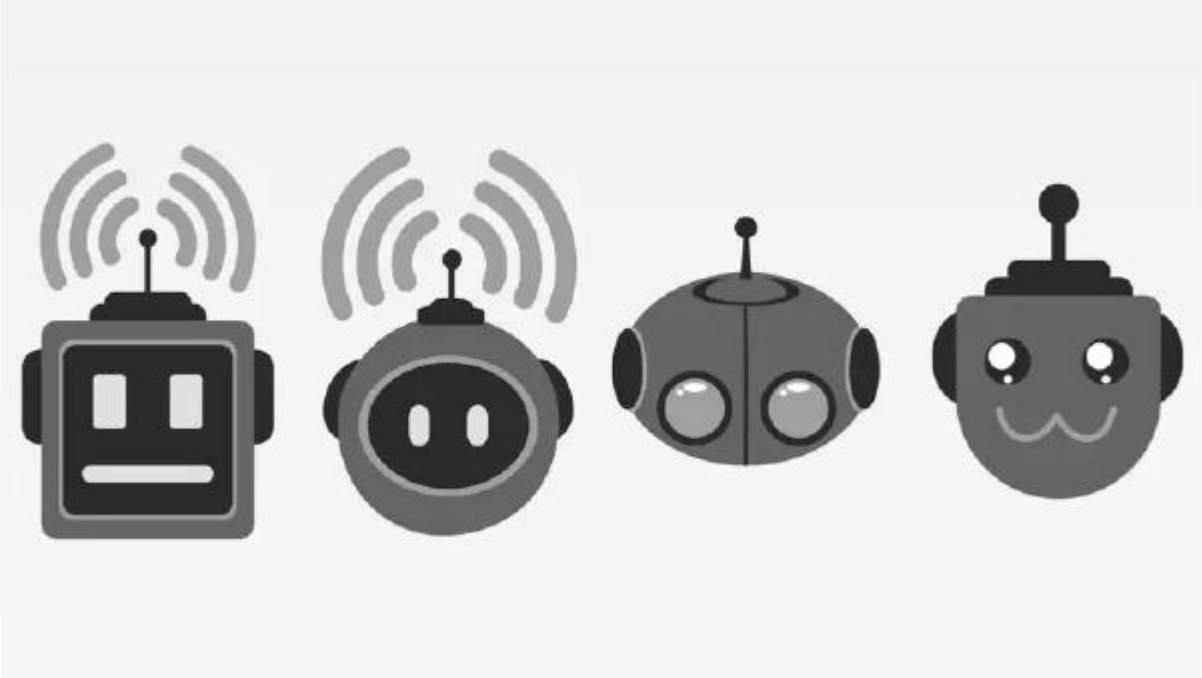


译者 | 大愚若智

编辑 | Natalie

AI 前线导读：本文概括介绍了聊天机器人的发展和应用趋势，并从开发框架、平台、后端 AI 服务等方面为希望着手此类开发的开发者提供了非常全面的建议和思路。

更多优质内容请关注微信公众号“AI 前线”，(ID：ai-front)

聊天机器人：目前发展得如何？

人工智能正在崛起！虽然机器还不至于在遥远的未来反抗自己的创造者——人类，但作为一个飞速发展的趋势，信息技术领域对基于机器的预测和决策等应用已经越来越普遍。围绕 AI 的炒作无处不在：无人驾驶汽车、智能图像处理（例如 Prisma），还有通信领域的各种新颖应用，例如对话式 AI，即聊天机器人（Chatbot）。

聊天机器人这个行业正在飞速扩展，但相关技术依然还很年幼。对话式机器人的应用目前依然比较虚幻，有些类似于文字模式的老派游戏“I smell a Wumpus”，但目前该技术已经逐渐发展成为一个重要的商务工具。聊天机器人提供了一种更加简单友好的全新接口，非常适合用于浏览信息和接收服务。IT 专家以及包括 Google、微软、Facebook 在内的行业巨头均认为该技术会在未来扮演重要的角色。

为了发挥对话式人工智能工具（或者如果你嫌名字太长，也可将其称之为“聊天机器人”）的巨大潜力，我们必须首先掌握一些基础信息并理解其相应的技术栈。本文将探讨可用于从事相关开发的各类技术，各种技术间的相似与不同之处，以及各自的优劣。

