Microsoft Conversational AI Platform for Developers

End-to-End Chatbot Development from Planning to Deployment

—

Stephan Bisser

Apress®

Microsoft Conversational AI Platform for Developers

End-to-End Chatbot Development from Planning to Deployment

Stephan Bisser

Apress®

Microsoft Conversational AI Platform for Developers: End-to-End Chatbot Development from Planning to Deployment

Stephan Bisser
Gratwein, Austria

ISBN-13 (pbk): 978-1-4842-6836-0 ISBN-13 (electronic): 978-1-4842-6837-7
https://doi.org/10.1007/978-1-4842-6837-7

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Printed on acid-free paper

I dedicate this book to my family, especially to my wife Nicole,
my two beautiful kids Nele and Diego,
to my father whom I miss very much, and to my mother.
Ich liebe euch!

Table of Contents

About the Author

Stephan Bisser is a technical lead at Solvion and a Microsoft MVP for artificial intelligence based in Austria. In his current role, he focuses on conversational AI, Microsoft 365, and Azure. He is passionate about the conversational AI platform and the entire Microsoft Bot Framework and Azure Cognitive Services ecosystem. Stephan and several other MVPs founded the Bot Builder Community, which is a community initiative helping Bot Framework developers with code samples and extensions. Together with Thomy Gölles, Rick Van Rousselt, and Albert-Jan Schot, Stephan is hosting SelectedTech, where they publish webinars and videos on social media around SharePoint, Office 365, and the Microsoft AI ecosystem. In addition, he blogs regularly and is a contributing author to *Microsoft AI MVP Book*.

He can be reached at his blog (`https://bisser.io`), at Twitter (@stephanbisser), on LinkedIn (`www.linkedin.com/in/stephan-bisser/`), and on GitHub (stephanbisser).

About the Technical Reviewer

Fabio Claudio Ferracchiati is a senior consultant and a senior analyst/developer using Microsoft technologies. He is a Microsoft Certified Solution Developer for .NET, a Microsoft Certified Application Developer for .NET, a Microsoft Certified Professional, and a prolific author and technical reviewer. Over the past ten years, he's written articles for Italian and international magazines and coauthored more than ten books on a variety of computer topics.

Acknowledgments

I want to thank my employer, **Solvion**, for being the best company making a lot of things possible which seemed to be impossible a while ago. Of course, I want to thank every colleague as our work-friendship is special. But I need to thank you, **Thomy**, in particular, as it is an honor to work together with you, not only on company topics but also on the community bits and pieces. With that I also may thank the whole MVP and non-MVP community, especially my friends **Rick** and **Appie** who keep me busy with new ideas and projects!

Thank you, **Herbert** and **Brigitte**, for being the best in-laws I could have wished for!

I want to give thanks to my dad for teaching me everything he knew, which is the reason why I am who I am today! *Ich vermisse dich*, **Papa**.

Thank you, Mama, for being here for me in my life and supporting me wherever you can! *Danke*, **Mama**!

I want to thank my two beautiful kids, **Diego** and **Nele**, for keeping me fit and bringing incredibly much joy into my life. Without you I would miss something. *Ich liebe euch beide!*

And lastly, I want to say thank you to my wife, **Nicole**, for supporting me no matter what. I know that it is sometimes hard to be with me, but I could not be happier to have you by my side. You are the best! *Ich liebe dich!*

Introduction

This book covers all steps which you will encounter when building a chatbot using the Microsoft conversational AI platform. You will learn the most important facts and concepts about the Microsoft Bot Framework and Azure Cognitive Services, which are needed to develop and maintain a chatbot. This book is mainly targeting developers and people who have a development background, as you will learn the concepts of modern end-to-end chatbot development. But as it covers the basic concepts as well, nearly every IT-savvy person who is interested in chatbot development can benefit from reading this book.

Starting with the theoretical concepts, the first three chapters will give detailed insights into the Microsoft conversational AI platform, the Microsoft Bot Framework, and Azure Cognitive Services, which are the services and platforms used throughout this book when it is about developing chatbots. Chapters 4–8 then cover the various stages of every chatbot development project, starting with the design phase where design principles are discussed, followed by the build stage where a real-world example chatbot will be developed using Microsoft Bot Framework Composer, which is a visual bot editing tool. After that, the test phase will be addressed to see how to properly test a chatbot before the last two chapters that will be dealing with the publish and the connect phase in order to offer the chatbot to end users.

