O'REILLY®

A Developer's Guide to Building AI Applications

Create Your First Intelligent Bot with Microsoft AI

Anand Raman
& Wee Hyong Tok

Microsoft

MICROSOFT AI PLATFORM

Build Intelligent Apps

Artificial Intelligence productivity for every developer and every scenario

Learn more about Microsoft AI

azure.com/ai

Learn how to build intelligence into your solutions using the Microsoft AI platform

aischool.microsoft.com

A Developer's Guide to Building AI Applications

Create Your First Intelligent Bot with Microsoft AI

Anand Raman and Wee Hyong Tok

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

A Developer's Guide to Building AI Applications
by Anand Raman and Wee Hyong Tok

Copyright © 2018 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Nicole Tache
Production Editor: Nicholas Adams
Copyeditor: Octal Publishing, Inc.
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

May 2018: First Edition

Revision History for the First Edition
2018-05-24: First Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *A Developer's Guide to Building AI Applications*, the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O'Reilly and Microsoft. See our *statement of editorial independence*.

978-1-492-03784-2

[LSI]

Table of Contents

A Developer's Guide to Building AI Applications 1
Introduction 1
The Intersection of Cloud, Data, and AI 4
The Microsoft AI Platform 9
Developing an Intelligent Chatbot 11
Adding "Plug and Play" Intelligence to Your Bot 33
Building an Enterprise App to Gain Bot Insights: The Conference Buddy Dashboard 36
Paving the Road Ahead 43

A Developer's Guide to Building AI Applications

Introduction

Artificial Intelligence is rapidly becoming a mainstream technology that is helping transform and empower us in unexpected ways. Let us take a trip to remote Nepal to see a fascinating example. Like the vast majority of Nepalese, Melisha Ghimere came from a remote village from a family of subsistence farmers who raised cows, goats, and water buffalos. Seven years ago, she watched her relatively wealthy uncle and aunt lose a lot of their herd to an outbreak of anthrax; they were never to recover their economic footing. Melisha went on to college thinking about the plight of her family. In college, she worked to develop a predictive early warning solution to help farmers. With a team of four students, they researched livestock farming, veterinary practices, and spoke to farmers. They built a prototype for a monitoring device that tracks temperature, sleep patterns, stress levels, motion, and the activity of farm animals. Melisha's AI system predicts the likely health of each animal based on often subtle changes in these observations. Farmers are able to track their animals, receive alerts, and actionable recommendations. Although her project is still in its infancy, the field tests have shown the solution was about 95% accurate in predicting risks to an animal's health. Melisha and her team were able to help a family prevent a deadly outbreak of an anthrax infection by identifying a diseased cow, before symptoms were evident to the farmer. Melisha's team was a regional finalist in Microsoft's Imagine Cup competition in 2016.[1]

Let me give you another example much closer to home, the power of AI in transforming the retail experience. Lowes Innovation Labs has now created a unique prototype shopping experience for home remodeling. For example, a customer

1 The Future Computed—Artificial Intelligence and its role in society—Microsoft

can now walk in and share her dream kitchen photos with a design specialist. Using an AI-powered application, the design specialist gains deep insight into the customer's style and preference. The application generates a match from the Lowe's dream kitchen collection, and the design of the kitchen is then shown in very realistic holographic mixed-reality through a Hololens.[2] The customer can now visualize, explore, and change the design to his taste in the mixed reality environment in real time. Applications like these are vanguards of the revolution in retail experiences that AI will bring for consumers.

Healthcare is another field that is at the cusp of a revolution. With the power of AI and a variety of data sources from genomics, electronic medical records, medical literature, and population data, scientists are now able to predict health emergencies, diagnose better, and optimize care. A unique example in this area comes from Cochrane, a highly reputed nonprofit organization dedicated to gathering and summarizing the best evidence from research to help doctors make informed choices about treatment. Cochrane conducts systematic reviews, which digests and analyzes explosively growing medical literature, and reduces it into fairly short and manageable pieces of work to give doctors the best possible guidance on the effects of healthcare interventions. For example, a recent systematic review of medical studies looked at whether steroids can help with the maturation of premature babies' lungs. The review showed conclusively that steroids can help save the babies' lives. This intervention has helped hundreds of thousands of premature babies. However, such reviews are very labor intensive and can take two to three years to complete. Cochrane's Project Transform was born out of the need to make systematic reviews more efficient, give more timely and relevant guidance to doctors, and therefore help save more lives. Project Transform uses AI to manipulate and analyze the literature and data very efficiently and therefore allow researchers to understand the data and interpret the findings. It creates a perfect partnership between human and machine, where a significant amount of the heavy overhead of systematic reviews is reduced, and the human analysis skills can be directed where they are most needed for timeliness and quality.

There's no field that will be left untouched by the transformational power of AI. I can point you to fields as diverse as astronomy where AI has accelerated the pace of new discoveries, and the area of conservation where ecologists and conservationists are working with AI-powered tools to help track, study, and protect elusive and endangered animals.

2 *http://www.lowesinnovationlabs.com/hololens*

NOTE A lot of times we become bogged down in the discussions of the appropriate algorithm or tools, but the real power of AI resides in the ideas and questions that precede it.

It's the conservationist pondering on how to create sustainable habitats, the doctor wondering how to better serve their patient, the astronomer's and citizen scientist's curiosity that expands our collective consciousness to the outer limits of the universe. AI has the potential to empower the noblest of human causes, and we are just at the beginning. The field is still nascent, and yet these breakthroughs highlight the explosive power of AI in reshaping our daily experiences, how we do business, and how we live our lives.

Five decades ago, the early inventors in AI could only dream of what most consumers take for granted today. From voice-powered assistants like Cortana, Siri, or Alexa, to smartphones and self-driving cars, we seem to be living in "sci-fi" pages. What do the next two decades hold for us? Five decades? At Microsoft, we have made it our mission to advance AI innovations by democratizing AI tools in the same way that we democratized the power of computing in the mainframe era by envisioning a personal computer in every home, school, and workplace.

As educator and computing pioneer Alan Kay said, "The best way to predict the future is to create it." In the same spirit, we are writing this book to give developers a start on creating the future with AI. In this book, we will show you how to create your first AI application in the cloud, and in the process learn about the wealth of AI resources and capabilities that are now rapidly becoming available to programmers. The application we create will be an AI-infused Bot, a "Conference Buddy," that helps create a novel Question and Answer experience for the attendees and speakers participating in a conference. As we build this Bot, you will get a glimpse into how AI can help understand conversations, perceive vast amounts of information, and respond intelligently. In the process, you will also get a glimpse into the landscape of AI tools and emerging developments in the field.

We selected a chatbot as our example because it is a relatively easy entry point into AI, and in the process we highlight resources and links to help you dig deeper. Chatbots are also ubiquitous, with interesting implementations, and transforming the way in which we interact with computers. We also give you a wider lens on the landscape of AI tools and a glimpse into exciting new developments in the field.

Here's a roadmap to the contents of this book:

"The Intersection of Cloud, Data, and AI"
: In the rest of this section, we will introduce AI and the powerful intersection of data, cloud, and AI tools that is creating a paradigm shift, helping enable systems of intelligence.

"The Microsoft AI Platform"
: Here, we explore the Microsoft AI platform and point out the tools, infrastructure, and services that are available for developing AI applications.

"Developing an Intelligent Chatbot"
: This section presents a discussion of chatbots, conversational AI, and highlights some chatbot implementations. How do you create an intelligent chatbot for the enterprise? We provide a high-level architecture using the Conference Buddy bot example, including code samples; discuss design considerations and technologies involved; and take a deep dive into the abstraction layer of the bot, which we call the Bot Brain.

"Adding "Plug and Play" Intelligence to Your Bot"
: This section explores how you easily give the bot new skills and capabilities such as vision, translation, speech, and other custom AI abilities as well as how you develop the Bot Brain's intelligence.

"Building an Enterprise App to Gain Bot Insights: The Conference Buddy Dashboard"
: This section highlights the Conference Buddy dashboard, which allows the conference speaker and attendees to see the attendees' questions and answer them in real-time. We also discuss how to instrument the Bot to get metrics and application insights.

"Paving the Road Ahead"
: In the final section, we consider an exciting development in the AI world with the release of Open Neural Network Exchange (ONNX) and also Microsoft's commitment to the six ethical principles—fairness, reliability and safety, privacy and security, inclusivity, transparency, and accountability—to guide the cross-disciplinary development and use of AI.

The Intersection of Cloud, Data, and AI

We define AI as a set of technologies that enable computers to assist and solve problems in ways that are similar to humans by perceiving, learning, and reasoning. We are enabling computers to learn from vast amounts of data, and interact more naturally and responsively with the world, rather than following pre-

programmed routines.[3] Technologies are being developed to teach computers to "see," "hear," "understand," and "reason."[4] The key groups of capabilities include:

Computer vision
: This is the ability of computers to "see" by recognizing objects and their relationships in a picture or video.

Speech recognition and synthesis
: This is the ability of computers to "listen" by understanding the words that people say and to transcribe them into text, and also to read text aloud in a natural voice.

Language understanding
: The ability of computers to "comprehend" the meaning of words and respond, considering the many nuances and complexities of language (such as slang and idiomatic expressions). When computers can effectively participate in a dialog with humans, we call it "conversational AI."

Knowledge
: The ability of a computer to "reason" by representing and understanding the relationship between people, things, places, and events.

What do these capabilities mean in the context of enterprise applications? The power of AI is powering applications that *reason* by unlocking the power of all data collected over time, across repositories and massive datasets through machine learning. These AI-powered systems *understand* and create meaning in unstructured data such as email, chats, and handwritten notes, all of which we previously could not process. And, more important, the systems are *interacting* with customers and engaging them in different channels and in ways that are hyperpersonalized.

In the same vein, businesses are using AI-powered applications to digitally transform every aspect of their organizations including: transforming their products through insights from customer data, optimizing business operations by predicting anomalies and improving efficiencies, empowering their employees through intelligent tools, and engaging their customers through conversational agents that deliver more customized experiences.

The following are examples of the questions that power the engines running AI applications:

Classifications
: Which category does it belong to?

---

3 *Time* magazine: Why you shouldn't be afraid of AI

4 The Future Computed—Artificial Intelligence and its role in society—Microsoft

Regression
: How much? How many?

Anomaly
: Is it weird?

Clustering
: How is it organized?

So how do you begin to design AI-powered solutions that take advantage of all the aforementioned capabilities? We design AI solutions to complement and unlock human potential and creative pursuits. There are significant implications of what it means to design technology for humans, and this includes considering ethical implications; understanding the context of how people work, play, and live; and creating tailored solutions that adapt over time.

One of the most fascinating areas of research is bridging emotional and cognitive intelligence to create *conversational AI* systems that model human language and have insight into the logical and unpredictable ways human interact.

According to Lili Cheng, corporate vice president of Microsoft AI and Research, "This likely means AI needs to recognize when people are more effective on their own—when to get out of the way, when not to help, when not to record, when not to interrupt or distract."[5]

The time for AI is now, given the proliferation of data, the limitless availability of computing powers on the cloud, and the rise of powerful algorithms that are powering the future.

Modern AI: Intersection of Data, Cloud Computing, and AI

Although AI research has been ongoing for decades, the past few years have seen a leap in practical innovations, catalyzed by vast amounts of digital data, online services, and enormous computing power. As a result, technologies such as natural-language understanding, sentiment analysis, speech recognition, image understanding, and machine learning have become accurate enough to power applications across a broad range of industries.