但在开始进一步介绍之前，首先来深入了解一下聊天机器人以及它们的技术拓扑。

为何、如何以及何时使用聊天机器人？

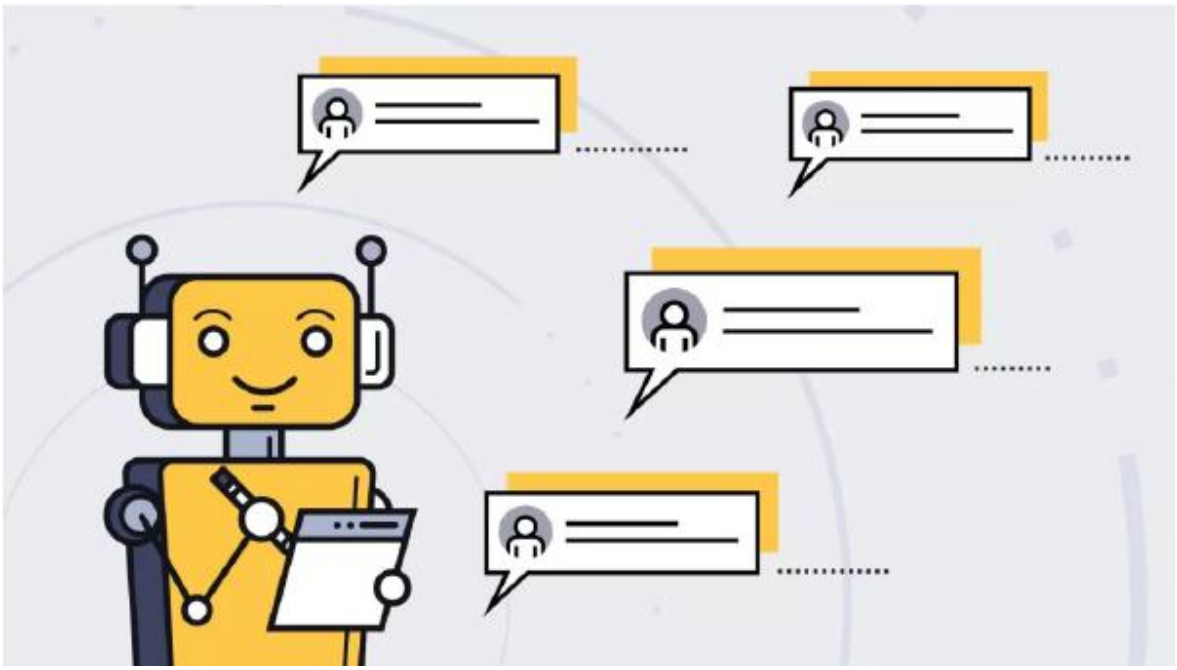
面对不同的业务任务，目前已经有各种各样的聊天机器人，从广告宣传到团队构建不一而足，但这些机器人通常都有一些共通的核心功能。

◆ 个人助理机器人

业务事务通常离不开专业的组织性协助，但不是所有人都有足够的预算请秘书来处理各种基本任务。好在聊天机器人可以提供帮助。通过妥善的编程，它们可以追踪我们的工作安排计划，针对即将进行的活动发出提醒。此类机器人比较实用，因为基于功能来看非常简单，并且可以使用各种非常快速的通信平台，例如用消息系统作为自己的接口。

◆ 客户支持机器人

聊天机器人还有可能从事更复杂的任务，例如代表企业与真实客户建立联系。哪怕对人类员工来说，客户支持工作流程大部分都是可预测并且可以实现脚本化的，因此很容易以机器人的方式实现。此时可通过典型的机器人行为算法接受用户查询，解析并获得相关信息，在数据库中找出类似案例，然后通过预设答案做出回应。



◆ 团队协作机器人

通过机器人为开发者团队提供支持，这种做法目前极为流行，而原因有很多。这种机器人可与开发环境无缝融合，借此持续关注并审视软件工程师有关软件质量

等目标的不同需求。然而这类机器人只能解决各种非常具体的任务，因此这种情况下并不需要用到极为复杂的商用机器人。这类机器人通常以某种非常简单的“记分员”为主，此外还包括有感知的聊天机器人，它们可以为开发服务器提供必要的支撑，并针对信息的提交、简单的工作安排创建报表。