CHAPTER 1

Introduction to the Microsoft Conversational AI Platform

Conversational AI (short for artificial intelligence) is a term which can be described as a subfield or discipline of artificial intelligence in general. This area deals with the question on how to expose software or services through a conversational interface, enhanced by AI algorithms. Those conversational interfaces usually range from simple text-based chat interfaces to rather complex voice and speech interfaces. The primary goal is to let people interact with software services through those conversational interfaces to make the interaction between humans and machines more natural. An example of that would be the following:Imagine you sit in your car and want to regulate the heating. In the present you would likely need to push a button or turn a switch within your car to turn the heating either up or down. In a more modern scenario, it would be far more natural to just say something like “Car, please turn the heating to 22 degrees Celsius” to achieve the same. In that second case, you would not need to push a button which distracts you from driving; you would only need to speak and tell the device what it should do. Therefore, you would not only have the possibility to stay focused on the main thing you do which is driving, but this could also be beneficial as it may be faster to say something rather than to control a system via buttons or switches.

This first chapter of the book will outline the key concepts and principles of conversational AI focusing on the Microsoft ecosystem, along with usage scenarios for conversational AI services.

© Stephan Bisser 2021
S. Bisser, *Microsoft Conversational AI Platform for Developers*, https://doi.org/10.1007/978-1-4842-6837-7_1

Key Concepts of Conversational AI

When talking about “Conversational AI,” you always need to think about an orchestration of various components. Those components need to be aligned to deliver the best possible conversational interface for the people using it. In many cases, a comprehensive conversational AI interface may consist of the following:

- Natural language processing (NLP)
- Text analytics
- Text-to-speech
- Speech-to-text
- Text/speech translation

But why is that so important nowadays? The answer is rather simple if you look at what people were using in the past. Figure 1-1 outlines that there is a bit of a paradigm shift happening currently, as it’s more important than ever to have a conversation with a service rather than “just” clicking and browsing through.

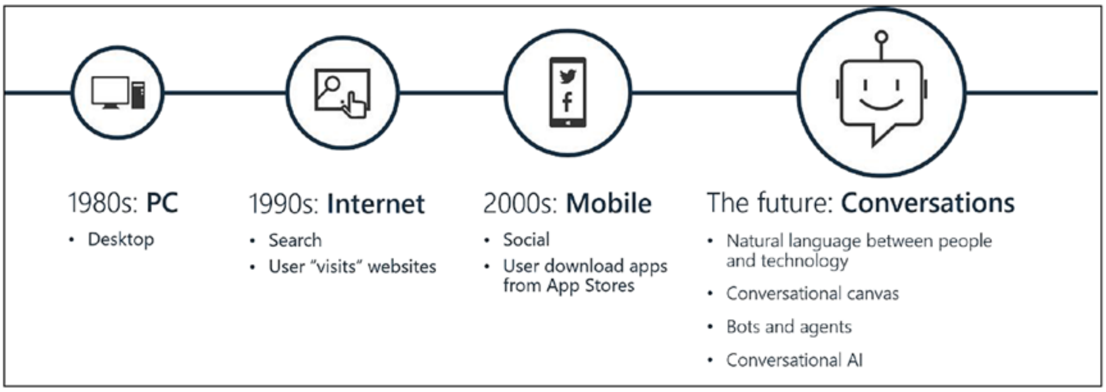


Figure 1-1. *Evolution of IT*

Many decades ago, when PCs became popular, the main use case for personal computers has been to work on different tasks on your own, without having the ability to connect with other people via your computer. With the introduction of the Internet, this evolved a bit, as you then could connect your computer with others. This led to the possibility for every user to interact with others via the Internet. In the 2000s, the mobile era began, where people started using smartphones and social media apps to connect via their mobile devices with others. From that point on, people did not need to use a personal computer or laptop anymore to interact with other people. You could just use your smartphone and start a chat over various social media channels with others and have a conversation with them. But until then, it has always been about the interaction between one person and another using different communication channels and services.

This is now slightly changing as there is now a new era starting, where the main focus is on the ability of humans having conversations not only with other people but also with the services they are using in a human way. Imagine a rather simple but quite unusual experience if you would not interact with your coffee machine as you are used to right now, clicking buttons to make yourself a coffee. With the new approach, it could be possible to talk to your coffee machine and say, "I want to have a cappuccino. Could you please make one?", instead. And this approach could also be used with software services as you use them right now, but instead of clicking buttons here and there to tell the service what it should do, you just have a natural conversation with that said service, as you would with another human being. To some extent this is a new sort of user interface. This new UI needs to be intuitive of course and therefore relies on capabilities to understand the user.