Let's examine the three important developments that are helping create modern AI: data and the digital transformation, cloud computing, and AI algorithms and tools.

5 *Time* magazine: Why you shouldn't be afraid of AI

Data and the digital transformation

At the center of AI is data, and the increasing digitization of our age is resulting in the proliferation of what is known as *big data*. Out of approximately 7.4 billion people on Earth, more than 4 billion own mobile devices and 3.8 billion are connected to the internet, and these numbers are projected to keep growing. The vast majority of new information in the world is now generated and consumed online, and an increasingly large fraction of the economy is migrating to online services, from shopping to banking, entertainment, media, and communications. As our lives have become increasingly digitized and sensors (microphones, cameras, location, and other sensors) have become cheap and ubiquitous, more data than ever before is available from which computers can learn and reason. At the same time, as we engage in online interactions and transactions digitally, new response and feedback data is generated that allows AI algorithms to adapt and optimize interactions.

The staggering amount and growth rate of data has led to significant innovation in how we efficiently store, manage, and mine the data for flexible, real-time analysis. Most such data flows to public or private clouds over the internet. "Big Data" systems help to handle the heterogeneous nature of such data, and support many analysis methods, such as statistical analysis, machine learning, data mining, and deep learning.

Such systems are at the heart of what makes it possible for computers to "see," "hear," and "reason," and to discern patterns often undetectable to human eyes.

Cloud computing

The internet, and the digital transformation of the world in turn, helped catalyze cloud computing. Processing the data and delivering large-scale online services requires massive computing power, reliable networks, storage, and data processing. The cloud provides a powerful foundation and platform to handle these challenges. It allows you to stream data from connected devices, offers massive data storage capacity and elastic, scalable computing power to integrate, analyze, and learn from the data.

You can also get the largest servers, latest GPUs, and latest cutting-edge hardware like Field Programmable Gate Arrays (FGPAs) to accelerate demanding computations without the exorbitant overhead cost of building and provisioning data centers and server farms. Enormous connectivity allows every type of device—what we know as the Internet of Things (IoT)—to bring massive amounts of data into the cloud on a real-time basis for analysis and AI at scale. Furthermore, the cloud provides the necessary infrastructure and tools to offer enterprise-grade security, availability, compliance, and manageability for the applications and services deployed on the cloud.

AI algorithms and tools

The explosion of use cases for AI driven by online services and the digital transformation in turn catalyzed enormous progress in AI algorithms. One of the most profound innovations in recent years has been deep learning. This technique, inspired by neural networks in the brain, allows computers to learn deep concepts, relationships, and representations from vast amounts of data (such as images, video, and text), and perform tasks such as object and speech recognition with accuracy comparable to humans. Today, open source tools such as the Cognitive Toolkit, PyTorch, and Tensorflow make deep learning innovations accessible to a wide audience. And all the major cloud vendors now have services that substantially simplify AI development to empower software engineers.

Modern AI lives at the intersection of these three powerful trends: digital data from which AI systems learn, cloud-hosted services that enable AI-powered interactions, and continuing innovations in algorithms that make the AI capabilities more powerful, while enabling novel applications and use cases.

Systems of Intelligence: A Paradigm Shift

In an insightful article, Jerry Chen, from Greylock Partners, explores the idea of *systems of intelligence*, which are powered by AI and are meant to recombine multiple datasets, business processes, and workflows to create a new context. An example is an application that combines web analytics with customer data and social data to predict end-user behavior, churn, or serve more timely content. The stories from the beginning of this chapter are examples of how such systems were created to generate new insights and value across various industries (farming, retail, healthcare, etc.)

The rise of *systems of intelligence* is one example of how the combination of data, cloud computing, and AI is ushering us into a world of pervasive online systems of intelligence. Whether it be ride-sharing services, online retail, social networks, media and entertainment, banking, investments, transportation, manufacturing, healthcare, and government, such systems of intelligence will mediate, manage, and optimize all interactions and exchanges. The extent of this paradigm shift is beyond the comprehension of most of us. Recall the prescient words of the famous historian of science, Thomas Kuhn:

> Led by a new paradigm, scientists adopt new instruments and look in new places. Even more important, during revolutions scientists see new and different things when looking with familiar instruments in places they have looked before. It is rather as if the professional community had been suddenly transported to another planet where familiar objects are seen in a different light and are joined by unfamiliar ones as well. (Kuhn, 1962)[6]

6 In *The Structure of Scientific Revolutions* (1962, 2nd ed. 1970)

In very much the same way, engineers, entrepreneurs, and business leaders, empowered with systems of intelligence, are seeing familiar problems in a different light and discovering entirely new economic opportunities.

It is also useful to look at the transformation of enterprise information technology over the last few decades:

Client-Server Revolution → Systems of records
: It was the client-server revolution that first enabled broad use of information technology to manage business. Organizations first built systems of records: Customer Relationship Management (CRM) systems; Human Capital Management (HCM) systems for HR; and Enterprise Resource Planning (ERP) systems for financials and key assets.

Internet Revolution → System of engagement
: The rise of the internet, mobile, and chat allowed us to create systems of engagement that interfaced between the systems of records and interacted directly with customers and suppliers.

AI Revolution → System of intelligence
: What is emerging now are systems of intelligence that integrate data across all systems of record, connect directly to systems of engagement, and build systems that understand and reason with the data. These systems can drive workflows and management processes, optimize operations, and drive intelligent interactions with customers, employees, suppliers, and stakeholders.

The IT industry has moved on from figuring out the basic questions underpinning big data and AI infrastructure, and it is now ushering the rise of machine learning and AI platforms, such as the Microsoft AI platform, as an application layer. This movement will inspire new business models based on big data/AI to emerge, in the same way that "web-enabled" businesses arose, and it will transform the consumer and business experience in ways that we can only get a glimpse of now. In the next section, we explore the Microsoft AI platform and highlight the tools and resources available for AI developers.

The Microsoft AI Platform

The Microsoft AI Platform aims to bring AI to every developer, and to empower developers to innovate and accelerate with a variety of services, infrastructure, and tools. From pre-built AI (that requires little to no training) to custom AI, the open Microsoft AI Platform enables developers to use various deep learning and machine learning frameworks and tools.

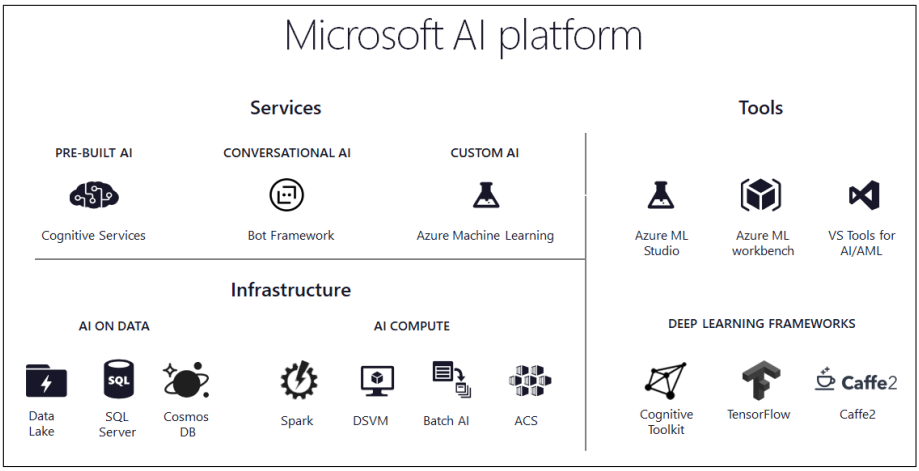

Figure 1-1. The Microsoft AI platform (figure courtesy of Microsoft)

The platform consists of the following services (illustrated in Figure 1-1):

Custom AI

Azure Machine Learning enables you to develop machine learning and deep learning models, train them in the cloud, and operationalize them. A variety of data and compute services are available in Azure to store and process your data.

Azure Machine Learning also provides an experimentation service, which allows you to start rapid prototyping on the desktop, and scale this to run on deep learning virtual machines, Spark clusters, and Azure Batch AI services. Additionally, Azure Machine Learning allows you to manage model performance and life cycle, and to collaborate and share solutions using Git. Docker containers enable you to deploy models into production faster in the cloud, on-premises, or to intelligent edge devices.

Pre-Built AI

You can consume high-level "finished" services that accelerate development of intelligent applications, with just a few lines of code. These services are customized to an organization's availability, security, and compliance requirements. Cognitive Services provides pre-built AI, via a set of APIs, SDKs, and services. This enables developers to infuse AI into applications, websites, bots, and more.

Bot Framework

Provides tools to accelerate development for conversational AI. Integrates seamlessly with Cortana, Office 365, Slack, Facebook Messenger, and more. The Bot Framework is a platform for building, connecting, testing, and deploying powerful and intelligent bots. With support for .NET, Node.js, and

REST, you can get the Bot Builder SDK and quickly start building bots with the Bot Framework. In addition, you can take advantage of Microsoft Cognitive Services to add smart features like natural language understanding, image recognition, speech, and more.

Among the coding and management tools in the Microsoft AI Platform is Visual Studio Tools for AI, which enable you to build, debug, test, and deploy AI with Visual Studio Code on Windows and Mac. In addition, you can leverage various tools such as Jupyter Notebooks, PyCharm, and more.

The Microsoft AI Platform also integrates with various machine learning and deep learning frameworks, including TensorFlow, Caffe2, Microsoft Cognitive toolkit (CNTK), Scikit-learn, MXNet, Keras, and Chainer.

To help you get started, you can leverage the resources that are available in the Azure AI Gallery, which provides end-to-end solution templates, reference architectural diagrams, and sample code.

In the next section, we explore how you can develop your next intelligent application using the Microsoft AI Platform. As an example, we walk through developing an intelligent chatbot and include discussions on conversational AI/chatbots, some interesting user stories, design considerations, and how to develop the chatbot's intelligence.

Developing an Intelligent Chatbot

The recent explosion and popularity of chatbots underscores our essential nature as social beings. Instead of filling out forms, clicking through screens, and finding our way around difficult-to-navigate websites, brands have been making a variety of services available through the convenience of a dialog interface using speech or text. Think of bots as applications that you can talk to. Chatbots interact with you with the ease of a conversation and help answer your questions or carry out tasks like securing your reservation, ordering food, or purchasing an item. When designed correctly, they can even fool you into thinking you are chatting with a human.

Some examples of chatbots in retail are the UPS bot, launched by the parcel delivery service giant, which allows customers to get the information they need about their packages, shipment, rates, and UPS locations. Macy's bot connects customers to information about orders, merchandise, common queries, and even takes actions like applying coupons and discounting items in shopping bags. Dixon's Carphone, a major European electronics and telecommunication retailer and service provider, wanted to bridge the gap between its customers' online research with its in-store experience. The company introduced a chatbot called Cami to help streamline the customer's experience by giving store employees access to what the customers were looking for online and directing them to the products

or recommend new ones within the store. In the retail industry alone, we are seeing chatbots augmenting existing staff capabilities, reducing cost and time associated with support, and transforming the overall customer experience.[7]

There are novel applications like those by Japanese navigation technology company, NAVTIME Japan, which introduced a personal assistant chatbot aimed at helping tourists plan their trips in real time as they travel around Japan. The chatbot answers questions like, "Where can I get some dinner?" or, "Where can I buy a souvenir?" It then provides geolocation services to let the user know whether they are close to a place that they wanted to visit. Visiting a country where you don't speak the language no longer needs to be an overwhelming experience.[8]

Companies are using "conversation-as-a-platform"—that is, bots that understand human language or use language as the primary interface—to help with internal operations, as well. From HR virtual assistants that field routine employee questions and help with recruiting efforts in answering questions and routing resumes, to productivity bots like meeting and digital assistants. Manufacturers are using bots to connect IoT to support staff, so that the staff can keep an eye on anomalies and receive alerts for predictive maintenance.

