answers A_1 through A_N, the per answer response is conditionally independent

$$\hat{R} = \underset{R \in R_1, \ldots, R_N}{\operatorname{argmax}} \{P(R \mid Q, B_{A_1}, \bar{B}), \ldots, P(R \mid Q, B_{A_N}, \bar{B})\}. \tag{4}$$

Here, $P(R \mid Q, B_{A_i}, \bar{B})$ denotes the probability the system assigns to response (R) generated by answer A_i, given the answer's belief about its dialog state and the system's belief, $\bar{B}$. This formulation allows the individual answers to manage their own dialog state and generate their own responses in parallel. Therefore, it is possible to scale to many domains and answers without substantially increasing the overall system response latency. The HR component operates as a metalayer, arbitrating between different answer responses, given its belief (i.e., $\bar{B}$) about the state of the overall interaction [15]. Next, we will briefly describe the key components in Figure 4.

Speech recognition

The speech recognition component maps the human speech represented in acoustic signals to a sequence of words represented in text. Let X denote the acoustic observations in the form of feature vector sequence and Q be the corresponding word sequence (i.e., query). The speech recognition decoder chooses the word sequence, $\hat{Q}$, with the maximum a posteriori probability according to the fundamental equation of speech recognition [9]:

$$\hat{Q} = \underset{Q}{\operatorname{argmax}} P(X \mid Q) P(Q), \tag{5}$$

where $P(X \mid Q)$ and $P(Q)$ are the probabilities generated by the acoustic and language models, respectively. Traditionally, speech recognition systems are trained to optimize the lexical form. However, displaying the grammatically and semantically correct version of the output (i.e., display form) has become an important requirement for PDAs, because it makes it easy for the user to infer whether the system correctly heard and recognized the spoken query. For example, the following two speech recognition outputs are lexically equivalent:

- how is the traffic on u s one oh one (lexical form)
- how is the traffic on US 101 (display form).

However, proper tokenization in the second hypothesis provides a valuable hint that the agent understood what the user meant. Typically, the tokenization is applied as a separate post-processing module after running the speech recognition decoder.

In recent years, advances in deep learning and its application to speech recognition have dramatically improved state-of-the-art speech recognition accuracy [9], [10], [20], [21]. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These advances played a key role in the adoption of PDAs by a large number of users making it a mainstream product.

LU

The problem of LU for PDAs is a multidomain, multiturn, contextual query understanding [17], [22]–[25], subject to the constraints of the back-end data sources and the applications in terms of the filters they support and actions they execute. These constraints are represented in a schema. In practice, while LU semantically parses and analyzes the query, it does not do so according to a natural LU theory [26]; rather, parsing and analysis are done according to the specific user experience and scenarios to be supported. This is where the semantic schema comes into play, as it captures the constraints of the back-end knowledge sources and service APIs, while allowing free form of natural language expression to represent different user intents in an unambiguous manner.

There are two main approaches to LU: rule based and machine learned [11], [32]. The rule-based approach is about hand authoring a set of rules to semantically parse the query [27]. It can also be used for addressing the errors and disfluencies introduced by a speech recognizer [28], [49]. State-of-the-art systems use machine-learned models for LU [12], [17], [23], [24]. In a commonly used LU architecture, a query is first classified into one of the supported domains (or a catch-all domain such as web). Typically, support vector machines, boosted decision trees, maximum entropy models [35], or neural networks [13], [33], [34] are used for modeling. These classifiers are trained in either binary or multiclass mode depending on the design choices [12], [23], [24]. Under each domain, there are a domain-specific intent and slot model. Intents and slots can be shared across different domains. A domain can be considered as a collection of related intents, which do not have any conflict. For example, a weather domain contains check_weather and get_weather_stats intents. Intent detection uses the same machine-learning techniques listed previously, and it is framed as a multiclass classification problem. Slot tagging is considered as a sequence classification problem. Conditional random fields, maximum entropy Markov models [35], and, more recently, deep learning techniques [12], [24] are used for slot tagging. LU models are trained in a supervised fashion using labeled data and properly weighted lexicons [36]. When a user issues a query, domain and intent classifiers are run to determine the domain and intent of the query, and the slot tagger tags semantic slots. Tagged slots are resolved into canonical values, and in some cases multiple slots are combined into a parameter. Parameters are used either to fetch results from the knowledge back end or to invoke an application API.

The goal of the LU component (for answer A_i) is to convert each input query into a set of semantic frames, F_{A_i}, given the context represented in the current beliefs about the dialog state B_{A_i}, that is, we seek

$$\hat{F}_{A_i} = \operatorname*{argmax}_{F} \{P(F|\hat{Q}, B_{A_i})\}. \tag{6}$$

Semantic frame, F_{A_i}, for answer A_i encapsulates the semantic meaning of a query with a tuple of domain, intent, and slot list:

$$\langle \text{DOMAIN, INTENT, SLOTS} \rangle.$$

The slots are a list of key-value pairs:

$$\langle \text{SLOT\_NAME, SLOT\_VALUE} \rangle.$$

The data structure for semantic frame, shown in Figure 5, is used to combine the different pieces of semantic analysis

```
<Domain score="0.8956">reminder</Domain>
     <Intent score="0.7949">create_single_reminder</intent>
   <Slots>
      <reminder_text ="call my mom" />
      <start_time <RawValue="9 am">
              <PropertyGroup Name="timex3">
                  <Property Name="value" Value="am" />
                  <Property Name="comment" Value="am" />
              </PropertyGroup>
      </start_time>
   </Slots>
```

FIGURE 5. The semantic frame for the query: "remind me to call my mom at 9 a.m."

generated by the domain, intent, and slot models to represent the complete semantic understanding of the query.

Some of the key PDA experiences that are handled are multiturn in nature. Without using contextual information, the queries could be ambiguous and potentially interpreted differently. For example:

- (turn 1) how is the weather in New York (weather)
- (turn 2) what about the weekend (weather).

Here is another scenario, where we observe the exact same query in the second turn:

- (turn 1) how is my schedule (calendar)
- (turn 2) what about the weekend (calendar).