Natural Language Processing

With this new approach of consuming and interacting with services in a more natural or human way, the ability of processing natural language is crucial. And this is one of the many fields in conversational AI which should involve not only developers and engineers but also domain matter experts. It doesn't make sense that you as a developer building a conversational AI app, like a bot, construct the language model for your app. The reason for that is that in most cases, you are not the domain matter expert your app should be developed for. Therefore, it makes sense to collaborate on those fields, like language models, with domain matter experts and the people who know the target audience of your bot. Those domain matter experts have in many cases a better understanding of the way users may interact with your bot and can therefore increase the quality of your cognitive skills within your bot. The key factors of such a natural language understanding model, which is a key component in every conversational AI app, are intents and utterances.

Intents are different areas within your language understanding model. Those intents can be used to indicate what the user's intention is when telling the bot something. They can be used as indicators or triggers within your business logic of an app, to define how the bot should respond to a given input. For example, if a user says, “What’s the weather today?” you certainly want the bot respond with something like “It will be mostly sunny with 28 degrees Celsius.” To define such an intent, like in the example “GetWeather,” you need to provide the natural language model with example phrases, which are called utterances. These utterances for the given intent could be

- What is the weather like?
- How is the weather?
- What’s the weather?
- Could you tell me the weather forecast?
- I would like to get some details on the weather please.

As you can see, these five example phrases differ in phrasing and syntax. And this is crucial for the quality level of your language model, as you need to empathize on the end users and how they will ask your bot about the weather. And the more utterances you will provide your model with, the more possibilities your model will cover.

But there is even more to that, as this basic example shows. Now with this scenario, user asks X and bot responds with Y. But you also need to somehow take the context of that conversation into consideration. Because if a user would ask a bot only the question about the weather, the bot does not know the location for which the user wants to know the weather. So, what you would need to add to your language model are so-called entities to indicate certain properties you need to know before you can process the request. In this scenario the location would be such an entity you would need to know before you can respond with details on the weather. To have this ability, you would need to enhance your language understanding model by adding the following utterances and tagging the entities, for example:

- What is the weather like in **{location=Seattle}**?
- How is the weather in **{location=New York}**?
- What’s the weather in **{location=Berlin}**?
- Could you tell me the weather forecast for **{location=Redmond}**?
- I would like to get some details on the weather in **{location=Amsterdam}** please.

The language understanding model is now able to detect location entities within given phrases and tag those locations for you. Within the business logic of your app, you can now use those entities, like variables, and process the request accordingly. Now it would be possible that you respond with the correct weather details if a user asks, "What's the weather today in Seattle?" And what you would need to handle within your business logic is the process if a user does not provide a location for querying the weather. In most cases it makes sense to respond with a question stating, "In order to give you the weather forecast, please tell me city you would like to know the weather for." Now this makes the interaction between the user and the bot more natural as there is some sort of context awareness.

But natural language processing is only one component within a conversational AI application. Imagine that you want to build a chatbot which extends your customer service scenario to provide users an additional contact which can be reached 24/7. Now in many cases, a chatbot could be a good way of handling questions or inquiries in a first step to relieve customer service employees. A chatbot can easily handle the basic questions, but what if a customer or end user has an urgent inquiry which cannot be handled by the bot? Analyzing the customer's messages to derive meaning from that can help you routing the messages to the right contact person in the back end. Therefore, it makes sense to use some sort of text analytics engine, which can detect the sentiment from messages. This way, you can decide within your chatbot's business logic if it is necessary to hand off to a human instead of continuing the conversation with the bot.

Language Translation

Let's extend the preceding use case with handling flight bookings, which can be handled by the bot. Now in many cases, the bot can help users with their bookings. But you would also need to take complaints into consideration. Even that can be managed by a bot; however, many people would like to get a clear response immediately if they want to issue a complaint. Using text analytics of some sort, you could detect the user's sentiment and decide based on the sentiment score, if it is necessary to immediately escalate the conversation to a customer service agent as the customer is angry or upset. This could prevent the case of unsatisfied customers which would lead to customer loss, as they would still be served accordingly based on their situation. Additionally, customer service agents would have more time to work on the "hard" cases and can serve customers more efficiently, as basic questions will be handled by the bot completely.

Furthermore, if you are building a bot which should deal with customer cases, depending on your offerings, it could be also beneficial to implement a multilingual bot. Now there are two possible implementation approaches:

- Build separate bots for each language.
- Build one bot and use translation to enable multilingualism.