◆ 出版商 / 新闻机器人

出版商类型的机器人可以每天不断收集各种感兴趣的信息。很多大型新闻机构（WSJ、NYT）以及技术类媒体（TechCrunch、MIT Technology Review）都会通过诸如 Facebook Messenger 等重要平台，以简要文字信息这种便利的方式分享内容。这种机器人背后的原理很简单：从用户处收集订阅信息，安排相关新闻的交付操作，然后处理与用户有关的其他请求（例如退订、更改所订阅的话题、随意浏览等）。

◆ 娱乐机器人

娱乐机器人目前不怎么常见，并且用途较为特殊：例如通过对话风格的工作流程管理赛事 / 电影 / 戏剧门票的预订工作。一些机器人还可以通过文字消息为娱乐网站提供全面的沉浸式体验。例如 Fandango Facebook Bot 可供用户观看新电影的预告片，阅读影评，查找附近的电影院。很棒吧！

◆ 旅游机器人

另外还有一种非常流行，并且应用范围正在飞速增长的机器人：旅游辅助机器人。通过使用这种面向客户的聊天机器人，可以帮助用户完成一些原本很繁琐的任务，例如选择最佳出行方式，借此将一些原本非常冗繁的流程转变为聊天应用中轻松随意的对话。旅游机器人不仅可以获取并确认预订信息，而且可以针对重要时间向用户发出通知，例如开始办票了、开始登机了、航班状态有变化了，此外还能从客户处收集各种有价值的反馈。

聊天机器人已成全新用户接口：机器人的最新愿景

虽然无论怎么说，聊天机器人也只是程序，只不过按照设计可以通过传统的对话方式，用文字形式（聊天平台）与人类用户进行交流。它会等待用户说些什么，然后按照程序安排作答。因此目前的聊天机器人，其本质都是一种很简单的算法：接受并理解输入的内容，提供相关回应作为输出结果。

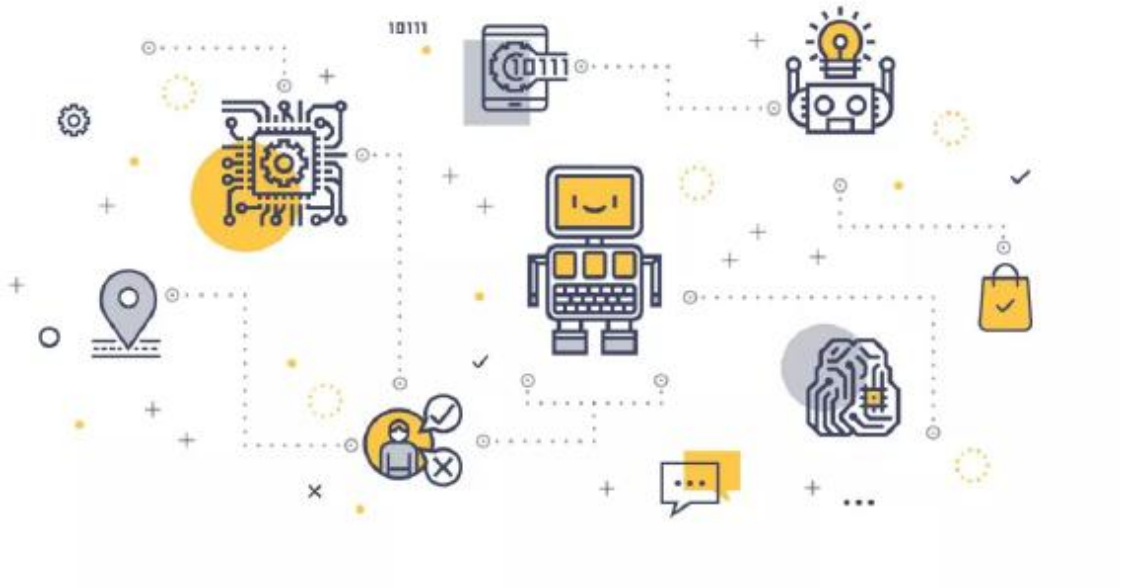
然而实际上聊天机器人远比上文说的复杂，因为它们可以掌控上下文情境的力量，这个情境可能是局部的（例如只适用于一次对话）或者全局的（例如可在多次对话中续存并能涵盖言境之外的领域，如预订披萨的机器，可以处理用户的最新订