Interpreting the follow-up queries in isolation is difficult, as they are ambiguous and require context for proper interpretation. LU models are built in a contextual manner to solve this problem [15], [22]–[24]. To handle multiturn interactions, in addition to basic domain, intent, and slot models, one needs to build context carryover models to help with state tracking [38] and on-screen selection models [39], as shown in Figure 4, for selecting items from a list of results presented to the user in follow-up turns.

QA

Because users are expecting PDAs to answer any question, open-domain QA [56] is another scenario that is handled by all PDAs to differing degrees. Examples of open-domain factoid QA include the following questions:

- How old is Bill Gates?
- How tall is Mount Everest?
- Who directed *Avatar*?

The answers to these questions are precise short phrases.

- Bill Gates is 60 years old.
- Mount Everest is 8,848 meters high.
- James Cameron directed the movie *Avatar*.

QA has had a long history [29] and has seen rapid advancement in the past decade, spurred by government-funded programs that required system building, experimentation, and evaluation of systems [30]. Advancements in search engine technology, such as query formulation and query-document analysis through click logs, have also contributed to innovation in QA [52].

The system architecture for a typical QA system is shown in Figure 6. For a given query directed at a PDA, the QA system first classifies the query into one of the question types (typically ten to 15). Note that the query may not be a question. Even if it is a question, it may not be supported by the QA system. These categories are also included as part of the question classification step. The answer-candidate generation step processes the question, generates various alternate formulations of it, and queries the knowledge sources, which include a knowledge graph, web documents, Wikipedia, and search engines. The answer-candidate ranking step extracts a number of features for each question/answer pair and applies a ranking model to rank and assign confidences to each answer candidate [56]. The overarching principles of QA systems are massive parallelism, confidence estimation, and integration of shallow and deep knowledge from many knowledge sources [37].

Knowledge back ends

A significant part of PDA scenarios is about accessing knowledge and entities. For example, when a user asks for a factoid regarding a movie or director, LU models tag such slots (i.e., facets) as movie name, release date, actors, and directors. Slots are used to build a query sent to these knowledge bases to fetch the relevant entity and relationships for which the user is looking. These knowledge bases store factual information in the form of entities and their relationships, covering many domains (people, places, sports, business, etc.) [57]. The entities and their relationships are organized using the World Wide Web Consortium Resource Description Framework (RDF) [40]. An RDF semantic knowledge base (also known as a *semantic knowledge graph*) represents information using triples of the form subject-predicate-object, where in graph form, the predicate is an edge linking an entity (the subject) to its attributes or another related entity, as seen in Figure 7. A popular open-source RDF semantic knowledge base is Freebase [46]. Other RDF semantic knowledge bases that are much larger in size are Facebook's Open Graph, Google's Knowledge Graph, and Microsoft's Satori. Both Google's Knowledge Graph and Microsoft's Satori knowledge graph have over 1 billion entities and many more entity relationships. They power entity-related results that are generated by Google's and Microsoft Bing's search engines. Recently, there has been a surge of interest in exploiting these knowledge sources, especially the RDF semantic knowledge bases, to reduce the manual work required for expanding conversational systems to cover new domains, intents, or slots [41]–[43], [51] or improving existing experiences [44], [45].

DM and policy

Many of the reactive scenarios enabled in PDAs require spoken DM to handle a wide range of tasks and domains. DM is at the bottom of the reactive stack, where all the information from upstream components is consolidated and the final decision about the system response is made and communicated to the user.

Much of the research on spoken dialog systems in academia has targeted single-domain applications [47], [50], where the problem of accurately tracking the user's goal (e.g., finding restaurants that satisfy a number of user constraints) has received considerable attention in the literature [53]. The primary line of research has been the statistical modeling of uncertainties and ambiguities encountered by dialog managers due to speech recognition and LU errors along with ambiguity in natural language expressions. Included among the most successful statistical approaches are graphical models that are concerned with decision making with delayed rewards [53]. However, large-scale production systems such as PDAs pose a different set of problems. The large number of supported domains, integration of task-oriented dialogs, QA, chitchat, and web answers and managing conversation in a coherent way pose new challenges

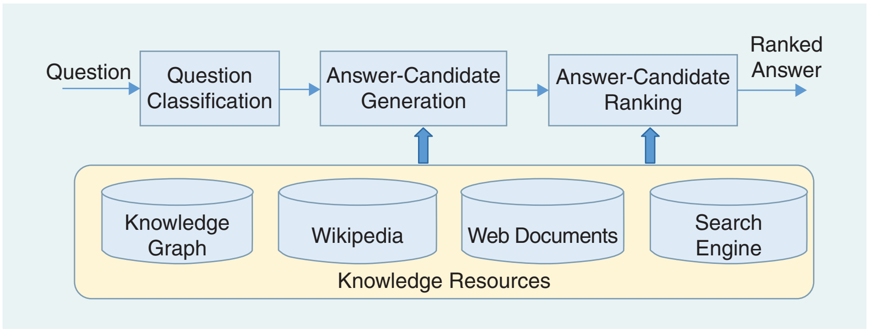


FIGURE 6. The QA system architecture.