Of course, it depends on the scenario and other circumstances like budget and timelines, but in many cases the second approach could be a good alternative to implement a multilingual bot. Using translation could enable your employees responsible for the bot's content and knowledge to only manage the content in one language instead of building knowledge bases or data silos for each language the bot should be using. The high-level message flow for such a multilingual bot could be as illustrated in Figure 1-2.

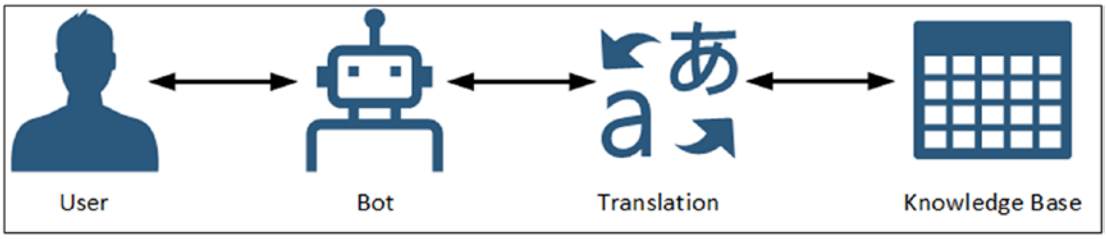


Figure 1-2. *Bot translation flow – high level*

Speech

Another angle which should not be neglected as well is the question if the bot should be a text-only bot or if it should also be voice-enabled. Voice is certainly a more sophisticated discipline when it comes to bot design. The reason for that is many people are used to writing with other people or bots without using accent phrases. But when people talk, they are used to talk with an accent instead of the grammatically correct way as if they would write something. Therefore, when you are building a bot which should be capable of having a conversation via speech, you would need to think of all the scenarios and accents people might use within their conversations. In addition, what you would need to figure it is the type of architecture you want to implement. For bots, especially in the Microsoft conversational AI platform, a solid scenario is to use

Azure Speech Services, which is a service targeting voice scenarios for implementing speech-to-text and text-to-speech. In such a case, you could use speech-to-text to gather all incoming voice messages and transform them into text, before redirecting those text messages to your bot's logic for executing the business logic. When the bot sends a response back to the user, you could use text-to-speech to translate that text-based message again into a voice-based message before sending it to the user's channel. This kind of architecture allows you to develop a bot which is capable of answering messages in a text-based scenario like chat as well as in a voice-based scenario like a phone call with the same business logic in the back end. The only difference from a development aspect would be to treat the user's channels differently to make sure that each call will use speech-to-text and text-to-speech before doing the natural language processing routines. With that in mind, a major advantage is that you can develop your bot once and then roll it out to text-based and voice-based channels at the same time without changing the code or needing to develop two different bots.

Usage Scenarios of Conversational AI

There are many different scenarios in which conversational AI can be useful. In fact, one could solve almost every business problem using AI of some sort. The major risk nowadays is to forget about the process of requirements engineering when starting to build conversational AI applications. Conversational AI is a subset of artificial intelligence, focusing on building seamless conversations between computers and humans. But as it is a big buzzword nowadays, the right use cases are often not evaluated thoroughly. Many companies and vendors try to implement AI services because it is a trend these days. But this often results in a situation where users do not benefit from the new solutions powered by AI. The key point which makes an AI-powered use case, especially one built using conversational AI, successful is to make the user experience significantly better or faster and not worse.

One simple example could be an intranet scenario in a large enterprise. Typically, those intranet solutions are designed to store a lot of information like documents in it. The problem with that is that in many cases, those solutions grow over many years and users are having trouble to find relevant information in that "information jungle," which is especially true for new employees using the intranet for the first time. If you imagine now which steps a user typically would need to do to find the right information, this process could look like the following:

1. Log in to the intranet solution.
2. Navigate to the correct area within the intranet where the information might be stored.
3. Navigate to the document store within that area.
4. Look through the documents to find the correct document.
5. Open the document.

Now imagine the same use case but served through a conversational AI app in the form of a chatbot. The users would typically need to do the following to achieve the same result:

1. Log in to the channel where the bot can be accessed.
2. Send a message like: "Could you please send me the document xyz" or "Could you please send me all documents with the metadata xyz."
3. Open the documents sent by the chatbot.

Now if you compare those two approaches, the user would be much faster to just tell the chatbot which information he wants to get, without the necessity of searching through the files himself. This in turn has the potential to save the users a lot of time, which they can invest in innovative tasks or tasks which they like to do as they are more enjoyable other than searching through documents. Of course, this depends on an architecture, in which the bot is integrated in the intranet solution in that case, being able to index all the documents, to send them out to users in a matter of seconds. But the key point here is that the bot or, in general, the applied AI solutions generate value for the end users instead of making the process worse.