After more than a decade of researching the nuances of human language and technologies to facilitate AI–human interaction, Microsoft released Xiaoice, a Chinese celebrity chatbot with millions of followers. Part of her popularity stems from the way she exhibits high *emotional quotient* (EQ) by remembering parts of a conversation and following up in later conversations. Young Chinese men and women turn to Xiaoice to talk about their issues, heartbreaks, and daily stresses; some were even quoted to say "I love you" to her.[9] Xiaoice's popularity and talent knows no bounds: she has published a book of poetry, helped Chinese people write more than 200 million poems, and is currently hosting a TV morning news program that has more than 800 million viewers.

Even though digital assistants like Cortana, Siri, and Alexa have long exhibited a high *IQ* in their task-based functions, Xiaoice illustrates the potential of combining the smarts of IQ with the empathy of EQ. Xiaoice, has been joined by Rinna in Japan, who is equally popular and now powers Nintendo's Pokemon games, and Zo in the United States, who has engaged users in rap battles.

There is still a lot of work, research, and opportunities to uncover in the area of raising the EQ of computer systems for translating the nuance of human language with its subtle changes in tone and meaning at any given context.

---

7 Building Bots-Chatbots in the Retail Industry

8 Navigation technology company uses chatbots to help tourists get around Japan

9 For Sympathetic ear, more Chinese turn to a smartphone program

Microsoft's ambitious vision extends beyond these chatbots to creating Conversational AI as a Platform, which puts natural language at the heart of computing. These systems will be imbued with AI-driven senses to create a seamless relationship between man and machine.

Evolution of Natural-Language Processing and Bots

Let's take a step back and discuss natural-language processing (NLP) and how, without the immense progress in the areas of NLP or Natural-Language Understanding (NLU) there would be very limited bot capabilities to speak of. Human language is often messy, imprecise, and vague with no explicit references to most parts of the speech. Until relatively recently, processing and analyzing natural language has been really challenging for computers, in spite of the sustained attempts of computer scientists since the 1950s to solve the problem of processing and analyzing textual components, sentiments, parts of speech and the various entities that make up speech. The recent advances in machine learning and the availability of vast amounts of digital text and conversational data through webchats, messaging systems, and services such as Twitter have helped us make immense progress in NLP.

NLP is essentially the ability to take a body of human-generated text and render it into machine readable language. NLP analyzes and extracts key metadata from text, including the following:

Entities
: NLP detects nouns, including people, places, and things.

Relations
: It identifies how the entities are related.

Concepts
: NLP extracts general concepts from the body of text that do not explicitly appear. For instance, the word "Excel" might return concepts like "Productivity Tools" and "Numbers," even if these terms do not appear in the text. This is a powerful tool for making connections that might not seem obvious at first glance.

Sentiment
: This scores the level of positivity or negativity in the text. This is useful, for example, to gauge sentiment related to a product or service. Or, in a customer support context, when to route a chat to a human upon detecting negativity.

Emotions
: This is sentiment analysis at a finer granularity; it classifies not just "positive" and "negative," but "anger," "sadness," and "joy."

Keywords

NLP extracts keywords and phrases to use as a basis for indexing, search, sorting, and so on.

Categories

This creates a hierarchical taxonomy for what the content is about and places it in a high-level category (text classification). This is useful for applications like recommending relevant content, generating ads, organizing emails, and so on.

Historically, you could implement NLP capabilities by either programming the rules directly, which made them difficult to adapt to new data or scale, or you could use machine learning models. But training machine learning models requires having access to rare expertise, large datasets, and complex tools, which limited their implementation to only large organizations that could afford it.

The availability of NLP capabilities like text classifiers and entity extractors as easy-to-use APIs in the cloud has powered the widespread use of chatbots. From the rise of open source tools to the arrival of cloud APIs, NLP capabilities once sequestered in academia and the research community are now accessible to a wider audience across industries.

An interesting example of NLP in the cloud is the Language Understanding (LUIS) service developed by Microsoft. LUIS uses machine learning to allow developers to build applications that can take user input in natural language and extract meaning and intent. A client application that converses with the user can pass user input to a LUIS app and extract the key concepts expressed by the user.

As with other Cognitive Services in the Microsoft AI platform, LUIS caters to the spectrum of developer expertise; you can use a prebuilt model, customize an existing one, or build your own from scratch. A model begins with a list of general user intents such as "book flight" or "contact help desk." After you identify the intent, you provide phrases called utterances (which is the substance of the dialog) for the intent. Then, you label the utterances with any specific details you want LUIS to pull out of the utterance. With prebuilt domains, you will have a set of entities and utterances for common categories like calendar, entertainment, communication, home automation, and more.

A system like LUIS is designed to extract the following key outputs:

Ability to recognize intents

What is the goal of the user? The intent is a purpose or goal expressed in a user's input, such as booking a flight, paying a bill, or finding a news article. You define and name intents that correspond to these actions. A travel app might define an intent named "BookFlight."

Ability to recognize utterances (dialog)

Utterances or dialog is text input from the user that your app needs to understand. It might be a sentence such as, "Book a ticket to Paris," or a fragment of a sentence like, "Booking," or, "Paris flight." Utterances and dialogs aren't always well formed, and there can be many dialog variations for an intent.

Ability to recognize entities

An entity represents detailed information that is relevant in the dialog. For example, in the dialog, "Book a ticket to Paris," "Paris" is a location entity. By recognizing and labeling the entities that are mentioned in the user's dialog, LUIS helps you choose the specific action to take to answer a user's request.

LUIS also allows developers to continuously improve the app through active learning. It also integrates with other AI tools in the cloud to power natural language in apps, bots, and IoT devices. Microsoft provides an additional set of tools through its Bot Framework, to incorporate LUIS and other Cognitive Services into the development of bots. The Microsoft Bot Framework provides an integrated development environment (IDE) to enable you to build, connect, test, deploy, and manage intelligent bots from one location.

Customers primed by their experiences with digital assistants and their widespread use of messaging apps, are engaging more and more with bots—they tend to make for a better user experience because they can typically respond faster and more effectively to user requests. For a lot of companies, bots are becoming a competitive differentiator. As we discussed earlier, many companies are strategically making chatbots available within the same messaging platforms their customers like to hang out in. Let us look at one bot use case—the Stack Overflow bot.

Your First Bot: The Scenario

Now, let's now look at how you can build your first bot. Imagine you are attending a technology conference presentation with hundreds of other enthusiastic attendees. As the speaker is presenting, you have a running list of questions. You want to ask your questions but:

- It is not Q&A time.
- You don't relish the idea of speaking up in public.
- You didn't raise your hand high enough or weren't picked during Q&A.
- You have a language barrier and cannot communicate fluently in the speaker's native language.

The reasons go on and on. Most people don't have an opportunity to fully engage with the speaker and content during conferences because of the logistics or other barriers.

What if you had a "Conference Buddy" chatbot that you could ask your questions as they occur to you and get answers as you go? And those questions also get routed on a dashboard where the speaker can engage and answer questions from the audience in real time.

The Conference Buddy chat client that we are going to build will have three functions:

1. Answer your greetings and introduce itself, as shown in Figure 1-2.

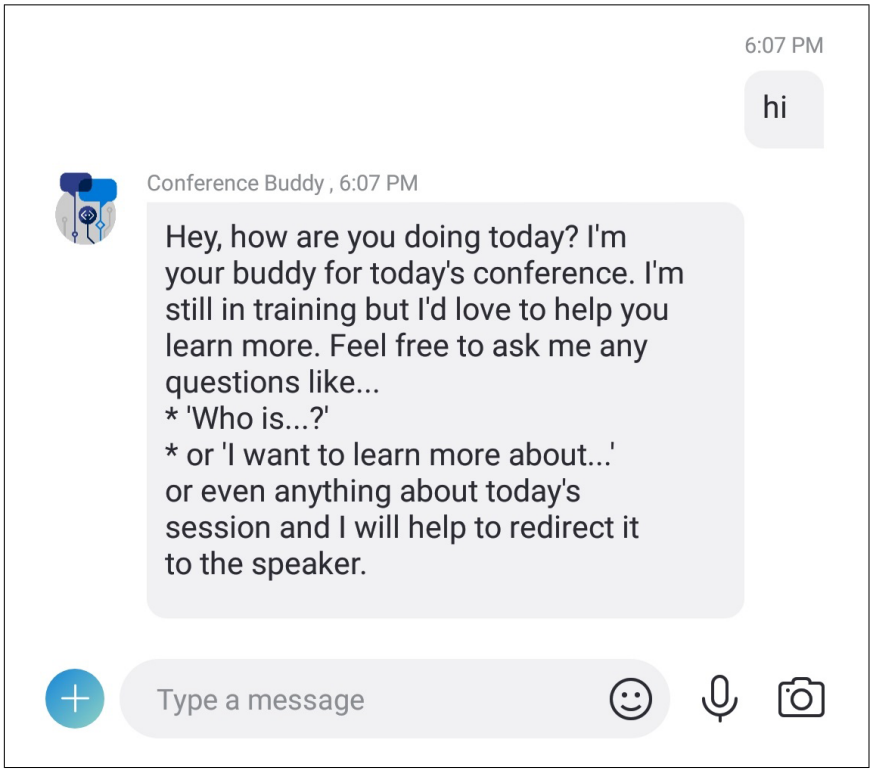

Figure 1-2. The Conference Buddy bot greeting

2. Answer some of your questions intelligently and automatically, when possible, as demonstrated in Figure 1-3.

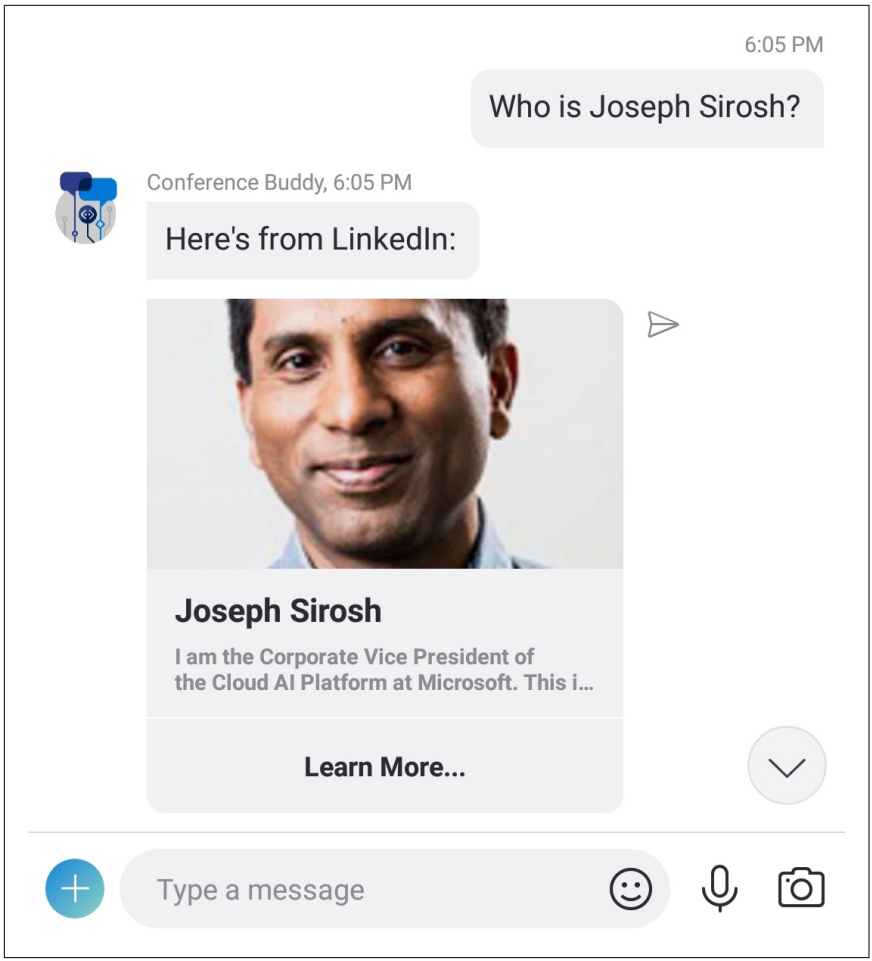

Figure 1-3. Conference Buddy Bot Question

3. Route your question for the speaker to a dashboard so the speaker can see all the questions from the audience, pick the question to answer, and engage, as illustrated in Figure 1-4.