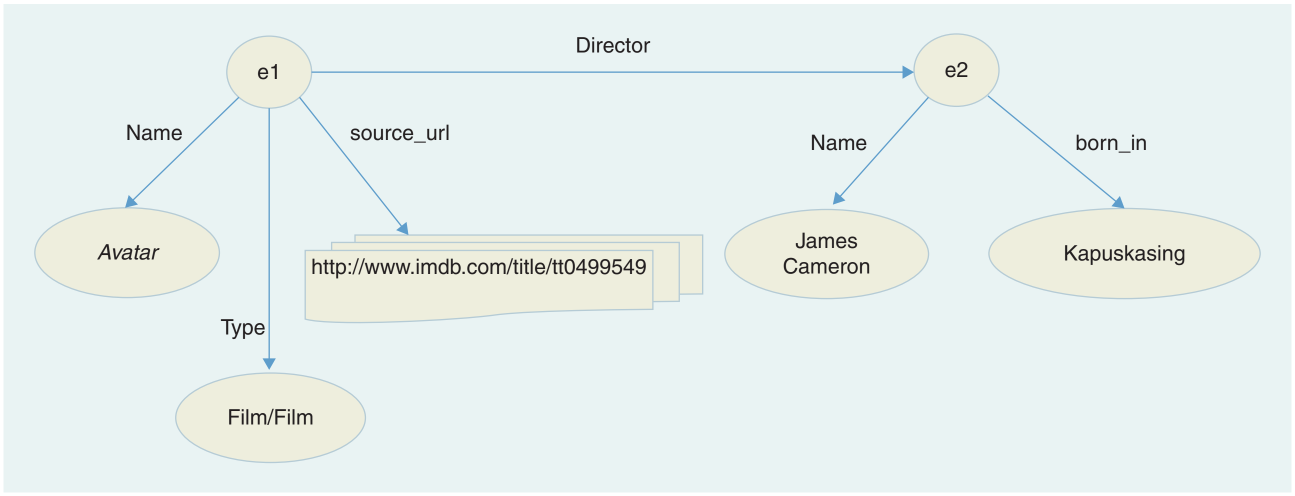


FIGURE 7. An example of part of a semantic knowledge graph representing the relationships, described as RDF triples, between the entities James Cameron (e1) and the film *Avatar* (e2).

[25]. Mixing different modalities to complete the tasks is another challenging area for PDAs.

In PDAs, a dialog manager supports execution of a variable number of goal-oriented tasks [16], [54]. Tasks are defined in terms of information to be collected from the user, corresponding LG prompts, and interfaces to resources (such as data hosted in external services and applications) that will execute actions on behalf of the user. As shown in Figure 4, at each turn, the dialog state is updated, taking into consideration the multiple LU results across different turns. SCO [38] does contextual carry over of slots from previous turns, using a combination of rules and machine-learned models with lexical and structural features from the current and previous turn utterances. Flexible item selection uses task-independent, machine-learned models [39] to handle disambiguation turns where the user is asked to select between a number of possible items. The task updater module is responsible for applying both task-independent and task-specific dialog state updates. Task-dependent processing is driven by a set of configuration files, or task forms, with each form encapsulating the definition of one task. Using the task forms, this module initiates new tasks, retrieves information from knowledge sources, and applies data transformations (e.g., canonicalization). Data transformations and knowledge source lookups are performed using resolvers [16], [54]. Dialog policy execution is split into task-specific and global policy. The per-task policy consists of analyzing the state of each task currently in progress and suggesting a dialog act to execute. The dialog acts include show results, disambiguation, prompt for missing value, prompt for no results found, start over, go back, cancel, confirm, complete the task, and so forth, in accordance with the ranked semantic frame output. The output of the task updater module is a set of dialog hypotheses representing alternative states or dialog actions for each task in progress. The dialog hypotheses are ranked using HR [15], [18], which generates a ranked order and score for each hypothesis. This acts as a pseudobelief distribution over the possible dialog/task states. HS policy selects a top hypothesis based on contextual signals, such as the previous turn task, rank order, and scores as well as business logic.

HR uses an implementation of LambdaMart [55] to rank hypotheses. Previously, various approaches have also been presented for reranking for spoken LU [58] but have focused on single-domain applications.

LG

Once the system response is determined by the dialog manager, it is communicated to the user in a natural way. If the query is a speech query, a spoken system response is returned. If the query is typed, there is no spoken system response but rather a natural language system response that the user sees on the screen as a card. In either case, the natural LG component receives the dialog output in the form of a system response along with the dialog state, which encapsulates the state of the interaction between the user and system (e.g., current turn identification, whether the user has prompted for the same information before) and generates a natural and grammatical utterance to convey the system response. There are a number of factors that feed into the LG design, such as information presentation, presenting enough information (to give a good overview of the state of the task) versus keeping the utterances short and understandable, handling error states, and repeated tries [59].

There are three main approaches to LG: 1) template-based, 2) rule-based (linguistic), and 3) corpus-based approaches [59], [61]. Most of the PDAs use the template-based approach, because comparatively less effort is needed to develop and maintain the templates. The template-based LG module typically starts out from a semantic representation (e.g., semantic frame), generating "QFC in Redmond is open from 7:00 a.m. to 10:00 p.m." in response to the user query, "Is QFC in Redmond open today?" For example, STOREHOURS: [PLACENAME("QFC"), LOCATION("REDMOND")] associates it directly with a template, such as [PLACENAME] in [LOCATION] is open from [TIMEBEGIN] to [TIMEEND], where the gaps represented by [PLACENAME], [LOCATION], [TIMEBEGIN], and [TIMEEND] are filled by looking up the relevant information in the dialog state. The TTS engine consumes the LG output and synthesizes the text into speech [19].

It is generally difficult to empirically evaluate the quality of proactive and reactive user experiences for PDAs.

Metrics and measurement for PDAs

It is generally difficult to empirically evaluate the quality of proactive and reactive user experiences for PDAs. PDAs are complex systems with many components in the system stack, spanning client and multiple cloud services, and it is hard to separate any one component from the rest. Each component has its own functional and quality metrics. Metrics could be offline, measured with sampled data sets, or online, measured with actual user traffic of the live system.

Component metrics

The following are the offline component metrics tracked individually to improve the quality of each component. They are computed using a set of ground truths generated by human judgments:

- *speech recognition*: word error rate (WER), sentence error rate, slot WER, keyword WER, semantic WER (for non-search tasks)
- *LU*: domain accuracy, intent accuracy, precision/recall, slot F1 measure, semantic frame accuracy
- *dialog*: dialog state tracking accuracy—in a distribution (e.g., N-best list) of dialog state hypotheses, percent accuracy of the top-ranked hypothesis, selection accuracy, SCO accuracy
- *LG:* mean opinion score (MOS) of the LG quality, bilingual evaluation understudy score
- *knowledge*: knowledge relevance, coverage, precision/recall
- TTS *synthesis*: MOS, intelligibility, expressiveness

- *proactive suggestions/notifications*: precision/recall.

In Figure 8, we show LU accuracy for domain, intent, slot tagging (F1 measure), and semantic frame for some of the reactive experiences supported in Cortana for a sample training and test data set [25], [45]. The domain, intent, and slot accuracy is around 90% across domains. The average semantic frame accuracy is 82%. Note that semantic frame combines domain, intent, and slots; therefore, errors in these components contribute to the semantic frame error rate.

In Figure 9, we show the impact of HR in picking the right hypothesis over the LU model confidences. HR improves in picking the correct semantic frame by 2%. HR has the full view of all the LU analyses coming from different domains, and so it can arbitrate between competing hypotheses.