单位置、时区等信息）。虽然前一种做法通常可以使用 Cookie、会话等临时记忆实现，但后者需要将信息存储在数据库内，或通过 API 的方式访问其他内部服务获取。

在介绍了上下文情境的概念后，还需要简要提及 Web 应用程序相关的一些术语（例如 Cookie、会话、数据库），这样才能组成一个完整的聊天机器人。聊天机器人需要与 Web 应用共享很多信息，而 Web 应用主要负责提供在线浏览的网页（同样需要接收请求并做出响应，所以会共用数据库等标准工具）。因此严格来说，聊天机器人也是一种 Web 应用。

所以说，对于各类信息和服务来说，聊天机器人实际上也是一种新类型的接口。这种接口很紧凑，可以轻松地访问，并且非常简单。借此你的服务也可以从原本的居所（你的网站）进一步扩展至各种平台，无需向用户宣传或营销即可随时访问（以前的机制为：用户在聊天信息中看到广告，点击链接访问广告网页，然后订购产品；现在的机制变成了：用户在聊天信息中看到广告，直接通过对话订购产品）。

聊天机器人如何理解各种任务？



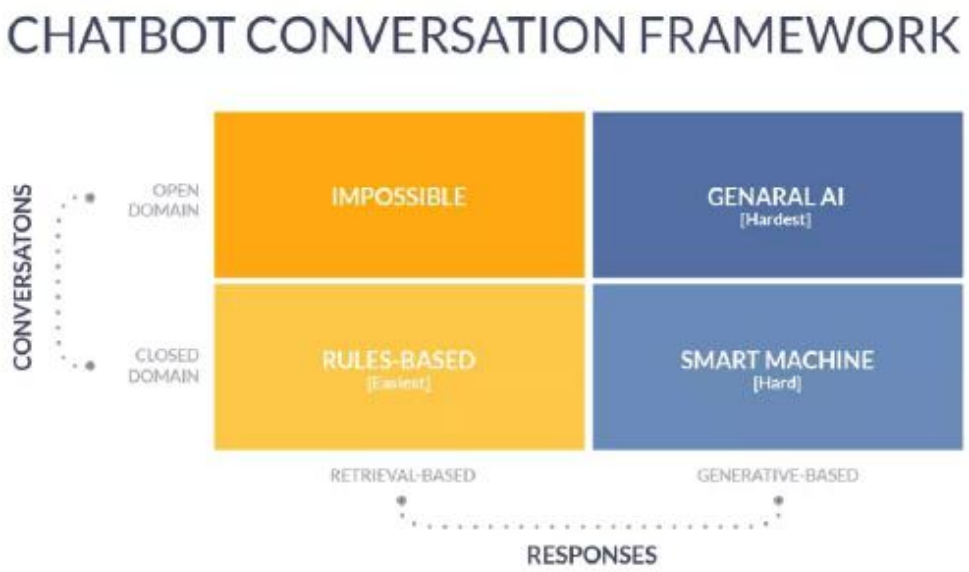
聊天机器人的内部原理看似简单，但实际上并不那么容易实现。机器人首先必须理解用户说了什么。实现方法有很多：可以对用户输入内容进行模式匹配，或使用自然语言处理（NLP）技术对用户意图进行分类。前者非常简单直观，但随着所输入内容的范围和规模逐渐扩大，维护难度会大幅增加。后者可通过机器学习技术解读所输入的内容，但是实现难度较大（至少在不使用已经应用相关技术的平台前提下会很难）。为了对各种可能的意图进行分类，并从各种可能中确定特定输入的实际目的，必须准备一系列样本。

好在有很多平台已经实现了这样的逻辑，我们不需要再考虑这些问题，可以直接使用它们提供的服务。然而我们依然需要熟悉主要的 NLP 分类及其本质：

- **实体（Entity）** 是指人类谈话（口头或书面）中，自然语言所包含的文字组合，与标准化解析后所代表的用户隐含意图之间的一种特殊映射关系。这有些类似于提取的变量，例如 DateTime 如果指定为“圣诞节”，那么可能就意味着“2017 年 12 月 25 日”。
- **意图（Intent）**，与实体截然不同，是一种将用户的消息映射为相关机器人操作（预测工作流）的常规特征。例如，“今天天气如何”这句话可按照全文直接映射至“weather_inquery”意图，而非映射至其他内容。
- **操作（Action）** 是指机器人可以执行并作为相关意图响应结果的步骤。通常这可能是传统的函数，并可能通过可选参数从调用方获得更详细的信息（上下文情境）。