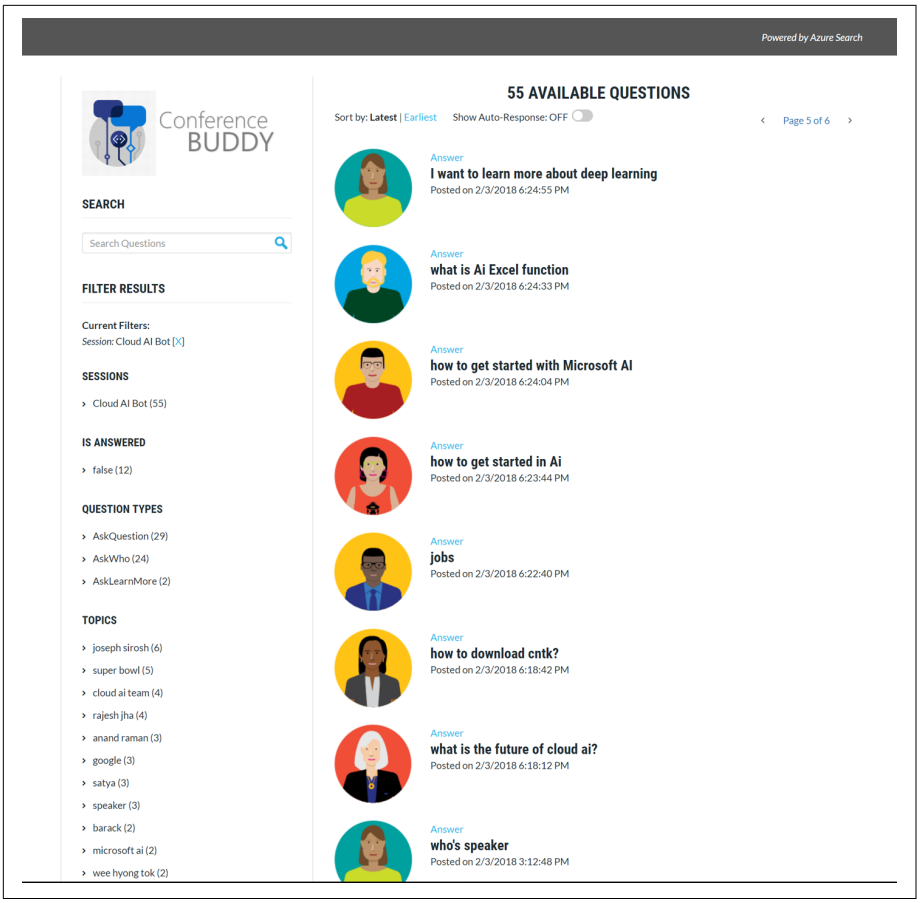

Figure 1-4. the Conference Buddy dashboard app

To get a feel for how this app looks and feels, I encourage you to visit the Github website *https://aka.ms/conferencebuddy* to see a demonstration and review the code for this sample.

An Overview of the Conference Buddy Bot Architecture

We will use a microservices architecture for building our bot (Figure 1-5) so that each component can be built, deployed, and scaled independently.

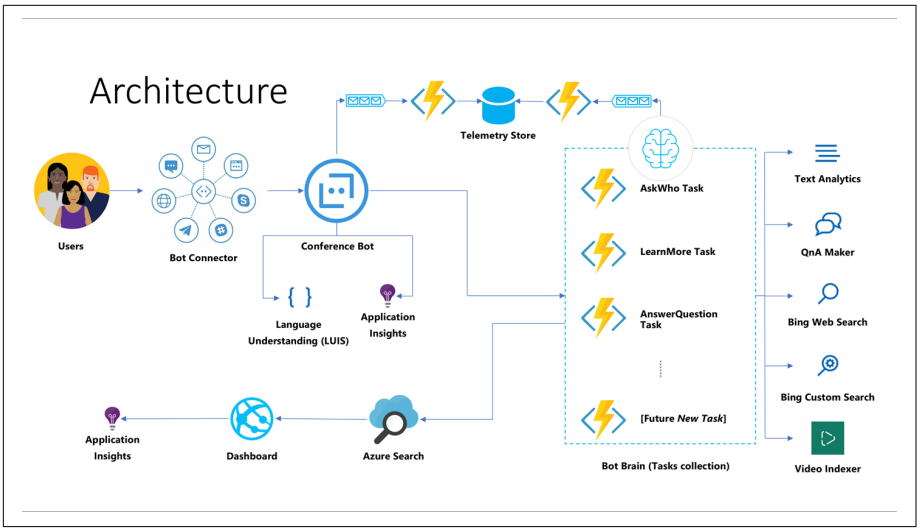

Figure 1-5. The Conference Buddy bot architecture

In our Conference Buddy bot, we have two major components:

Conference Bot
: This component intelligently handles all message related events.

Bot Brain
: This abstracts the business logic in the bot. Within the Bot Brain there are Individual Bot Tasks.

The questions and results are displayed on the Conference Buddy dashboard. Let's take a look at all of these components individually.

Conference Bot

The Conference Bot, built on the Bot Framework, intelligently handles all participant message events. The bot is omnichannel, which means users can email, Skype, or use a custom message service that will connect through the bot connector to reach the Conference Bot. Figure 1-6 shows the greeting when the Conference Buddy app is invoked.

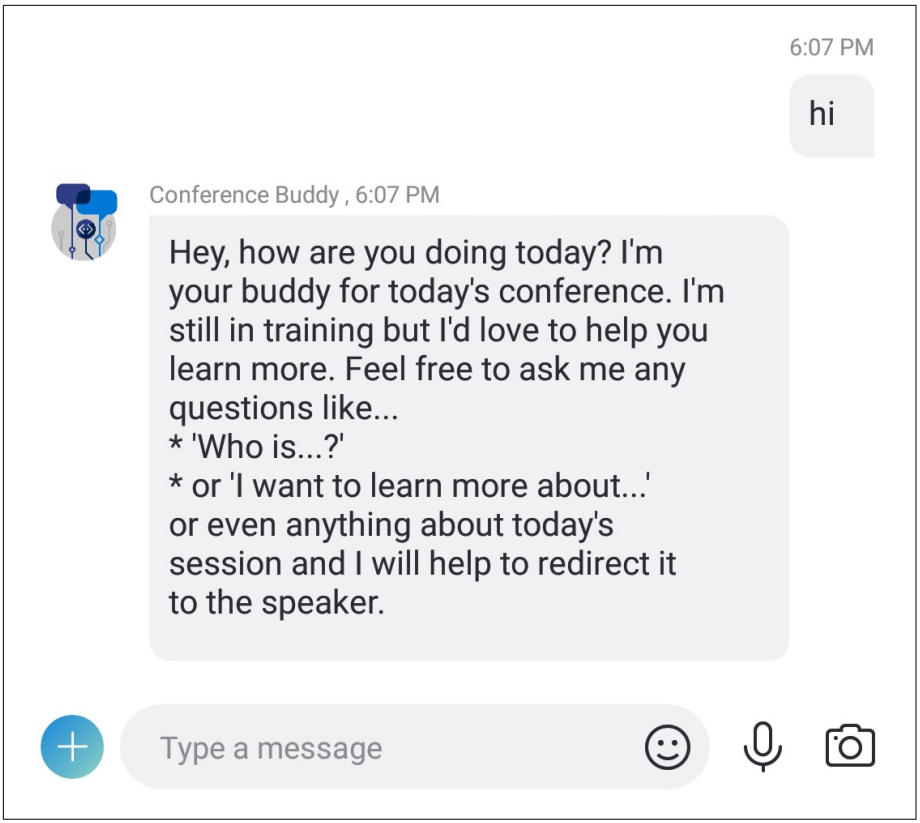

Figure 1-6. Conference Buddy bot greeting

The Conference Buddy does several things that are indicative of good design principles when it comes to the opening dialog:

- First the bot greets you with, "How are you?"
- Then, it introduces itself: "I am your buddy for today's conference. I'm still in training but would love to help you learn more."
- It gives you specific details on what it does: "Feel free to ask me any questions…"
- Finally, built in to the details is a suggested format on how you can phrase your questions: "Who is…" "I want to learn more about…" which will make it easier to process as well.

The Conference Buddy sends the message it receives from the user to LUIS to determine the intent of the message. Then, it selects the appropriate bot task in the Bot Brain to call via HTTP post to process the message. We dive into more

details when we discuss the conversation flow as an example in the next subsection.

Bot Brain

The Bot Brain is the logic or business intelligence that powers the Conference Buddy bot. In Figure 1-5, we summarized the current bot tasks within the Bot Brain:

- Ask Who task
- Learn More task
- Answer Questions task

The Bot Brain is a collection of intelligent bot tasks where each bot task is a function to complete a single task and is independent of other bot tasks. This is one of the ways we implemented microservices in our architecture. If one of the bot tasks returns an erroneous message, it is much easier to debug and fix the source of the issue without affecting the other bot tasks or components in the architecture.

Another example of microservices implementation is how we ensured that the Conference Bot that handles the message events is separate from the logic. In this way, each component is concerned only with carrying out its own functions; this makes it easier to optimize performance and scale, as well. An analogy of how microservices works is to imagine a car assembly factory where different parts are shipped from China, Japan, the United States, and so on. From the assembly workers' perspective, they don't care about the specific inner workings of each part, they just want to put each part in the right place, plug it in, and make sure the entire car works. When one part fails to work, it is usually easy to isolate the part and either replace or fix it without worrying about dismantling the entire car.

What enables the Conference Bot to talk to the Bot Brain is the Data Contract. The Data Contract is comprised of a Request Object and Response Object, and specifies the format and properties of what the request and response should include. The Data Contract is what enables the abstraction within our Conference Buddy bot. Without the Data Contract, the Conference Bot would not be able to invoke the Bot Brain or access the bot tasks to process the response.

In addition, the Bot Brain is reusable and can be used by other bots or applications. The Bot Brain can also evolve and grow in intelligence by adding new bot tasks.

We illustrate how to teach the Bot Brain new skills in the next section.

Bot task

The bot task is a function of the business logic that can take advantage of multiple Cognitive Services APIs to process the users' messages. The bot task is a function within the Bot Brain collection. Each bot task can be deployed to the same web service, or separate web services, and scaled independently from one another.