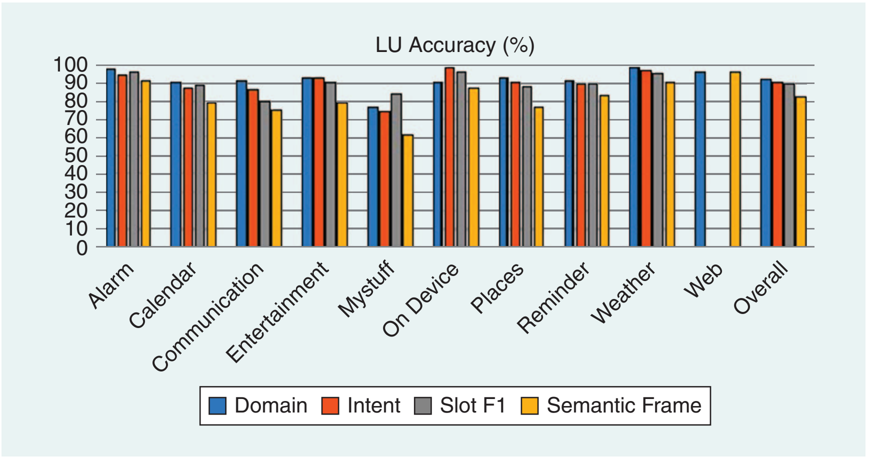


FIGURE 8. The LU domain, intent, slot, and semantic frame accuracy.

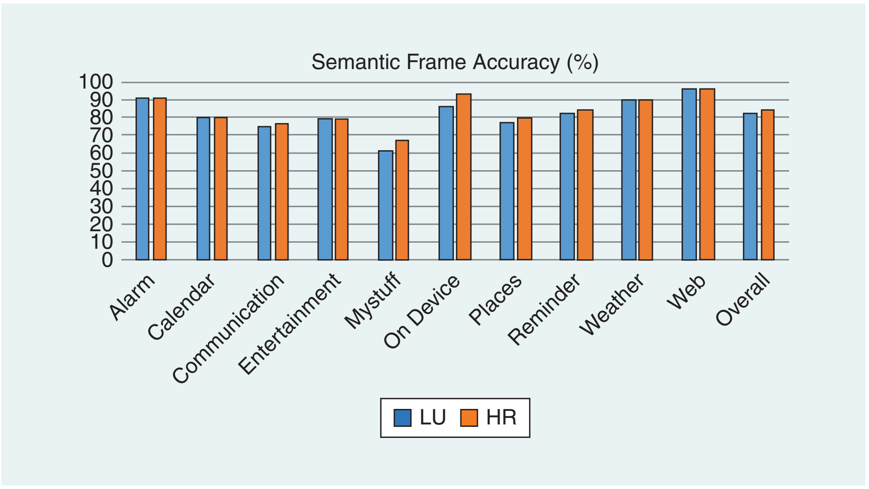
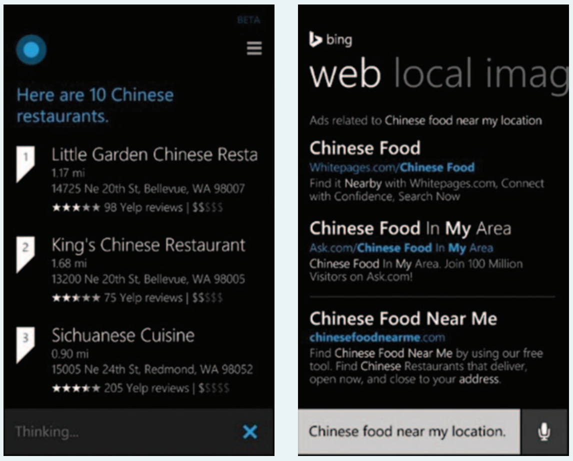


FIGURE 9. The semantic frame accuracy for LU versus HR.

End-to-end quality metrics

The fact that individual component accuracies are high may not mean that the PDA, as a product, has high accuracy. There are several factors contributing to this. For example, speech recognition may not be accurate even if LU is accurate, and knowledge results may not be relevant. There may be operational service reliability issues, back-end availability, network communication issues, robustness of wake-up word detection, and so forth, which all contribute toward quality of the user experience with the product. Moreover, a user's intent could be understood by the LU component, but the underlying application or service may not support that intent. Integrating different client and service components is a challenging software engineering problem, as it uncovers numerous scenario, design, service, and client shortcomings, which take the most time in improving the system. Therefore, end-to-end (E2E) product quality metrics are critical for the success of the product, as they correlate well with the actual user experience. They are also used for evaluating the contribution of the individual components to the overall product experience based on the analysis of how much each component contributes to the E2E error rates. Some of these metrics are as follows.

- *E2E accuracy* is measured through human judgment of query–response pairs on a five-point scale, where the user is shown a screenshot containing the system response, similar to the ones in Figure 10. Human judges assign a score between 1 (terrible) and 5 (perfect) to each


BETA

Here are 10 Chinese restaurants.

1 Little Garden Chinese Resta
1.17 mi
14725 Ne 20th St, Bellevue, WA 98007
★★★★★ 98 Yelp reviews | $$$$$

2 King's Chinese Restaurant
1.68 mi
13200 Ne 20th St, Bellevue, WA 98005
★★★★★ 75 Yelp reviews | $$$$$

3 Sichuanese Cuisine
0.90 mi
15005 Ne 24th St, Redmond, WA 98052
★★★★★ 205 Yelp reviews | $$$$$

Thinking...

bing

web local imag

Ads related to Chinese food near my location

Chinese Food
Whitepages.com/Chinese Food
Find it Nearby with Whitepages.com, Connect with Confidence, Search Now

Chinese Food In My Area
Ask.com/Chinese Food In My Area
Chinese Food In My Area. Join 100 Million Visitors on Ask.com!

Chinese Food Near Me
chinesefoodnearme.com
Find Chinese Food Near Me by using our free tool. Find Chinese Restaurants that deliver, open now, and close to your address.

Chinese food near my location.