Another example would be in the field of business reporting. There are a lot of solutions available in the market which offer services for different reporting purposes. The big problem with many of those is the complexity for the end users in most cases. As a reporting tool should be able to deal with complex data structures to build sophisticated reports, the user interface to build those reports is also quite complex. It would be far easier if you could tell the service which KPIs you want to see using natural language. Therefore, many vendors, including Microsoft, are building services into their reporting solutions, which offer the possibility to query the data using language instead of the complex reporting interface, which they are used to. This way, the end users potentially save a lot of time, as they only need to tell the system what they want to see which usually is a matter of seconds, instead of spending a lot of time in building dashboards upfront. Therefore, conversational AI can be seen as some sort of new user interface, designed to expose software services through a conversational interface, which should be relatively easy to handle as humans are used to speak and interact using natural language.

Another key point of conversational AI is that, by introducing some sort of conversational AI in organizations, those services potentially could take away people's fear of artificial intelligence. In many parts of the world, especially in manufacturing companies, people have some sort of fear that AI services, or generally spoken machines, will take over their job sooner or later. Therefore, they refuse to use any of those "modern" solutions as they think that the machine will take their job and they would face serious problems. By introducing "lightweight" AI services, in the form of chatbots and other conversational AI solutions, companies have the chance to tell people that these will be acting as virtual assistants for their employees, designed to help and assist people. Those virtual assistants should be designed to handle repetitive tasks of users, so that they have the chance to spend more time on innovative or harder tasks, which virtual assistants could not accomplish. By following such an approach, adoption rates of conversational AI services in organizations are typically much higher, if people see those services as assistants designed to take over work, which they do not like to do.

One good example for such a scenario would be in IT support. Every company has some sort of helpdesk team, which basically deals with a variety of different questions and problems. One of the most common problems is the question on the password reset. Imagine that you would work in an IT helpdesk and get the question how to reset a password several times a day. At some point, this could be an annoying question, but still you would need to help users struggling with an expired or forgotten password. But if you would introduce an IT helpdesk chatbot, which would be able to deal with those basic questions like how to reset a user's password, you would have more time to work on the more sophisticated problems, or even work on the process optimization within the support unit. And as the chatbot will hardly ever tell you that he is annoyed of answering the same questions over and over again, the users will be served properly, if not more efficient, as the chatbot can immediately deliver the answer to the user having the problem, as he is always available, even outside of business hours.

Service Offerings Around Conversational AI in Microsoft Azure

As Microsoft's public cloud platform, Azure offers many different services and solutions for the different areas like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS); it also offers many services in the field of AI as a Service (AIaaS). The service range is quite broad, covering many different service types used in a conversational AI application, which can be seen in Figure 1-3.