For example, the "Ask Who" task uses two Cognitive Services APIs—Bing Web Search and Bing Image Search—combining the results in a response object and sending it back to the Conference Bot. The Conference Bot then creates a rich graphical card to be returned the user. Because all of the bot tasks in our Conference Buddy bot invoke the Bing Web Search API, our chatbot will have immediate access to the world's online knowledge through an intelligent search that will provide relevant results. In addition, the "Learn More" bot task calls the Custom Bing Search, which allows you to use the same AI functionality that powers the Bing Search to a restricted number of websites. In our example of the Conference Buddy bot, we restricted the websites to themes related to the conference.

The bot tasks are all within a single Azure Functions app but exist as separate functions. Azure Functions is a solution for easily running small pieces of code, or "functions," in the cloud. You can write just the code you need for the problem at hand, without worrying about an entire application or the infrastructure to run it. Each bot task can also send the message to the Azure Search Questions Store to be consumed by the dashboard.

NOTE We provide sample code to help you get started on each of the common patterns and other scenarios built with the Bot Builder Samples Repository at Github.

Conversation Flow: An Example of the Conference Buddy Bot in Action

To get an idea of how the Conference Buddy bot works in action, let's examine a typical conversation flow:

1. The user invokes the Conference Bot by sending the first message.
2. The Conference Bot responds with a greeting and introduction of what it can do.
3. The user asks a question; for example, "Who is Lili Cheng?"
4. The Conference Bot routes the message to LUIS to determine the intent of the message: LUIS parses the message and, for our example, returns "This is an Ask Who Task."

5. The Conference Bot then selects the appropriate bot task within the Bot Brain to call via HTTP Post. In our example, the "Ask Who" task will do the following:
   a. Send the string to Bing Web Search and grab the results.
   b. Send the string to Bing Image Search in parallel.
   c. Combine the image and text into a response object/data contract that the Conference Bot understands.
6. The Conference Bot sends a graphical card as results to the user.
7. The Conference Bot sends results to Azure Search to be archived so that the dashboard can use it.
8. The user can click the link on the card to get more information from the source of the article.

Figure 1-7 illustrates the "Who is?" response card for "Lili Cheng."

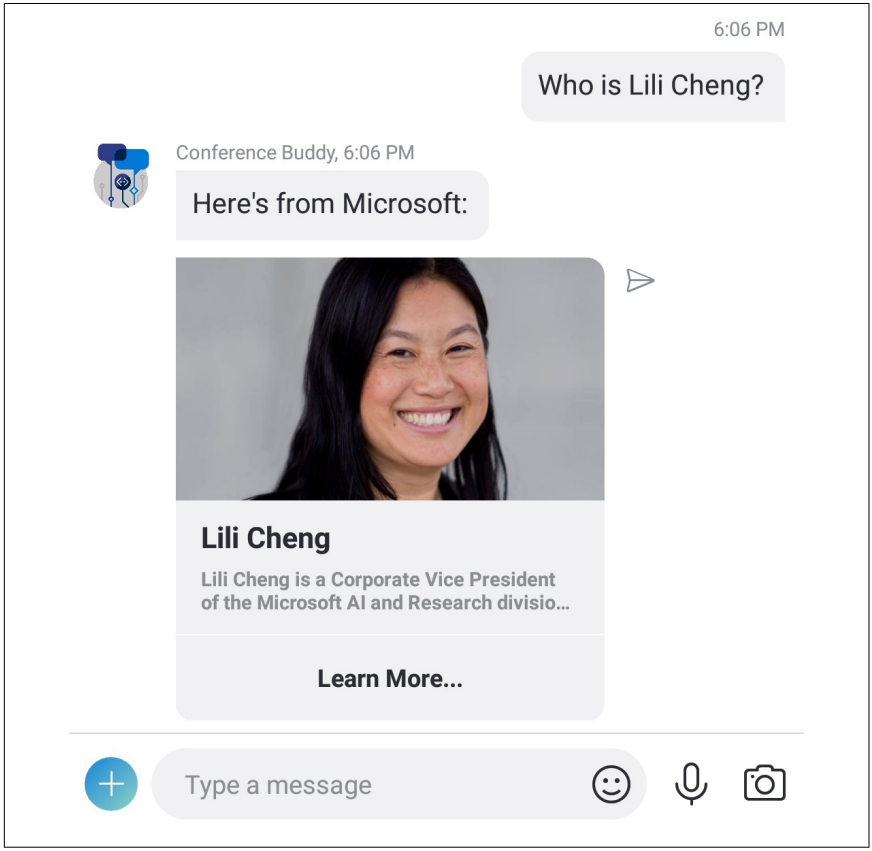

Figure 1-7. A "Who Is?" card

Let's demonstrate the "Learn More" task to illustrate this entire process:

1. Suppose that the user asks, "I want to learn more about Azure Machine Learning."
2. The Conference Bot routes the message to LUIS to determine the intent of the message: LUIS parses the message and, for our example, returns "This is a Learn More task."
3. The Conference Bot then selects the appropriate bot task to call via HTTP Post to process the message: in our example, "The Learn More Task" will call Text Analytics to extract key phrases and send parallel requests to the following:
   a. **Video Indexer**: Video Indexer is a Cognitive Service that will get the transcript of the video, break it into keywords, annotate the video, analyze sentiment, and moderate content. You can upload specific videos related to the session, and it will play the video right at the moment at which the speaker is discussing the keyword entered.
   b. **Bing Custom Search**: Enables the power of Bing Search on a restricted number of websites to increase the relevancy and speed of results. In the case of the Conference Buddy bot, we included websites that dealt only with the conference themes.
   c. **Bing Web Search**: Bing Web Search is activated in case the prior Video Indexer and Bing Custom Search don't yield any results.

Now let's look at the some of the design considerations and take a deeper dive into the bot's architecture.

Conference Buddy Bot design considerations

At a high level, a bot is like any other application or website, so the same design considerations apply for building a good user interface (UI) and user experience (UX).

In building our Conference Buddy bot, we considered the following questions in turn:

Identifying the purpose
: What is the purpose for building the bot? What is the goal of the users in interacting with the bot? In this case, we want to give the conference attendees a platform to ask questions and engage the speaker.

General bot design pattern
: What general design bot pattern does it follow? The Conference Buddy bot is an example of a Knowledge Base bot pattern—a bot that finds and returns

the information that a user has requested, with an option to escalate to a human agent. Here are some other common bot patterns and examples:

- **Task automation and completion**: This pattern enables the user to complete a specific task or set of tasks without any assistance from a human. An example might be a Password-Reset bot that can walk users through resetting their password and free up help desk personnel to focus on more complex issues. Another example might be an HR assistant bot that can field an employee's request to change their last name and update the Employee Profile and records.
- **Bot to web**: This sends the user to a web browser to complete a task and then resumes the conversation with the user after the task has been completed. Typical use cases involve handling security, authentication, and authorization. After the user is verified, the bot will then access personal data (with user approval) and continue the task at hand. An example might be a virtual insurance agent chatbot that verifies an existing customer and then helps the customer to upgrade their plan or make changes to their policy.
- **Handoff to human**: This pattern hands off via a smooth transition to a human when it identifies a scenario requiring human intervention.
- **Bots in apps**: This helps users navigate more complex apps and hands off to a human when needed. An example is a help desk user app bot that handles first response when a user interacts with it.
- **Bots in websites**: This pattern assists users in navigating complex websites and finding information quickly.

Messaging platform

On which messaging platform will the bot reside? What devices and platforms do our users care about? There are popular existing messaging channels like Skype, Facebook Messenger, Slack, Kik, and others, or you can build a custom messaging layer on an existing mobile app or website.

The key is first figuring out where your target audience spends time. Are you a popular gaming platform and want to introduce an in-game reward bot? A small business building a following on social media? A large bank with a popular mobile app? The location of your bot will also be tied to the specific reason you are building it.

To reach as many audience members as possible, we decided to make our Conference Buddy bot omnichannel. To do this, you would need to develop a special plug-in for each source that takes care of the specific protocol between the information source and your system. The Microsoft Azure Bot Service Framework allows you to connect with more than one messaging channel and receive the

information in a standard format regardless of its source. Figure 1-8 shows the Microsoft Azure Bot Service screen, on which adding new channels is a matter of several clicks.[10]

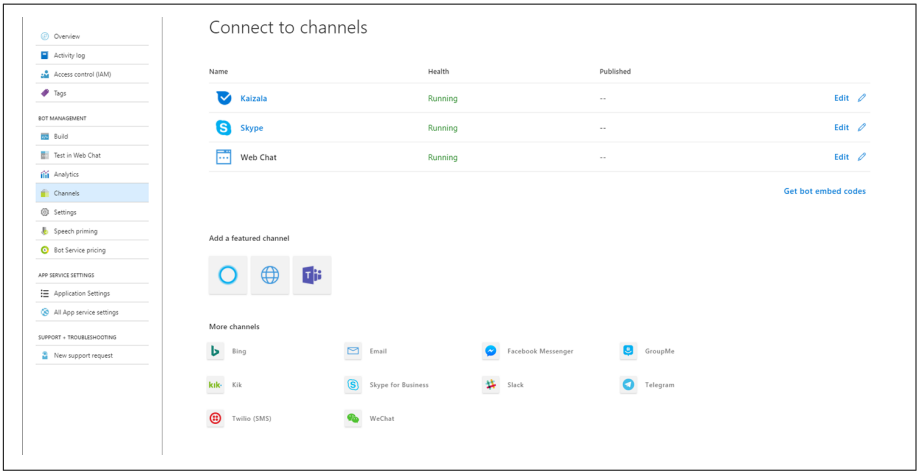

Figure 1-8. Multiple connections to channels

Overall bot architecture

How will your bot transform the information to a standard format that can be ingested for processing? And how does it return the information in a way that will be consumed by the channel? For our Conference Buddy bot, implement a data contract to abstractly describe the data to be exchanged. A data contract precisely defines, for each parameter or return type, what data is serialized (turned into XML) to be exchanged.

NOTE For general principles of bot design and more in-depth recommendations, see the Principles of Bot Design section of the Microsoft Azure Bot Service documentation.

Conference Buddy Bot Architecture Details

Let's take a deeper dive into the Conference Buddy bot architecture details and explore the code samples that power the chatbot.

10 Connect a Bot to Channels

Root Dialog

Whereas a traditional application starts with a main screen and users can navigate back to start over, with bots you have the *Root Dialog*. The Root Dialog guides the conversation flow. From a UI perspective, each dialog acts like a new screen. This way, dialogs help the developer to logically separate out the various areas of bot functionality.

For the Conference Buddy bot, each dialog invokes the next, depending on what the user types and the intent. This is called a *waterfall dialog*. A waterfall dialog is a type of dialog that allows the bot to easily walk a user through a series of tasks or collect information. The tasks are implemented as an array of functions where the results of the first function are passed as input into the next function, and so on. Each function typically represents one step in the overall process. At each step, the bot prompts the user for input, waits for a response, and then passes the result to the next step.

So, let's consider our Conference Buddy bot. If the user types:

```
"Hello there, buddy!"
```

The Root Dialog will send the string to LUIS and wait for a response. LUIS will evaluate the string and send back a JSON object with the results. For each intent, LUIS gives a confidence score, it highlights the `topScoringIntent` and identifies the entities in the query, as well. The following code shows an example response:

```
{
"query": "Hello there, buddy",
"topScoringIntent": {
  "intent": "Greeting",
  "score": 0.9887482
 },
"intents": [
   {
      "intent": "Greeting",
      "score": 0.9887482
   },
   {
      "intent": "who",
      "score": 0.04272597
   },
   {
      "intent": "learnmore",
      "score": 0.0125702191
   },

   },
 ],
 "entities": [
   {
      "entity": "buddy",
      "type": "Person",
```

```
      "startIndex": 20,
      "endIndex": 24,
      "score": 0.95678144
   }
 ]
}
```

When LUIS returns the intent as "`Greeting`", the Root Dialog processes the function "Greeting Intent." This function displays the Greeting Dialog, which in our example does not need to invoke a bot task. The control will remain with the Greeting Dialog until the user types something else. When the user responds, the Greeting Dialog closes and Root Dialog resumes control.