(a) (b)

FIGURE 10. The system responses for the query "Chinese food near my location." (a) The correct result from the places domain and (b) the web search result.

query response pair. Success includes the ratings of 3 (okay), 4 (pretty good), and 5 (perfect).

- *Side-by-side* (*SBS*) compares system A with system B, where the two systems could be the same system (or competitive systems) at different points in time differing in updates and improvements. SBS is an A/B test on a five- or seven-point scale, where human judges pick one system over the other based on the results shown to the judge. For example, in Figure 10, we show two PDA system responses for the query "Chinese food near my location." In an SBS evaluation, the human judges compare the two responses as to whether the left/right response is better than the other on a scale of $+3$ to -3, where 0 shows that they are equal. SBS is a more sensitive metric compared to E2E accuracy.
- *Online user/system behavior-based metric* measures the user satisfaction and dissatisfaction with the PDA experiences. It is a model that uses a set of feedback signals from the user and the system that correlates with the quality of the user experience. The signals include actions executed, query reformulation, total elapsed time for task completion, landing page dwell time, click through, start over, cancel, click back rates, and latency.

It is quite challenging to evaluate PDAs, as they provide a wide range of experiences, including voice commands, task completion, chitchat, QA, and web search. Therefore, success/failure signals for online measurement could be quite different [62]. An instance where no click has occurred on the screen (i.e., abandonment) in one experience may mean user satisfaction (e.g., showing a correct weather card), but it may mean dissatisfaction in another domain (e.g., showing a list of restaurants and prompting the user to select), where the user leaves satisfied in the former but dissatisfied in the latter. There are also additional metrics used for business and overall product success, including user count, daily/monthly average users, sessions per user, query volume, unique query count, and number of proactive page views.

In Figure 11, we show the E2E query–response pair accuracy. In the figure, E2E Success* denotes the accuracy after leaving out the use cases the scenario is not designed to handle in the first place. For example, the user wants to delete an alarm on the PDA, but the scenario is not supported by design. Instead, the PDA shows either an irrelevant web search result or invokes the alarm application.

In Figure 12, we show the distribution of user dissatisfaction with regard to the sources of error across different components of the system. The numbers are based on feedback of about 10,000 real users. LU along with unsupported system action (e.g., user wants to delete an alarm but system does not support that action) are the biggest sources of user dissatisfaction. Frequent fallback to web search (i.e., text links in search results) when the system does not a have precise answer, a lack of LG (for scenarios where the user expects the system to talk), and speech recognition errors are the main buckets of user dissatisfaction with PDAs.

Technology and user experience challenges

While the functionality and types of tasks a PDA can perform are quite diverse and users find great value in using them, there are still a number of user experience and technical challenges that have yet to be addressed properly. We categorize these challenges into the following groups.

User experience challenges

User experience challenges include the challenges in the following list.

- *Operation errors*: There is a discrepancy between the user's mental model of the PDA's functionality and scenario coverage versus the actual PDA functionality. Users are often unaware of the total extent of the operations a PDA can perform. Users may not understand how to use the PDA application or what they need to say to get the result they desire. Current user interfaces lack the ability to

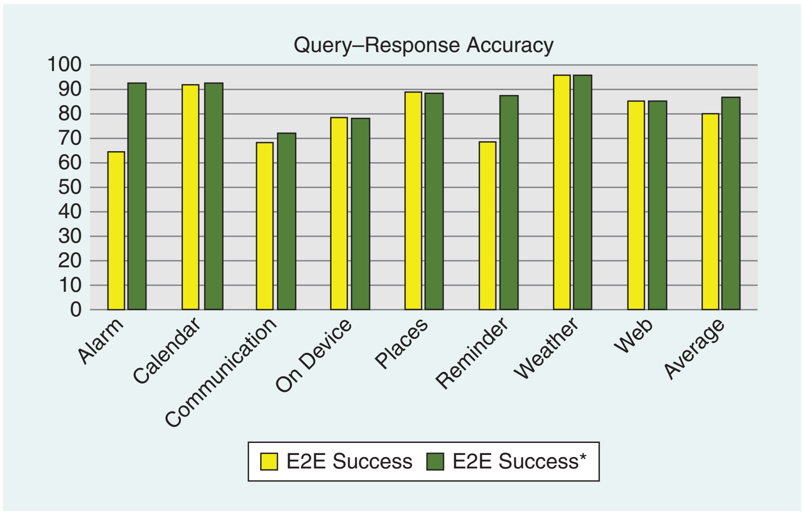


FIGURE 11. The E2E query–response accuracy. E2E Success* denotes the accuracy after leaving out the use cases that the scenario is not designed to handle in the first place.

provide sufficient information about how to use the system and intuitive sequence of operations to complete a task. PDAs are also not able to adapt to different user profiles and their way of operations.

- *Lack of competence*: PDAs are not at a level to reliably decide when to help the user, what to help the user with, and how to help him or her. This also creates a trust issue between the user and the agent, and the user may not feel comfortable delegating a task to the agent.
- *Privacy and security*: Privacy and security of the user's data and profile are a concern for users. Questions such as how much a PDA can/should know about its user and what the control mechanisms are remain open.