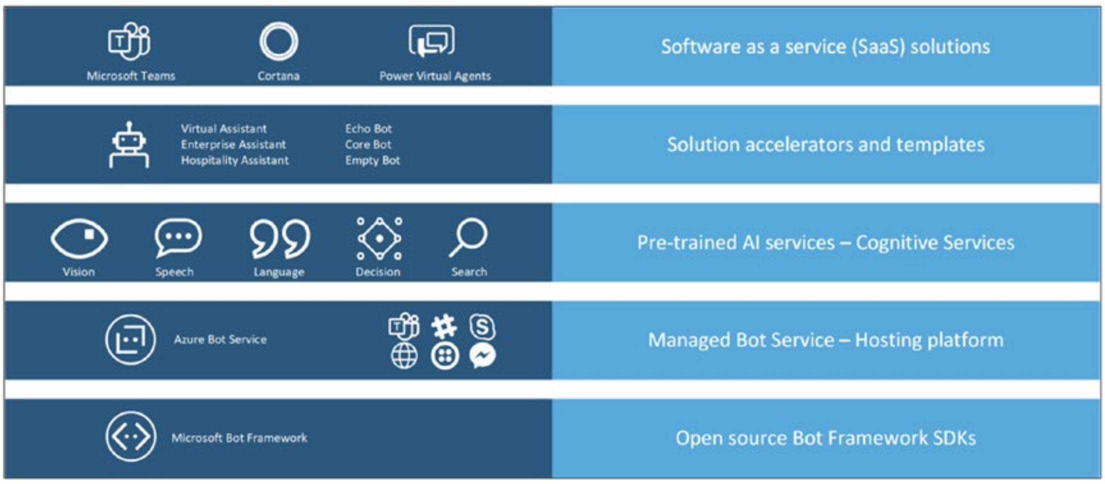


Figure 1-3. *Microsoft conversational AI platform*

Bot Framework and Azure Bot Service

The foundation of every conversational AI application or chatbot is the Microsoft Bot Framework. This is an open-source SDK, which is by now available in C#, JavaScript, and Python (all generally available) as well as Java (Preview). This SDK offers an open, modular, and extensible architecture for building bots and conversational AI apps. It provides a built-in dialog system which is agile and can be customized and adapted. One of the main advantages of this SDK is the fact that it shares the same implementation across all the development languages. This means that developers can build bots using .NET and JavaScript and do not need to learn two different implementation approaches or concepts as they are unified across all programming languages. Moreover, the Microsoft Bot Framework SDK integrates seamlessly into the Microsoft Azure platform as well as all the Microsoft Cognitive Services.

When deploying a bot built using the Microsoft Bot Framework, the Azure Bot Service acts as the hosting platform in Azure. This service basically accelerates the deployment and management cycles of a Bot Framework bot. The Azure Bot Service (ABS) allows us to connect a bot with multiple channels in a matter of minutes, as Microsoft is managing the integration between a bot and the connected channels for us. So you do not need to spend days setting up a new connection between your Azure environment and Facebook, for example, just to activate your bot in the Facebook Messenger – this is done by Microsoft. All you need to care about is that your bot's logic and code are supported by the channel's platform. And a big upside of this is the fact that Microsoft is constantly updating the list of supported channels. By the time of writing, the following channels are supported (with subject to change):

- Cortana
- Office 365 email
- Microsoft Teams
- Skype
- Slack
- Twilio (SMS)
- Facebook Messenger
- Kik Messenger
- GroupMe
- Facebook for Workplace
- LINE
- Telegram
- Web Chat
- Direct Line
- Direct Line Speech

As the preceding list shows, Microsoft is also integrating third-party platforms like Facebook Messenger or Slack. This extends the possible chatbot use cases as the bot can be used in Slack and Microsoft Teams simultaneously sharing the same code base.

Of course, you would need to make sure that the bot's code is supported in both platforms, especially when dealing with front-end matters, because, for example, Microsoft Teams is supporting a concept called Adaptive Cards, which is explained in a later chapter. This solution offers the possibility to render rich attachments like images, audio files, buttons, and texts in one single message. The upside of this is that the definition of such an Adaptive Card is done using JSON and can therefore be replaced in runtime. The downside is that many channels do not yet support this new concept, like Slack or Facebook Messenger. So, if you build a bot for Teams and Slack, you will need to treat both channels differently in different areas within your conversation. But the overall goal is to write a bot once and deploy it in multiple channels. This way you can minimize your development and deployment cycles and maximize the number of possible users and the reach. The details of the Bot Framework and all its concepts and patterns are explained in a later chapter.

Cognitive Services

Microsoft offers a set of prebuilt machine learning services called "Cognitive Services" which should help developers, without data scientist skills, to build intelligent applications. This set of comprehensive APIs is designed to deliver the ability to develop an application that can decide, understand, speak, hear, and search without a huge amount of effort. The following list summarizes all Cognitive Services which are currently available:

- **Decision**
    - **Anomaly Detector (preview)**
        - A service helping users to foresee any kind of problem before it occurs.
    - **Content Moderator**
        - An API which can be used to moderate content using machine assistance as well as a human review tool for images, text, and videos.
    - **Personalizer**
        - A service based on reinforcement learning to provide personalized and tailored content for end users.

- **Language**
    - **Immersive Reader (preview)**
        - A service which is currently in preview providing the ability to embed text reading and comprehension functionality into applications.
    - **Language Understanding**
        - Integrate natural language understanding functionality based on machine learning models into applications as well as bots to understand end users and derive actions based on their needs.
    - **QnA Maker**
        - A cloud-based service giving developers and business users to build knowledge bases based on frequently asked questions to be used in a conversational manner.
    - **Text Analytics**
        - Text Analytics empowers users to integrate sentiment and language detection, entity, and key phrase extraction from an input text into apps and bots.
    - **Translator**
        - A neural machine translation API for developers offering easy-to-use interfaces to conduct real-time text translation supporting more than 60 languages.
- **Speech**
    - **Speech to Text (part of the speech service)**
        - A real-time speech-to-text API giving users the ability to convert spoken audio into text including the ability to use custom vocabularies to overcome speech recognition barriers.