Now let's explore the following Root Dialog sample code to see how the rest of the intents are processed:

```
public Task StartAsync(IDialogContext context)
{
   context.Wait(MessageReceivedAsync);
   return Task.CompletedTask;
}
private async Task MessageReceivedAsync(IDialogContext context,
                 IAwaitable<object> result)
{
   try
    {
       var activity = await result as Activity;
       string message = WebUtility.HtmlDecode(activity.Text);
       if (string.IsNullOrEmpty(message) == true)
        {  return;
        }
        // Handle the explicit invocation case in Skype
       string channelId = GetChannelId(activity);

        if (channelId == "skype" &&
            message.StartsWith(activity.Recipient.Name) == true)
        {
          message =
          message.Substring(activity.Recipient.Name.Length).Trim();
        }
        else if (channelId == "skype" &&
                 message.StartsWith
                ("@" + activity.Recipient.Name) == true)
        {
       message =
       message.Substring
      (activity.Recipient.Name.Length + 1).Trim();
     }
     // Handle intents
     LUISResult luisResult = await GetEntityFromLUIS(message);
        string intent =
       luisResult.intents?.FirstOrDefault()?.intent   ??string.Empty;
        string[] entities =
```

```
luisResult.entities?.Select
(e => e.entity)?.ToArray() ?? new string[0];
   if (intent == "greeting")
   {
     await ProcessGreetingIntent(context, message);
   }
   else if (intent == "who")
   {
     await ProcessQueryIntent
     (context, activity, BotTask.AskWho, message, entities);
   }
   else if (intent == "learnmore")
   {
     await ProcessQueryIntent
    (context, activity, BotTask.AskLearnMore, message, entities);
    }
   else
   {
     await ProcessQueryIntent(
           context, activity,
           BotTask.AskQuestion, message, entities);
   }
```

The Root Dialog does not get invoked unless a user types a message. When the Conference Buddy bot receives the first message, we do a special handling in the code for messages coming from the Skype channel.

We discussed what happens when LUIS returns the Greeting Intent. In our example chatbot, we anticipate three other possible intents from LUIS:

- If the intent is "Who," the Root Dialog posts the question to the bot task "Ask Who."
- If the intent is "Learn More," the Root Dialog posts the question to the bot task "Learn More."
- For all other intents, the Root Dialog sends the text to the "Ask Question" message.

At this point, the Root Dialog hands control to the appropriate bot task.

The Bot Brain abstraction layer

The abstraction layer handles the Post call to a bot task within the Bot Brain. This is where the benefit of the microservices implementation becomes clear. The Root Dialog has handled the message and LUIS processed the intent. At this level, the Bot executes the relevant bot task.

Let's explore the code:

```
private static async
Task<string>ProcessQueryIntent(IDialogContext context,
```

```
      Activity activity,BotTask task, string query, string [] topics)
   {
     // Prepare the request to invoke a bot task within the bot brain
     AskQuestionRequest request = new AskQuestionRequest()
{
       ConversationId = activity.Conversation.Id,
       Question = query,
       SessionId = SessionId,
       Topics = topics != null ? topics.ToArray() : new string[0],
       UserId = string.IsNullOrEmpty(activity.From.Name)
      == false ? activity.From.Name : activity.From.Id
 };
 // Invoke the bot task to process the request
     AskQuestionResponse askQuestionResponse =
    await
   HttpClientUtility.PostAsJsonAsync
   <AskQuestionResponse>
   (new Uri(BotBrainUrl + task.ToString()), RequestHeaders, request);
    // Handle the response returned from the
        bot task to be shown as cards depending on channel
  if (askQuestionResponse.Results?.Count() > 0 == true)
     {
         IMessageActivity foundMsg = context.MakeMessage();
         AskQuestionResult result = askQuestionResponse.Results[0];
         if (string.IsNullOrEmpty(result.Source) == false)
         {foundMsg.Text = string.Format
         ("Got it. Meanwhile, from {0}:", result.Source);
         }
        else
         {
          foundMsg.Text = "Got it. Meanwhile, here's what I found:";
             }
           await context.PostAsync(foundMsg);
           IMessageActivity cardMessage;
           string channelId = GetChannelId(activity);
           if (channelId == "kaizala")
           {
              cardMessage = await
              GetKaizalaCardMessage(context, request, result);
           }
           else if (channelId == "directline" || channelId == "emulator")
          {
           cardMessage =
           GetAdaptiveCardMessage(context, request, result);
          }
           else
          {
           cardMessage = GetHeroCardMessage(context, request, result);
          }
           await context.PostAsync(cardMessage);
          }
           else if (task != BotTask.AskQuestion)
         {
```

```
        IMessageActivity notFoundMsg = context.MakeMessage();
        notFoundMsg.Text =
        "I can't seem to find it.
         Can you rephrase the question and try again?";
        await context.PostAsync(notFoundMsg);
       }

    return "success";
    }
```

What's important in this layer, no matter which bot task is called, the request, invocation, and response are handled the same way. The Data Contract called `AskQuestionRequest` combines the `ConversationID`, `Query`, `SessionID`, and `UserID` to pass to the bot task through an HTTP Post.

The HTTP Post is the call into a bot task within the Bot Brain. When the appropriate bot task executes the query, it prepares the response in the `AskQuestionResponse` where no matter which bot task, the response is handled generically.

Because the Conference Buddy bot is omnichannel, the response card is displayed differently according to the channel; the last part of the code shows how the bot implements adaptive cards.

The Data Contract

Without the Data Contract, there will be no abstraction layer at all. The Data Contract code that follows acts as the formal agreement between the bot and Bot Brain and abstractly describes the data to be exchanged.

Let's explore the code and see the details behind how the `AskQuestionRequest`, which specifies the details to be sent with each query, and the details behind the `AskQuestionResponse`, which specifies the details for each response, no matter what the bot task does:

```
 namespace ConferenceBuddy.Common.Models
{
[DataContract]
    public class AskQuestionRequest
    {
      /// <summary>
     /// The session identifier
    /// </summary>
    [DataMember(Name = "sessionId")]
    public string SessionId { get; set; }
    /// <summary>
    /// The conversation identifier
    /// </summary>
    [DataMember(Name = "conversationId")]
    public string ConversationId { get; set; }
    /// <summary>
    /// The user identifier
```

```
   /// </summary>
   [DataMember(Name = "userId")]
    public string UserId { get; set; }
    /// <summary>
    /// The text of the question
   /// </summary>
   [DataMember(Name = "question")]
    public string Question { get; set; }
    /// <summary>
   /// The topics of the question
  /// </summary>
[DataMember(Name = "topics")]
   public string [] Topics { get; set; }
   }
   [DataContract]
   public class AskQuestionResponse
   {
   /// <summary>
   /// The unique id of the response
   /// </summary>
   [DataMember(Name = "id")]
   public string Id { get; set; }

    /// <summary>
   /// The results of the response
   /// </summary>
   [DataMember(Name = "results")]
   public AskQuestionResult [] Results { get; set; }
   }
[DataContract]
 public class AskQuestionResult
   {
    /// <summary>
    /// The title of the result
    /// </summary>
  [DataMember(Name = "title")]
   public string Title { get; set; }

    /// <summary>
       /// The answer of the result
      /// </summary>
 [DataMember(Name = "answer")]
 public string Answer { get; set; }
    /// <summary>
    /// The image url of the result
    /// </summary>
    [DataMember(Name = "imageUrl")]
     public string ImageUrl { get; set; }
     /// <summary>
     /// The source of the result
     /// </summary>
    [DataMember(Name = "source")]
    public string Source { get; set; }
```

```
    /// <summary>
   /// The url of the result
   /// </summary>
  [DataMember(Name = "url")]
  public string Url { get; set; }

     /// <summary>
    /// The url display name of the result
   /// </summary>
       [DataMember(Name = "urlDisplayName")]
       public string UrlDisplayName { get; set;
      }
 }
```

The Data Contract allows the separation of functions between how a query is processed and how the response is generated. Think of the Data Contract as the postal carrier. From the postman's perspective, the specific details of the contents in the letter/package are irrelevant. What matters is the format of the "To" and "From" address to allow delivery to the right location.

If we had to make different HTTP calls to each bot task, the Conference Buddy bot will be unwieldy and difficult to build, test, deploy, and scale. In the next section, we see how the microservices implementation makes it simpler to develop the Bot Brain's intelligence and teach our Conference Buddy bot new skills.

Adding "Plug and Play" Intelligence to Your Bot

We can teach our Conference Buddy bot new skills by developing the Bot Brain's intelligence. So far, we have built a Conference Buddy bot that has three main bot tasks:

- Ask Who task
- Learn More task
- Answer Question task

We built the Conference Buddy architecture in a flexible way, so a developer can easily add more bot tasks. So, let's expand on our Conference Buddy bot scenario. Suppose that the conference is broadcast globally and the audience members hail from different countries and speak different languages, whereas the speaker understands only English. You might want to add a new task to allow your bot to handle questions in different languages and translate the question to English for the speaker to address.

For our bot, we will make an additional call to Cognitive Services: Microsoft Translator. This is a machine translation service that supports more than 60 languages. The developer sends source text to the service with a parameter indicat-

ing the target language, and the service sends back the translated text for the client or web app to use.

The translated text can now be used with the other Cognitive Services that we have used so far, such as text analytics and Bing web search.

To make a call to a new Cognitive Service, you need to log in to your Azure portal. This Quick Guide walks you through editing the bot code and using Azure Functions to invoke various APIs. In the sample code that follows, we illustrate how to add the new translation bot task.