取决于具体平台，上下文情境可能多种多样，在具体形式或拓扑方面并没有什么严格的规定。但情境主要体现为键 / 值映射的形式，可以持续追踪实体的最新含义，并区分不同短语所包含的含义 / 意图之间的差异。

机器人的不同类型：开放 / 闭合域，基于生成 / 基于检索



虽然已经说过，聊天机器人实际上是由某种类型的 Web 接口组成，但还要注意，这实际上是一种人工智能应用，因此自然会涉及到分类学方法。在介绍过机器人用来理解语言的方法后，一起来看看忽略具体类型和响应方式的情况下，各种机器人的拓扑。

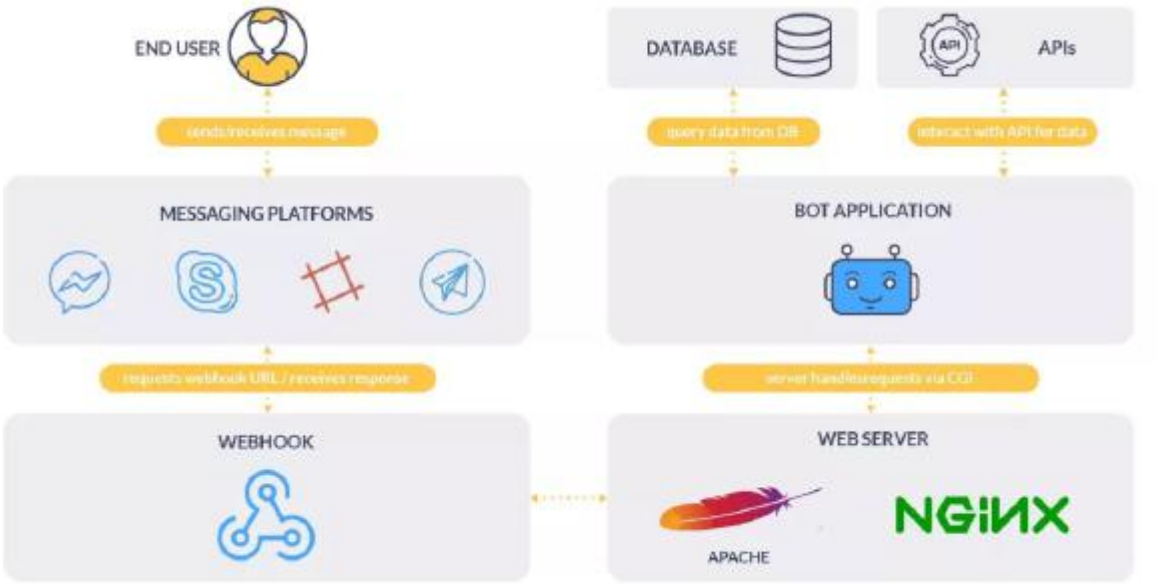
首先可以根据操作范围（是否严格限定于某一闭合域，如天气机器人或披萨机

器人，或进行各种类型的通用对话），以及为了响应用户输入所要进行的计算方式（是否会直接检索预定义的响应内容，还是根据输入的信息生成对应的响应结果），将对话式 AI 划分为不同类型。

无论何种方式，对于检索形式的响应，最重要的一点在于要严格区分静态和动态响应。前者是最简单的，有些类似于直接套用现成的模板，输入的每个信息都有对应的答案。后者则更像一种知识库，可以根据相关性得分返回一系列可能的响应结果。

对于适用于某种闭合域的聊天机器人，只需要解决围绕有限的几个问题所展开的交流即可，例如预订酒店 / 餐厅 / 航班，购买披萨，买鞋等。很明显，此时输入的内容也是有限的，对于一个负责定够披萨的机器人，完全无需考虑用户可能谈到的政治、心理学或哲学问题。

开放领域的机器人主要侧重于与用户自身的对话，然而并不需要完全理解用户说的每个字，无需检索实体和意图，也不需要追踪上下文情境。这种机器人只需要努力模拟现实生活中的对话，其主要目的在于娱乐用户，或解答类似 FAQ 那样的通用问题。