  - **Text to Speech (part of the speech service)**
    - Using neural text-to-speech capabilities, this API offers the ability to convert text-to-speech for natural conversational interfaces supporting a broad range of languages, voices, and accents.
  - **Speech Translation (part of the speech service)**
    - A cloud-based automatic translation API offering speech translations in real time.
  - **Speaker Recognition (preview)**
    - Speaker Recognition is a service providing the functionality to recognize and identify individual speakers to either determine the identity of an unknown speaker or use speech as a verification method.
- **Vision**
  - **Computer Vision**
    - An API which can analyze the content in images as well as extract text from images and recognize familiar objects in images like brands or landmarks.
  - **Custom Vision**
    - Based on the Computer Vision service, it enables Custom Vision users to create their own computer vision models based on their requirements.
  - **Face**
    - The Face API is tailored to analyze faces in images to be used as facial recognition service which can be integrated in any kind of application.
  - **Form Recognizer (preview)**
    - This API is an AI-powered document extraction solution used to automate the extraction of information like text, key-value pairs, or tables from documents.

- **Ink Recognizer (preview)**
    - Ink Recognizer allows users to recognize ink content like shapes, handwriting, or the layout of inked documents targeting various scenarios like note-taking or document annotation.
- **Video Indexer**
    - This API can extract metadata from audio and video content automatically providing valuable insights like spoken words, faces, speakers, or even complete scenes.

- **Web search**
    - **Bing Autosuggest**
        - Bing Autosuggest offers intelligent type-ahead capabilities which can help end users to complete queries much faster and more efficiently.
    - **Bing Custom Search**
        - This service can be used to create a customized search engine which allows you to offer ad-free, commercial-grade results based on your scenarios.
    - **Bing Entity Search**
        - An API which offers the ability to recognize and classify named entities to search and find comprehensive results based on those entities.
    - **Bing Image Search**
        - Targeting developers seeking to integrate image search capabilities into apps, this API allows you to add image search options to apps and websites.
    - **Bing News Search**
        - Bing News Search empowers developers to add a customizable news search engine to websites or apps offering the ability to filter by topic, local news, or metadata.

- **Bing Spell Check**
    - An API which can be used to assist end users in identifying and fixing spelling mistakes in real time.
- **Bing Video Search**
    - Like the Bing Image Search, this API allows developers to add a range of advanced video search capabilities, like video previews, trending videos, or videos based on specific metadata to apps.
- **Bing Visual Search**
    - This service allows end users to search for content using images instead of regular text-based search queries.
- **Bing Web Search**
    - With Bing Web Search, developers can build solutions to retrieve documents, web pages, videos, images, or news stored on the Internet with a single API call.

As you can see, the list of Cognitive Services is quite long, and there are many different services which can be used in a variety of use cases. As this book is about the conversational aspect of artificial intelligence, we certainly will not cover all those Cognitive Services in detail. But some of those APIs will be explained more precisely in a later chapter, as many of them play an important role to build a conversational AI application.

Solution Accelerators and Templates

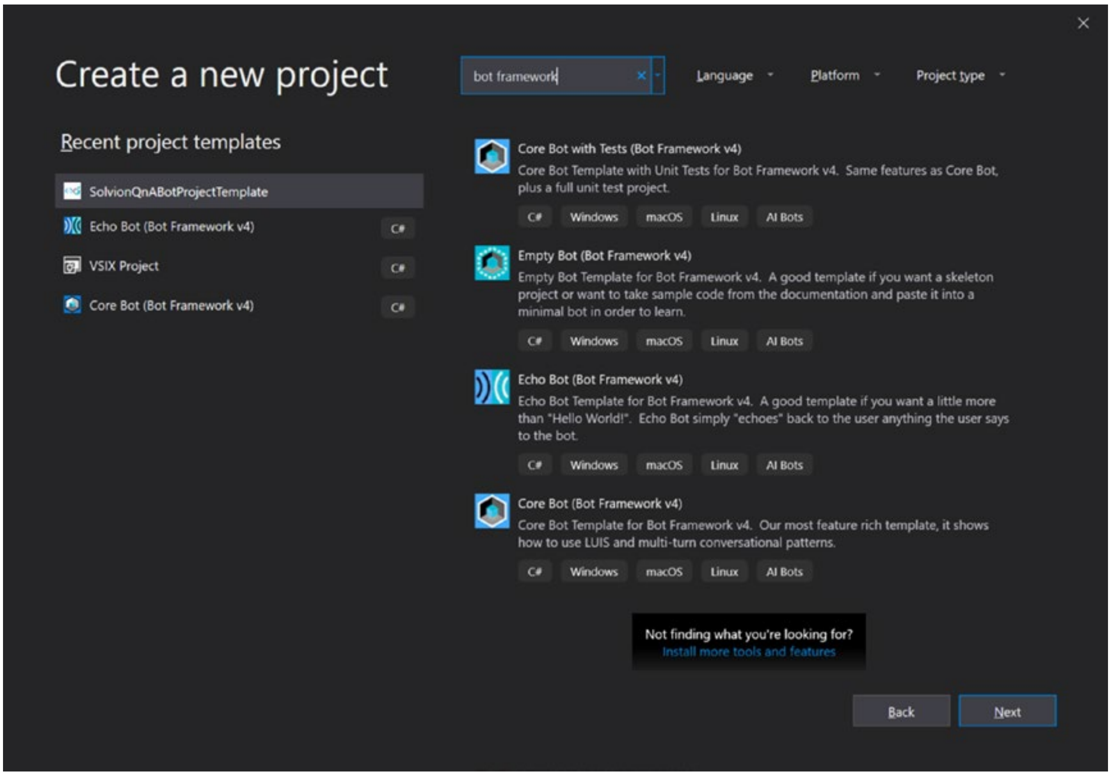
When starting building bots using the Microsoft conversational AI platform, it is probably a good approach to use some kind of template instead of building everything from scratch. Microsoft did a good job in offering a variety of templates for developers and chatbot architects which can be used as a starting point for building a new project. Currently, the following templates are available for .NET, JavaScript/TypeScript, and Python development which are described in Table 1-1.