Technical challenges

Technical challenges include

- *Experience scaling*: It is critical to integrate third-party applications and services into PDAs to scale both reactive and proactive experiences that can be handled by PDAs. Building the right tools and infrastructure to easily enable such integration is an open problem. User feedback shows that unsupported experiences are one of the single biggest sources of user dissatisfaction. For example, a user wants to be able book a taxi or calculate the mortgage payments for a house using speech with his or her PDA, but these, and many more scenarios, may not be supported by the PDA.
- *Speech recognition challenges*: Despite all the recent progress in speech recognition with the application of deep learning techniques, issues such as background noise, speaker accent, Bluetooth, side speech, pocket dial, and unintentional wake up remain to be addressed [9]. To truly fulfill its promise, a PDA should recognize all the personal words that the user cares about. This includes any name (not just English), any place, and any thing (e.g., user's contact list), essentially leading to an open-domain, unlimited vocabulary speech recognition problem.
- *LU challenges*: Domain scaling to cover many more domains, high-quality LU model development, and continual refinement with feedback loop data are the main challenges. Building reusable models across different tasks is an important problem to solve as well.
- *DM*: Heterogeneous knowledge back ends and application interfaces are a bottleneck for expanding the domains and tasks a PDA can cover. The APIs for different applications/services for the actions they perform as well as data/knowledge back ends are not standardized and require custom interface work and query building.
- *Locale/market expansion*: Building a proactive or reactive experience, not for English but for other languages and markets, is another open problem. This requires reusing or building all the resources (e.g., data, content, and models) and capabilities for new locales and markets.
- *Different device/end points*: Even though smartphones were the initial target device for deploying PDAs, soon it became evident that the underlying intelligence and agent capability should be highlighted in other devices and end points. For example, Cortana started with the phone, but it is now made available in PCs and tablets and even on Xbox. This in turn creates another problem; not all experiences make sense for a given device, or the same query may be interpreted differently on different devices. For example, users cannot send SMS from a PC or Xbox using PDAs, but they can do it on a phone. The query "go home" could mean "get driving directions to home" on the phone experience, but it may mean "go to shell" on the Xbox device.
- *Service challenges*: PDAs use numerous services to enable a given scenario. There are also software engineering and service challenges that impact the overall user experience. For example, if the latency for handling a user request is too high, it reflects negatively on the user experience. In fact, instability may even stop users from using the scenario altogether. Likewise, users expect high reliability and availability (e.g., > 99.9%) from the services handling the requests.

All of these are requirements and constraints that influence the system design.

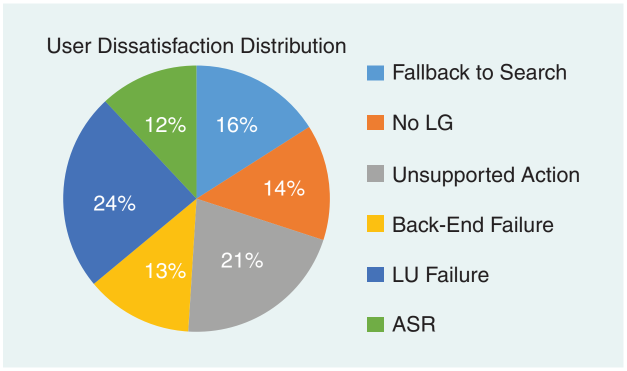


FIGURE 12. The E2E overall user dissatisfaction distribution over different components.

Moving forward

Research on human work habits and task management [3], [63], [64] shows that people usually complete all their important tasks yet may fail to successfully complete tasks with soft deadlines or may forget less-critical details. In the short term, PDAs can provide great utility by becoming the digital memory that users can depend on for help with completion of everyday tasks. In fact, it is these scenarios that are used most by the users (e.g., reminders, meetings, and some proactive notifications and alerts).

Because time is a critical asset, improving personalized time management and utility through proactive and reactive task delegation and completion seems to be a plausible and desirable long-term goal for PDAs. In the future, it is the scalable and seamless third-party integration that can substantially increase the scenario and experience coverage and determine whether PDAs will fulfill the promise of a true personal assistant that users can depend upon to manage their personal and work life, effectively making them more productive. The walls between applications may start to break down if PDAs achieve app/service composition to complete new tasks in a scalable way.

PDAs will surface on many different devices and environments. This will create new signal processing challenges, such as accurate speaker separation and tracking in multispeaker environments (e.g., home, car), robustness with respect to different device types, and robust speech recognition across age, gender, and accent. Advances in algorithms, signal processing, and machine learning would be needed to solve these problems.

On the industry front, the investment and competition in PDA technology will keep increasing over the next decade. It is seen by some that PDAs may set the balance of power in the next phase of the Internet, if it becomes the gateway to applications and services with the proliferation of IoT devices. It is too early to call it an inflection point for PDA technology. It is likely that true natural language human–computer interaction with gadgets may take another decade to be second nature.

Acknowledgments

I would like to thank the past and present members of the Language Understanding and Dialog Systems Group at Microsoft, who built the conversational understanding and DM capabilities of Cortana: Alex Rochette, Asli Celikyilmaz, Beatriz Diaz Acosta, Chandra Akkiraju, Daniel Boies, Danko Panic, Derek Liu, Divya Jetley, Diamond Bishop, Elizabeth Krawczyk, Gabrielle Knight, Hisami Suzuki, Jean-Phillipe Robichaud, John Nave, Khushboo Aggarwal, Kjel Larsen, Logan Stromberg, Minwoo Jeong, Nikhil Ramesh, Omar Zia Khan, Paul Crook, Puyang Xu, Rachel Morton, Ravi Bikkula, Roman Holenstein, Roy Tan, Steve Kofsky, Tasos Anastasakos, Vasiliy Radostev, Vipul Agarwal, Young-Bum Kim, and Zhaleh Feizollahi.

Author

Ruhi Sarikaya (rsarikay@amazon.com) is the director of applied science at Amazon. He received his B.S. degree from Bilkent University, Ankara, Turkey, in 1995; his M.S. degree from Clemson University, South Carolina, in 1997; and his Ph.D. degree from Duke University Durham, North Carolina, in 2001, all in electrical and computer engineering. He was a principal science manager at Microsoft from 2011 to 2016, where he founded and managed the team that built language understanding and dialog management capabilities of Cortana and Xbox One. Before Microsoft, he was with IBM Research for ten years. Prior to joining IBM in 2001, he was a researcher at the University of Colorado at Boulder for two years. He has authored more than 100 technical papers and has more than 60 issued/pending patents. He is a Senior Member of the IEEE.

References

[1] T. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience with a learning personal assistant," *Commun. ACM*, vol. 37, no. 7, pp. 80–91, 1994.

[2] H. Chalupsky, Y. Gil, C. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. Russ, and M. Tambe, "Electric elves: Agent technology for supporting human organizations," *AI Mag.*, vol. 23, no. 2, pp. 11–24, 2002.

[3] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, et al., "An intelligent personal assistant for task and time management," *AI Mag.*, vol. 28, no. 2, pp. 47–61, 2007.

[4] SRI International. (2003–2009). CALO: Cognitive Assistant that Learns and Organizes. [Online]. Available: pal.sri.com

[5] N. Yorke-Smith, S. Saadati, K. L. Myers, and D. N. Morley, "The design of a proactive personal agent for task management," *Int. J. Artif. Intell. Tools*, vol. 21, no. 1, 2012.