如何构建一个聊天机器人？

了解了这些基础知识后，可以开始考虑构建一个了。不过开始之前还需要决定自己打算使用的平台或工具。

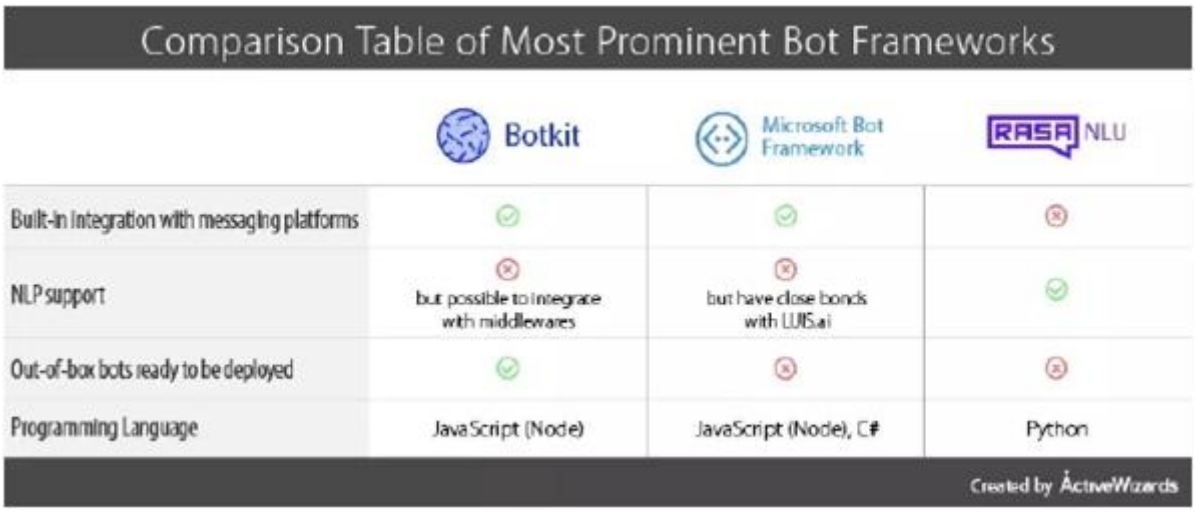
如果希望快速上手并尝试，但还没准备好由自己来写代码，建议先从**无需编程的聊天机器人生成器**着手。这类应用主要面向非技术型用户，非常简单易用。此时无需过多考虑技术细节，只需要完成一些纯粹的概念性工作即可（不过依然有一定的学习曲线）。这种方式最适合用于构建简单的机器人，但并不适合复杂的商业用途，

因为这类方式最大的不足在于缺乏，甚至完全不具备自然语言处理能力（这种能力是复杂的机器人所必须的）。类似的平台有很多，例如 Chatfuel、ManyChat、Octane AI、Massively AI。