Let's explore the code:

```
[FunctionName("AskQuestion")]
 public static async Task<HttpResponseMessage>
 Run(
     [HttpTrigger(AuthorizationLevel.Function, "post", Route =
      "AskQuestion")]HttpRequestMessage request,
      [Table("Session", Connection =
      "AzureWebJobsStorage")]ICollector<SessionTableEntity> sessionTable,
      TraceWriter log)
 {
     MediaTypeHeaderValue contentType = request.Content.Headers.ContentType;

     // Check if content type is empty
     if (contentType == null)
     {
      return request.CreateResponse
     (HttpStatusCode.BadRequest, "Missing content-type from header.");
     }
    else if (contentType.MediaType.Contains("application/json") == false)
    {
        return request.CreateErrorResponse
        (HttpStatusCode.UnsupportedMediaType,
        string.Format("Request's content type ({0}) is not supported.",
        string.Join(", ", contentType.MediaType)));
    }

    // Read content from request
    AskQuestionRequest requestBody = await
    request.Content.ReadAsAsync<AskQuestionRequest>();

    // Verify content contains a valid image uri
    if (string.IsNullOrEmpty(requestBody.Question) == true)
    {
      return request.CreateResponse(HttpStatusCode.BadRequest,
      "Question is missing from the request content.");
    }
    else if (string.IsNullOrEmpty(requestBody.SessionId) == true)
    {
    return request.CreateResponse(HttpStatusCode.BadRequest,
          "Session id is missing from the request content.");
     }
```

```
    // Translate question
    requestBody.Question = await
    ServicesUtility.Translator.TranslateTextAsync(requestBody.Question);

   // Answer the question
   AskQuestionResponse response =
    await AnswerQuestion(requestBody, sessionTable);

    // Return request response with result and 200 OK
    return request.CreateResponse(HttpStatusCode.OK, response);
}

 public static async
 Task<AskQuestionResponse> AnswerQuestion(AskQuestionRequest request,
  ICollector<SessionTableEntity> sessionTable)
{
    // Get unique identifier
    string id = Guid.NewGuid().ToString();
    DateTime timestampUtc = DateTime.UtcNow;

    // Run keyphrases extraction
     request.Topics = await ServicesUtility.GetTopics
     (request.Question, request.Topics);

     // Run search services
    string queryWithTopics = request.Topics?.Count() > 0 ?
    string.Join(" ", request.Topics).Trim() : request.Question;

  Task<BingWebSearchResult> bingWebSearchTask =
   ServicesUtility.BingSearch.SearchWebAsync
  (query: request.Question, count: SettingsUtility.MaxResultsCount);
  Task<BingWebImagesResult> bingWebImagesTask =
   ServicesUtility.BingSearch.SearchImagesAsync
  (query: request.Question, count: SettingsUtility.MaxResultsCount);

  await Task.WhenAll(bingWebSearchTask, bingWebImagesTask);

     BingWebSearchResult bingWebSearchResult = bingWebSearchTask.Result;
     BingWebImagesResult bingWebImagesResult = bingWebImagesTask.Result;

     // Process results
     AskQuestionResponse response = new AskQuestionResponse()
     {
     Id = id,
    Results = new AskQuestionResult[0]
     };

    if (bingWebSearchResult.WebPagesResult?.Values?.Count() > 0)
    {
       response.Results =
       ServicesUtility.GetAskQuestionResults(bingWebSearchResult);
     }
```

```
    if (response.Results.Any(r => string.IsNullOrEmpty(r.ImageUrl)
     == true) == true && bingWebImagesResult?.Values?.Count()
    > 0 == true)
{
     response.Results =
     ServicesUtility.AddImageToResults(response.Results,
     bingWebImagesResult);
}

  // Upload search document
 await
 ServicesUtility.UploadDocumentToSearchService
 (SettingsUtility.AzureSearchIndexName,
      new SessionSearchDocument
      (id, timestampUtc, "AskQuestion", request, response));

      // Write to session table
      sessionTable.Add(new SessionTableEntity
     (id, timestampUtc, "Question", request, response));

    return response;
 }
```

In the first part of the code, the function `AskQuestion` reads the content from the request and translates the question into English using the Translator. It then extracts the Key Phrases using Text Analytics and sends the query to Bing Web Search and Bing Image Search to create a card for the response. The Key Phrases go to Azure Search to power the bot's analytics as well as the dashboard. In this example, we do not translate the response back into the original language, but that could be an option for other implementations.

Now that we have successfully added a new bot task, we can continue to develop the Bot Brain's intelligence to add more abilities like adding other APIs such as vision, speech, and more through our Cognitive Services. Let's explore the Conference Buddy dashboard.

Building an Enterprise App to Gain Bot Insights: The Conference Buddy Dashboard

The Conference Buddy dashboard is part of the Conference Buddy bot. The dashboard acts as the question and answer repository for both conference attendees and the speaker to explore. The Conference Buddy dashboard (see Figure 1-9) does the following:

- Displays questions asked from all audience members in real time

- Allows the speaker to quickly search, sort, or filter the results by a Session, Bot Skills, or Topic to view relevant questions submitted

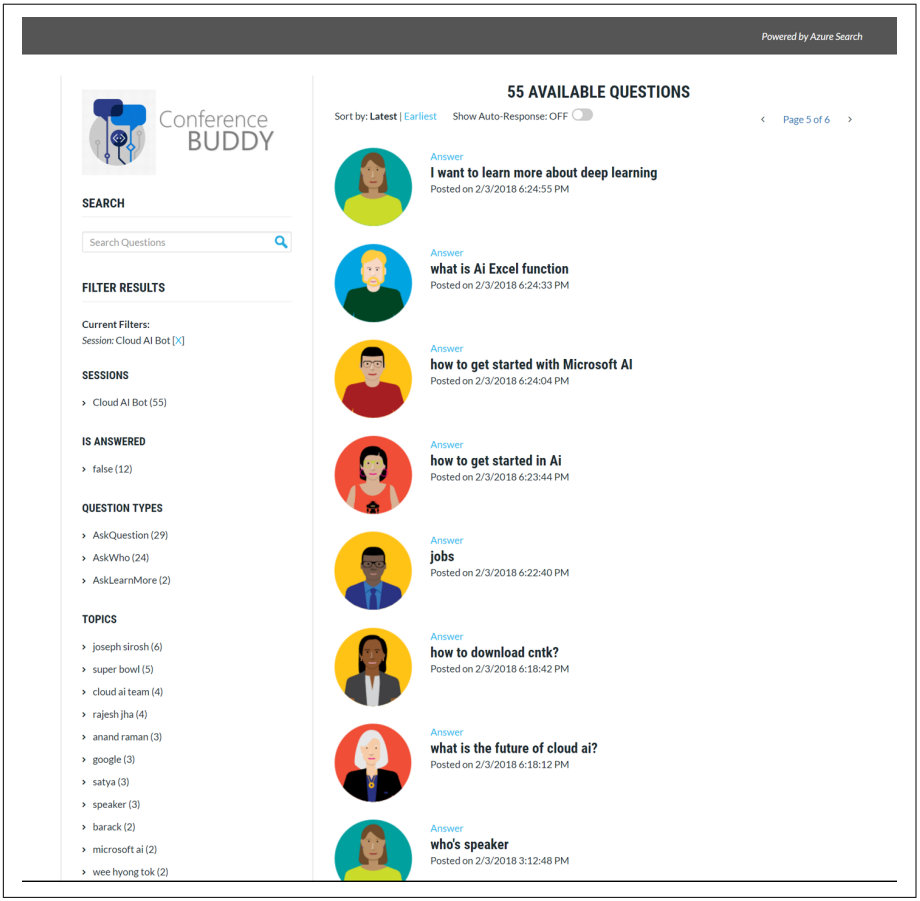

Figure 1-9. The Conference Buddy dashboard app

We built the dashboard using the ASP.NET Core MVC, which is a rich framework for building web apps and APIs using the Model-View-Controller design pattern. You can find information here to guide you through building a Web App similar to our Conference Buddy dashboard.

Many web apps will need a search capability for the application content. Having an easy-to-use search API in the cloud can be a big boon to developers. We embed Azure Search in our Conference Buddy dashboard to search the questions being asked. Azure Search is a simple Search-as-a-Service API that allows developers to embed a sophisticated search experience into web and mobile applications without having to worry about the complexities of full-text search and without having to deploy, maintain, or manage any infrastructure.

For the Conference Buddy dashboard, Azure Search does the following:

- Powers the search functionality
- Indexes the key phrases extracted to appear as topics
- Allows results to be filtered and sorted for ease of referencing

The following sample code shows the call into Azure Search:

```
public class SessionSearchService
{
    private ISearchIndexClient IndexClient;

    public SessionSearchService()
    {
     string searchServiceName =
     ConfigurationManager.AppSettings["SearchServiceName"];
     string index =
     ConfigurationManager.AppSettings["SearchServiceIndexName"];
     string apiKey =
     ConfigurationManager.AppSettings["SearchServiceApiKey"];

      SearchServiceClient searchServiceClient = new
      SearchServiceClient
     (searchServiceName, newSearchCredentials(apiKey));
      this.IndexClient =
      searchServiceClient.Indexes.GetClient(index);
      }
 public SessionSearchService()
 {
  string searchServiceName =
   ConfigurationManager.AppSettings["SearchServiceName"];
   string index =
   ConfigurationManager.AppSettings["SearchServiceIndexName"];
   string apiKey
   = ConfigurationManager.AppSettings["SearchServiceApiKey"];

   SearchServiceClient searchServiceClient = new
   SearchServiceClient
  (searchServiceName,new SearchCredentials(apiKey));
   this.IndexClient = searchServiceClient.Indexes.GetClient(index);
   }

   public async Task<DocumentSearchResult>
                 SearchAsync(string searchText,  stringsessionFacet,
                string topicsFacet, string skillFacet,
  string isAnsweredFacet, int currentPage,
 int numResultsPerPage, bool sortByLatest)
 {
   // Execute search based on query string
  try
   {
      if (string.IsNullOrEmpty(searchText) == true)
```

```
  {
    searchText = "*";
   }

 SearchParameters sp = new SearchParameters()
  {
   SearchMode = SearchMode.Any,
   Top = numResultsPerPage,
   Skip = currentPage * numResultsPerPage,
     // Limit results
     Select = new List<String>()
     {
      "id", "userId", "sessionId", "question",
     "skill", "topics","timestampUtc", "answerTitle",
      "answerContent",  "answerImageUrl", "answerSource",
     "answerUrl", "answerUrlDisplayName", "isAnswered"
      },
     // Add count
     IncludeTotalResultCount = true,
     // Add facets
    Facets = new List<String>()
     { "sessionId,count:0", "topics,count:20",
     "skill,count:0", "isAnswered,count:0" },
      MinimumCoverage = 75
     };

     string orderBy = sortByLatest == true ? "desc" : "asc";

 sp.OrderBy = new List<String>() { "timestampUtc " + orderBy };

 // Add filtering
 IList<string> filters = new List<string>();

if (string.IsNullOrEmpty(sessionFacet) == false)
{
  filters.Add(string.Format
 ("sessionId eq '{0}'", sessionFacet));
}

if (string.IsNullOrEmpty(skillFacet) == false)
{
    filters.Add(string.Format("skill eq '{0}'", skillFacet));
}

if (string.IsNullOrEmpty(topicsFacet) == false)
{
     filters.Add(string.Format
     ("topics/any(kp: kp eq '{0}')", topicsFacet));
 }

if (string.IsNullOrEmpty(isAnsweredFacet) == false)
  {
  filters.Add(string.Format
```

```
      ("isAnswered eq {0}", isAnsweredFacet));
      }

      sp.Filter =
      filters.Count() > 0 ?
      string.Join(" and ", filters) : string.Empty;

      return await
      this.IndexClient.Documents.SearchAsync(searchText, sp)
}
     catch (Exception ex)
    {
      Console.WriteLine
       ("Error querying index: {0}\r\n", ex.Message.ToString());
    }
    return null;
   }
```

Bot Insights: Instrumenting your Bot

Most web applications and bots will want to analyze the usage and other statistics of the bot. Such analytics can also help detect and diagnose exceptions and application performance issues. The Azure Bot Service provides Bot Analytics, which is an extension of Azure Application Insights. Application Insights provides *service-level and instrumentation data*, such as traffic, latency, and integrations. Bot Analytics provides *conversation-level* reporting on user, message, and channel data. Bot Analytics affords you the full benefit of insights without having to write a single line of code.

To enable Analytics on the bot (see Figure 1-10), do the following:

1. Install a small instrumentation package in your application and set up an Application Insights resource in the Azure portal. The instrumentation monitors your app and sends telemetry data to the portal. The application can run anywhere—it doesn't need to be hosted in Azure. Follow the steps in the "Create an Application Insight Resource" guide.
2. Open the bot in the dashboard. Click Settings and scroll down to the Analytics section.
3. Type the information to connect the bot to Application Insights. All fields are required.