Table 1-1. *Bot Framework templates (source: https://github.com/microsoft/BotBuilder-Samples/tree/master/generators)*

Template	Description
Echo bot	A good template if you want a little more than "Hello World!" but not much more. This template handles the very basics of sending messages to a bot, and having the bot process the messages by repeating them back to the user. This template produces a bot that simply "echoes" back to the user anything the user says to the bot.
Core bot	The most advanced template, the core bot template provides six core features every bot is likely to have. This template covers the core features of a conversational AI bot using LUIS.
Empty bot	A good template if you are familiar with Bot Framework v4, and simply want a basic skeleton project. Also, it is a good option if you want to take sample code from the documentation and paste it into a minimal bot in order to learn.

Note All the preceding templates can be used in a variety of scenarios. If you are a .NET developer, you can use the Visual Studio extension from `https://github.com/Microsoft/BotBuilder-Samples/tree/master/generators/vsix-vs-win/BotBuilderVSIX-V4` which installs the solution templates to your Visual Studio IDE.

With the extension installed, you can simply start a new Visual Studio Bot Framework project as with any other template, which is shown in Figure 1-4.


Create a new project

bot framework | Language | Platform | Project type

Recent project templates

- SolvionQnABotProjectTemplate
- Echo Bot (Bot Framework v4) C#
- VSIX Project C#
- Core Bot (Bot Framework v4) C#

Core Bot with Tests (Bot Framework v4)
Core Bot Template with Unit Tests for Bot Framework v4. Same features as Core Bot, plus a full unit test project.
C# Windows macOS Linux AI Bots

Empty Bot (Bot Framework v4)
Empty Bot Template for Bot Framework v4. A good template if you want a skeleton project or want to take sample code from the documentation and paste it into a minimal bot in order to learn.
C# Windows macOS Linux AI Bots

Echo Bot (Bot Framework v4)
Echo Bot Template for Bot Framework v4. A good template if you want a little more than "Hello World!". Echo Bot simply "echoes" back to the user anything the user says to the bot.
C# Windows macOS Linux AI Bots

Core Bot (Bot Framework v4)
Core Bot Template for Bot Framework v4. Our most feature rich template, it shows how to use LUIS and multi-turn conversational patterns.
C# Windows macOS Linux AI Bots

Not finding what you're looking for?
Install more tools and features

Back | Next

Figure 1-4. *Bot Framework Visual Studio templates*

If you want to build bots using .NET core without using Visual Studio but any other web development IDE, you can utilize the .NET Core templates.

Note The .Net Core Visual Studio Bot Framework templates can be installed from here: `https://github.com/microsoft/BotBuilder-Samples/tree/master/generators/dotnet-templates`.

These templates allow you to scaffold your bot project completely via the command line with the `dotnet new` command like this:

```
dotnet new echobot -n MyEchoBot
```

But if you are a web developer who is comfortable in developing solutions in JavaScript or TypeScript using Visual Studio Code, for example, you can use the Yeoman generator for Bot Framework v4 from here `https://github.com/microsoft/BotBuilder-Samples/tree/master/generators/generator-botbuilder` to scaffold