如果希望进行更复杂的开发，则需要用到机器人框架和 AI 服务。

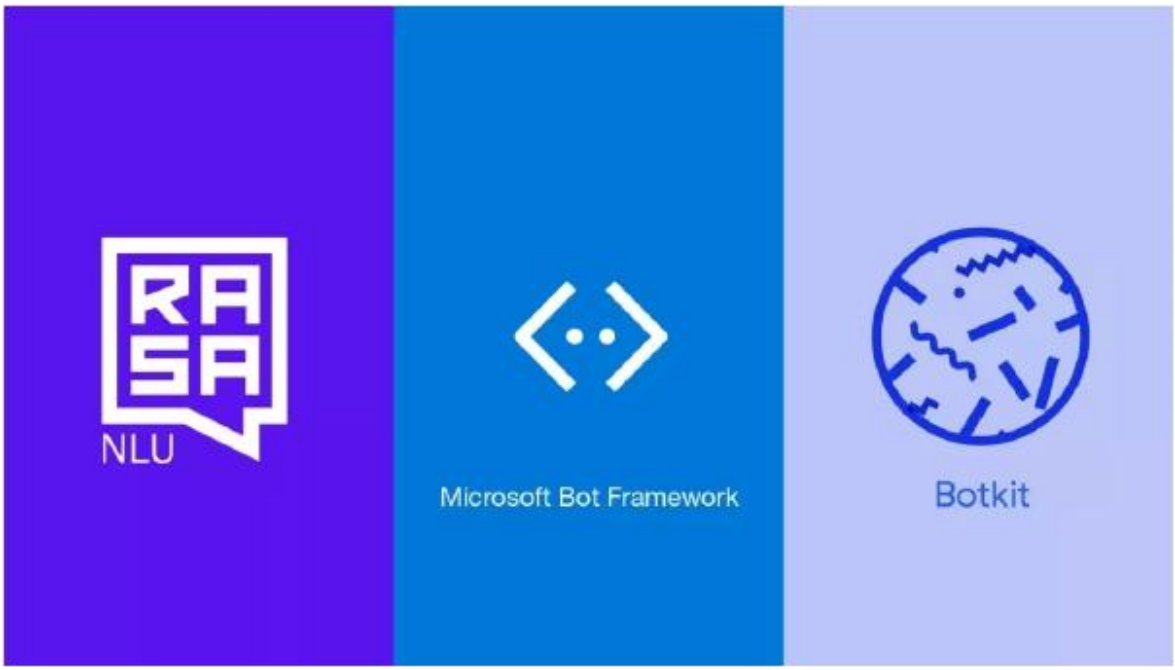
- **框架** 包含了聊天机器人工作流程内各种通用功能的抽象，为了便于使用通常会打包提供。聊天机器人框架与其他软件框架（例如 Web 应用框架）类似，可以为我们提供各种必须的工具。通常这类框架都是面向某种特定编程语言实现的。此外一些机器人框架还提供了托管式的，甚至可交互的开发环境，借此进一步简化机器人的创建过程。
- **AI 服务** 则是另一种独立的云端平台，通常会提供以交互式方法创建聊天机器人逻辑所需的 GUI，并具备由机器学习技术驱动的自然语言处理功能，可通过 RESTful API 通信。

另外还有一种比较新颖的方法，可以将聊天机器人与交互式语音响应（IVR）系统直接集成到 Web 服务构建工具或框架中。例如这一领域最新的成果之一是由 Aspect 开发的 Customer Experience Platform（CXP）解决方案，该解决方案可用于构建 Web 服务。借此我们可以轻松地创建由数据支撑的网站，并通过基于文本和语音的机器人接口提供给用户使用。有关这种方法的详细信息可参阅 这里。
主要的机器人框架：Botkit、Microsoft Bot Framework、Rasa NLU



正如上文所述，框架其实是一种面向特定编程语言的库。下文将介绍三个主要的框架：适用于 Node.js 的 Botkit for node.js，适用于.NET（也适用于 Node.js，但下文将主要围绕.NET，尤其是 C#）的 Microsoft Bot Framework 及 Bot

BuilderSDK，以及适用于 Python 的 Rasa NLU。



◆ Botkit

首先从 Botkit 开始。按照设计，该框架可以帮助忙碌的用户快速简单地针对自己的需求构建机器人，同时无需过多深入具体的技术细节。该框架可通过一个统一的接口发送和接收消息。这个框架最初主要用于 Slack，但现在其功能通过扩展已经可以支持连接至多种消息平台。该框架提供了直观的工作流，并以简明扼要的方式加以整理，具备翔实的文档，同时还提供了大量线上聊天机器人范例，借此用户可以快速上手，并进一步开发自己的机器人。

然而要注意，该框架不包含自然语言处理功能，不过可以通过中间件连接至现有的或自行开发的服务中实现自然语言处理。

该框架库包含大量核心功能和连接器，为多种平台提供了支持。核心功能是以事件监听器的形式实现的，如.on_message_receive、.hears 等。连接器的实现则取决于所连接的平台。

Botkit 的使用非常简单。首先需要安装 Botkit，这一工作可通过 npm 安装实现：`npm install—save botkit`。随后创建一个包含代码的文件（可以参阅这里【https://github.com/howdyai/botkit/blob/master/console_bot.js 】查看最基础的控制台机器人范例），并使用 node.js 运行：`node my_bot.js`。