[6] S. Sohrab, M. Zhang, C. J. Karr, S. M. Schueller, M. E. Corden, K. P. Kording, and D. C. Mohr, "Mobile phone sensor correlates of depressive symptom severity in daily-life behavior: An exploratory study," *J. Med. Internet Res.*, vol. 17, no. 7, p. e175, 2015.

[7] S. Schiaffino and A. Amandi, "User-interface agent interaction: Personalization issues," *Int. J. Human-Computer Studies*, vol. 60, no. 1, pp. 129–148, 2004.

[8] P. Maes, "Agents that reduce work and information overload," *J. ACM*, vol. 37, no. 7, pp. 30–40, 1994.

[9] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, *Robust Automatic Speech Recognition, a Bridge to Practical Applications*. New York: Academic, 2015.

[10] G. Saon, H. K. J. Kuo, S. Rennie, and M. Picheny, "The IBM 2015 English conversational telephone speech recognition system," in *Proc. Interspeech*, Dresden, Germany, 2015.

[11] G. Tür and R. De Mori, Eds., *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Hoboken, NJ: Wiley, 2011.

[12] A. Deoras, R. Sarikaya, G. Tür, and D. Hakkani-Tür, "Joint decoding for speech recognition and semantic tagging," in *Proc. Interspeech*, 2012, pp. 1067–1070.

[13] R. Sarikaya, G. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE Trans. Audio, Speech, Language Process.*, vol. 22, no. 4, pp. 778–784, 2014.

[14] N. Gupta, G. Tür, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Gilbert, "The AT&T spoken language understanding system," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 1, pp. 213–222, Jan. 2006.

[15] J. P. Robichaud, P. Crook, P. Xu, O. Z. Khan, and R. Sarikaya, "Hypotheses ranking for robust domain classification and tracking in dialogue systems," in *Proc. Interpseech*, Singapore, 2014, pp. 145–149.

[16] P. A. Crook, A. Marin, V. Agarwal, K. Aggarwal, T. Anastasakos, R. Bikkula, D. Boies, A. Celikyilmaz, S. Chandramohan, Z. Feizollahi, R. Holenstein, M. Jeong, O. Z. Khan, Y. B. Kim, E. Krawczyk, X. Liu, D. Panic, V. Radostev, N. Ramesh, J. P. Robichaud, A. Rochette, L. Stromberg, and R. Sarikaya, "Task completion platform: A self-serve multi-domain goal oriented dialogue platform," in *Proc. North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, CA, 2016, pp. 47–51.

[17] J. R. Bellegarda, "Spoken language understanding for natural interaction: The Siri experience," in *Natural Interaction with Robots, Knowbots and Smartphones*. New York: Springer-Verlag, 2014, pp. 3–14.

[18] O. Z. Khan, J. P. Robichaud, P. Crook, and R. Sarikaya, "Hypotheses ranking and state tracking for a multi-domain dialog system using ASR results," in *Proc. Interspeech*, Dresden Germany, 2015.

[19] A. J. Hunt and A. W. Black. "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 1996, vol. 1, pp. 373–376.

[20] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups" *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[21] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, Singapore, 2014, pp. 338–342.

[22] A. Bhargava, A. Celikyilmaz, D. H. Tur, and R. Sarikaya, "Easy contextual intent prediction and slot detection," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, May 2013, pp. 8337–8341.

[23] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 2014, pp. 136–140.

[24] C. Liu, P. Xu, and R. Sarikaya, "Deep contextual language understanding in spoken dialogue systems," in *Proc. Interspeech*, 2015, pp. 120–124.

[25] R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J. P. Robichaud, A. Celikyilmaz, Y. B. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T.

Anastasakos, Z. Feizollahi, N. Ramesh, H. Suzuki, R. Holenstein, E. Krawczyk, and V. Radostev, “An overview of end-to-end language understanding and dialog management for personal digital assistants,” in *Proc. IEEE Spoken Language Technology,* San Diego, CA, 2016.

[26] T. Winograd, “Understanding natural language,” *Cognitive Psych.*, vol. 3, no. 1, pp. 1–191, 1972.

[27] W. A. Woods, “Transition network grammars for natural language analysis,” *Commun. ACM*, vol. 13, no. 10, pp. 591–606, 1970.

[28] W. Ward and S. Issar, “Recent improvements in the CMU spoken language understanding system,” in *Proc. Workshop Human Language Technology, Association for Computational Linguistics*, 1994, pp. 213–216.

[29] R. F. Simmons, “Natural language question-answering systems: 1969,” *Commun. ACM*, vol. 13, no. 1, pp. 15–30, 1970.

[30] T. Strzalkowski and S. Harabagiu, Eds., *Advances in Open-Domain Question-Answering*. Berlin, Germany: Springer-Verlag, 2006.

[31] C. L. Isbell and J. S. Pierce, “An IP continuum for adaptive interface design,” in *Proc. Human–Computer Interaction Int. 2005*, Las Vegas, NV.

[32] E. Levin, R. Pieraccini, and W. Eckert, “A stochastic model of human-machine interaction for learning dialog strategies,” *IEEE Speech Audio Process.*, vol. 8, no. 1, pp. 11–23, 2000.

[33] Y. Shi, Y. C. Pan, M. Y. Hwang, K. Yao, H. Chen, Y. Zou, and B. Peng, “A factorization network based method for multi-lingual domain classification,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 2015, pp. 5276–5280.

[34] G. E. Hinton, S. Osindero, and T. Yee-Whye, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[35] I. H. Witten and F. Eibe, *Data Mining: Practical Machine Learning Tools and Techniques*. San Mateo, CA: Morgan Kaufmann, 2005.

[36] X. Liu and R. Sarikaya, “A model based approach to weight dictionary entities for spoken language understanding,” in *Proc. IEEE Spoken Language Technology,* Lake Tahoe, NV, 2014.

[37] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, et al., “Building Watson: An overview of the DeepQA project,” *AI Mag.,* vol. 31, no. 3, pp. 59–79, 2010.

[38] D. Boies, R. Sarikaya, A. Rochette, Z. Feizollahi, and N. Ramesh, “Techniques for updating partial dialog state,” U.S. Patent Application 20150095033, 2013.