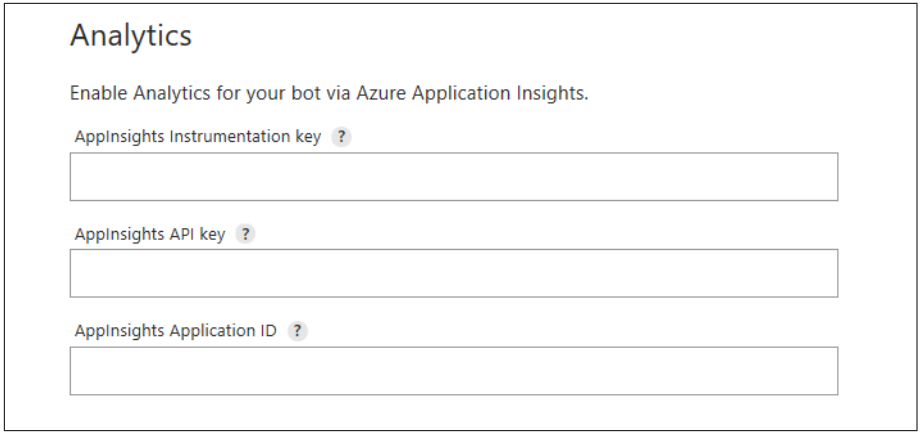

Figure 1-10. Enable Analytics screen

AppInsights Instrumentation Key

To find this value, open Application Insights, and then navigate to Configure → Properties.

AppInsights API key

Provide an Azure App Insights API key. Learn how to generate a new API key. Only Read permission is required.

AppInsights Application ID

To find this value, open Application Insights, and then navigate to Configure → API Access.

View Analytics for the bot

To access Analytics, open the bot in the developer portal, and then click Analytics.

Analytics allows you to specify messages/data via the following:

Channel
: You can choose which channels appear in the graphs. Note that if a bot is not enabled on a channel, there will be no data from that channel.

Time period
: Analysis is available for the past 90 days only. Data collection began when Application Insights was enabled.

Grand totals
: Active users and messages sent.

Retention

How many users sent a message and came back, as demonstrated in Figure 1-11.

RETENTION - % USERS WHO MESSAGED AGAIN (LAST 10 DAYS)

		Days later									
Date	Users	1	2	3	4	5	6	7	8	9	10
4/23/2017	0	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
4/24/2017	13	31%	8%	8%	8%	8%	8%	8%	15%	8%	
4/25/2017	43	2%	2%	2%	5%	5%	2%	5%	2%		
4/26/2017	27	4%	4%	4%	4%	4%	4%	4%			
4/27/2017	37	3%	3%	3%	3%	3%	3%				
4/28/2017	22	5%	5%	5%	5%	5%					
4/29/2017	23	9%	9%	9%	9%						
4/30/2017	14	7%	7%	7%							
5/1/2017	17	18%	12%								
5/2/2017	87	9%									

Figure 1-11. Insights screen showing users who messaged again

Users

The Users graph tracks how many users accessed the bot using each channel during the specified time frame, as shown in Figure 1-12.

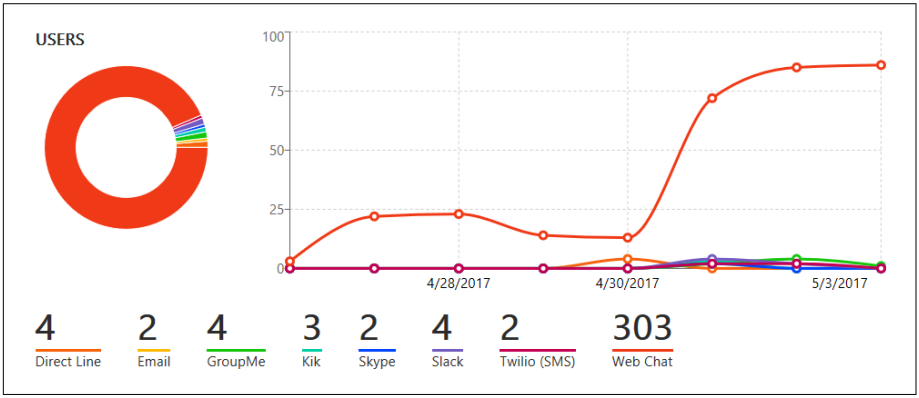

Figure 1-12. Insights screen showing users

Messages

The Message graph tracks how many messages were sent and received using a given channel during the specified time frame (Figure 1-13).

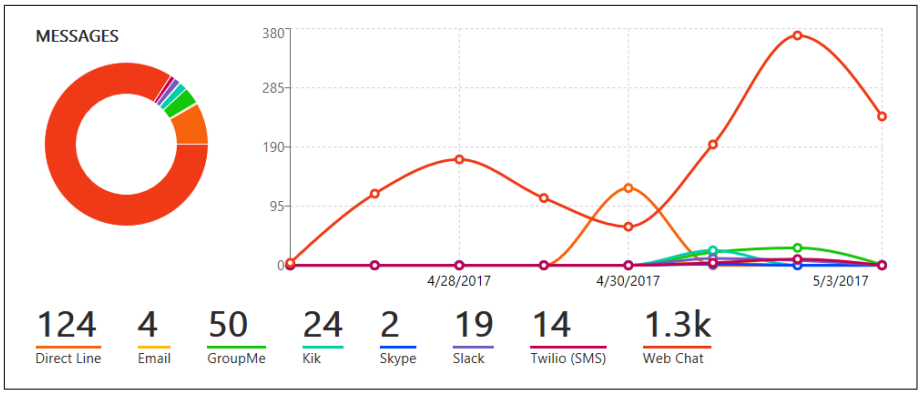

Figure 1-13. Insights screen showing messages

Paving the Road Ahead

Our commitment to democratizing AI goes beyond the cloud AI platform. In partnership with Facebook, we recently introduced the Open Neural Network Exchange (ONNX, also see *https://onnx.ai/*) format, an open source standard for representing deep learning models that enables models to be ported between frameworks. Caffe2, PyTorch, Microsoft Cognitive Toolkit (CNTK), Apache MXNet, and other tools are developing ONNX support, and a long list of partners such as AMD, Nvidia, IBM, Intel, and AWS have announced support. ONNX is the first step toward an open ecosystem in which AI developers can easily move between state-of-the-art tools and choose the combination that is best for them.

By providing a common representation of the computation graph, ONNX helps developers choose the appropriate AI framework for their task, allows authors to focus on innovative enhancements, and enables hardware vendors to streamline optimizations for their platforms.

In the latest update to Windows 10, developers will be able to use AI to deliver more powerful and engaging experiences. Data scientists can train and deliver ONNX models for use by Windows developers using Azure Machine Learning. This is part of Microsoft's overall investment in delivering a great development experience for AI developers on Windows.

Microsoft believes in bringing AI advances to all developers, on any platform, using any language, and with an open AI ecosystem that helps us to ensure that the fruits of AI are broadly accessible.

Beyond paving the path in providing tools and resources to democratize AI, we are also engaging in the challenging questions these powerful new technologies

are forcing us to confront. "*We become what we behold. We shape our tools and then our tools shape us*."[11]

How do we ensure that AI is designed and used responsibly? How do we establish ethical principles to protect people? How should we govern its use? And how will AI affect employment and jobs? These questions cannot be answered by technologists alone; it is a societal responsibility that bears discussions across government, academia, business, civil society, and other stakeholders.

We recently published a book, *The Future Computed: Artificial Intelligence and its Role in Society*, that addresses the larger issues governing AI and the future. It also outlines what we at Microsoft have identified as our six ethical principles to guide the cross-disciplinary development and use of artificial intelligence:

Fairness

The first principle underscores issues of social justice. How do we design AI systems to treat everyone in a fair and balanced way, when the training data we use might be marred with assumptions and biases? Can an AI system that provides guidance on loan application or employment, for instance, be designed to make the same recommendations for everyone with similar financial circumstances or professional qualifications? As a developer, you will need to be cognizant of how biases can be introduced into the system, work on creating a representational dataset, and beyond that educate end users to understand that sound human judgement must complement computer system recommendations to counter their inherent limitations.

Reliability and Safety

As we become more dependent on AI systems, how can we ensure that our systems can operate safely, reliably, and consistently? Consider a failure of an AI-powered system in a hospital, which literally means the difference between the lives and deaths of people. We must conduct rigorous design and testing under various conditions, including security considerations on how to counter cyberattacks and other malicious intents. Sometimes, systems might react unexpectedly, depending on the data. At Microsoft, we had a painful example of unexpected behavior when we unveiled Tay, a conversational chatbot on Twitter. Tay was an experiment in conversational AI that quickly went wrong when users began feeding Tay racist and sexist comment that she quickly learned and reflected back. We took down Tay within 24 hours. Developers must teach end users what the expected behaviors within normal conditions are so that when things go wrong the humans can quickly intervene to minimize damage.

11 There is an ongoing debate about who the originator of this quote is: Marshall McLuhan, Winston Churchill, and Robert Flaherty are among them. Check this link for the evolution of the discussion.

Privacy and Security

As our lives become more digitized, privacy and security take on additional significance. This discussion goes beyond what technologies are used to ensure the security of data; it must include regulations around how data is used and for what reasons.

Inclusivity

We want to ensure AI systems empower and engage people across the spectrum of abilities and access levels. AI-enabled services are already empowering people struggling with visual, hearing, and cognitive disabilities. Building systems that are context aware with increasing emotional intelligence will pave the path for more empowerment.

Transparency and Accountability

Designers and developers of AI systems need to create systems with maximum transparency. People need to understand how important decisions regarding their lives are made. With accountability, internal review boards need to be created to give guidance and oversight on the systems they are in charge of.

If "developers are writing the script for the future," as Joel Spolsky, the CEO of Stack Overflow stated, that puts you, the developer, in the front seat of this larger conversation. "Every line of code is a decision made by the developer. We are building the future out of those decisions."

At Microsoft, we are committed to empowering you with the tools, ethical framework, and best practices to foster responsible development of AI systems that will enrich human lives and power innovations, which will in turn solve our most pressing problems today and anticipate the ones to come in the future. Finally, it's important to remember that while many times we get bogged down in the discussions of the exciting algorithms and tools, the real power of AI resides in the ideas and questions that precede it. It's the conservationist pondering how to create sustainable habitats, the doctor wondering how to better serve a patient, the astronomer's and citizen scientist's curiosity that expands our collective consciousness to the outer limits of the universe. AI has the potential to empower the noblest of human causes and we are just at the very beginning of an exciting technological transformation.

Acknowledgments

Joseph Sirosh, Wilson Lee, and Vinod Anantharaman

About the Authors

Anand Raman is the Chief of Staff for AI Platform and head of AI Ecosystem at Microsoft. Previously he was the Chief of Staff for Microsoft Azure Data Group covering Data Platforms and Machine Learning. In the last decade, he ran the product management and the development teams at Azure Data Services, Visual Studio, and Windows Server User Experience teams at Microsoft.

Wee Hyong Tok is a Principal Data Science Manager with the AI Platform team at Microsoft. He leads the Engineering and Data Science team for the AI for Earth program.

Wee Hyong has worn many hats in his career—developer, program/product manager, data scientist, researcher, and strategist, and his track record of leading successful engineering and data science teams has given him unique superpowers to be a trusted AI advisor to many customers.

Wee Hyong coauthored several books on artificial intelligence, including the first book on *Predictive Analytics Using Azure Machine Learning* and *Doing Data Science with SQL Server*. Wee Hyong holds a Ph.D. in computer science from the National University of Singapore.