[39] A. Celikyilmaz, Z. Feizollahi, D. Hakkani-Tür, and R. Sarikaya, “Resolving referring expressions in conversational dialogs for natural user interfaces,” in *Proc. Empirical Methods on Natural Language Processing,* 2014, pp. 2094–2104.

[40] N. Shadbolt, W. Hall, and T. Berners-Lee, “The semantic web revisited,” *IEEE Intell. Syst.*, vol. 21, no. 3, pp. 96–101, 2006.

[41] R. Baeza-Yates and A. Tiberi, “Extracting semantic relations from query logs,” in *Proc. ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2007, pp. 76–85.

[42] L. Heck, D. Hakkani-Tür, and G. Tür, “Leveraging knowledge graphs for web-scale unsupervised semantic parsing,” in *Proc. Interspeech*, August 2013, pp. 1594–1598.

[43] A. El-Kahky, X. Liu, R. Sarikaya, G. Tür, D. Hakkani-Tür, and L. Heck, “Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014, pp. 4067–4071.

[44] Y. Ma, P. Crook, and R. Sarikaya, “Statistical inference over knowledge graphs for task-oriented spoken dialog systems,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, Brisbane, Australia, 2015, pp. 5346–5350.

[45] Y.-B. Kim, M. Jeong, and R. Sarikaya, “Semi-supervised slot tagging with partially labeled sequences from web search click logs,” in *Proc. North American Chapter of the Association for Computational Linguistics,* 2015, pp. 84–92.

[46] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” in *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, New York, 2008, pp. 1247–1250.

[47] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, “GALAXY-II: A reference architecture for conversational system development,” in *Proc. Int. Conf. Spoken Language Processing,* 1998, vol. 98, pp. 931–934.

[48] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck, “Learning deep structured semantic models for web search using clickthrough data,” in *Proc. 22nd ACM Int. Conf. Information and Knowledge Management*, 2013, pp. 2333–2338.

[49] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T. J. Hazen, and L. Hetherington, “Jupiter: A telephone-based conversational interface for weather information,” *IEEE Speech Audio Process.*, vol. 8, no. 1, pp. 85–96, 2000.

[50] R. Pieraccini and J. Huerta, “Where do we go from here? Research and commercial spoken dialog systems,” in *Proc. 6th SIGdial Workshop on Discourse and Dialogue*, 2005.

[51] Y. B. Kim, K. Stratos, R. Sarikaya, and M. Jeong, “New transfer learning techniques for disparate label sets,” in *Proc. Association for Computational Linguistics,* 2015, pp. 473–482.

[52] J. Huang and E. N. Efthimiadis, “Analyzing and evaluating query reformulation strategies in web search logs,” in *Proc. 18th ACM Conf. Information and Knowledge Management*, 2009, pp. 77–86.

[53] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The hidden information state model: A practical framework for POMDP-based spoken dialogue management,” *Comput. Speech Lang.*, vol. 24, no. 2, pp. 150–174, 2009.

[54] A. Marin, P. Crook, O. Z. Khan, V. Radostev, K. Aggarwal, and R. Sarikaya, “Flexible, rapid authoring of goal-orientated, multi-turn dialogues using the task completion platform,” in *Proc. Interspeech 2016*, San Francisco, CA, 2016, pp. 1571–1572.

[55] C. J. C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu, “Learning to rank using an ensemble of lambda-gradient models,” *J. Mach. Learn. Res.*, vol. 14, pp. 25–35, 2011.

[56] W. T. Yih, X. He, and C. Meek. “Semantic parsing for single-relation question answering,” in *Proc. Association for Computational Linguistics,* 2014, pp. 643–648.

[57] M. Nickel, M, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proc. IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[58] M. Dinarelli, A. Moschitti, and G. Riccardi, “Discriminative reranking for spoken language understanding,” *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 2, pp. 526–539, Feb. 2012.

[59] V. Rieser and O. Lemon, “Natural language generation as planning under uncertainty for spoken dialogue systems,” in *Empirical Methods in Natural Language Generation*. Berlin, Germany: Springer-Verlag, 2010, pp. 105–120.

[60] R. J. Kate, Y. W. Wong, and R. J. Mooney, “Learning to transform natural to formal languages,” in *Proc. 12th Nat. Conf. Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, 2005, pp. 1062–1068.

[61] A. H. Oh and A. I. Rudnicky, “Stochastic language generation for spoken dialogue systems,” in *Proc. 2000 ANLP/NAACL Workshop on Conversational Systems*, 2000, vol. 3, pp. 27–32.

[62] J. Jiang, A. H. Awadallah, R. Jones, U. Ozertem, I. Zitouni, R. G. Kulkarni, and O. Z. Khan, “Automatic online evaluation of intelligent assistants,” in *Proc. 24th Int. World Wide Web Conf.*, 2015, pp. 506–516.

[63] M. Czerwinski, E. Horvitz, and S. Wilhite, “A diary study of task switching and interruptions,” in *Proc. ACM Conf. Human Factors in Computing Systems,* Vienna, Austria, 2004, pp. 175–182.

[64] V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut, “What a to-do: Studies of task management towards the design of a personal task list manager,” in *Proc. ACM Conf. Human Factors in Computing Systems,* Vienna, Austria, 2004, pp. 735–742.

[65] Statista. (2016). Number of mobile phone users worldwide from 2013 to 2019 (in billions). [Online]. Available: http://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/

[66] Siri. *Apple*. [Online]. Available: http://www.apple.com/ios/siri/

[67] Google Now. *Google*. [Online]. Available: https://www.google.com/landing/now/

[68] Cortana. *Microsoft*. [Online]. Available: https://www.microsoft.com/en-us/mobile/experiences/cortana/

[69] Alexa. *Amazon Developer*. [Online]. Available: https://developer.amazon.com/public/solutions/alexa

[70] M. Meeker. (2016, June 1). Internet trends 2016—Code conference. Kleiner Perkins Caufield Byers. Menlo Park, CA. [Online]. Available: http://www.kpcb.com/blog/2016-internet-trends-report

[71] Samsung. (2014, Apr. 23). 10 sensors of Galaxy S5: Heart rate, finger scanner and more. *Samsung Newsroom*. [Online]. Available: http://news.samsung.com/global/10-sensors-of-galaxy-s5-heart-rate-finger-scanner-and-more