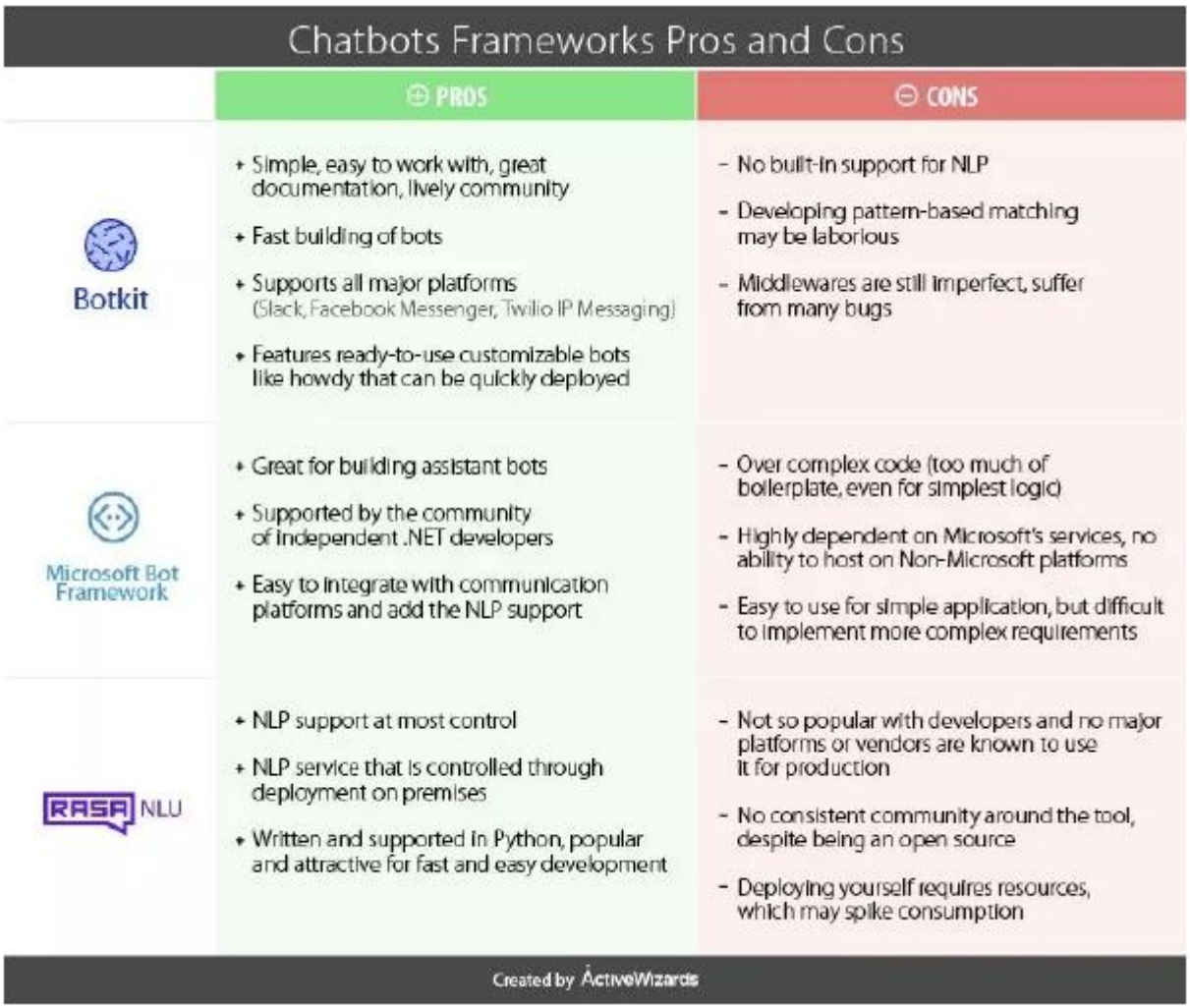
◆ Microsoft Bot Framework

微软十分热衷于紧跟技术发展趋势，为我们提供了一种开发聊天机器人的全新方式。

Microsoft Bot Framework 功能极为全面，并提供了简单易用的构建 SDK，其中包含两个主要组件：用于实现集成的框架 Bot Connector SDK 本身，以及最基本的机器人逻辑和 LUIS.ai，后者主要用于实现自然语言处理，让机器人感觉上更像人类。虽然 LUIS.ai 更像是一种功能而非组件，但该框架本身也可不借助该功能直接使用，并且单独使用就已经可以实现让人印象深刻的效果。这套工具非常健壮，在集成方面也可以提供近乎无限的潜力（例如与 Messenger、Skype 等集成）。该开发环境为交互式开发和简单的测试提供了必要的工具，用户还可将自己开发的机器人公布出来并接受微软的审核，如果审核通过还可纳入 BotDirectory（https://bots.botframework.com/ ），并被更多人使用。

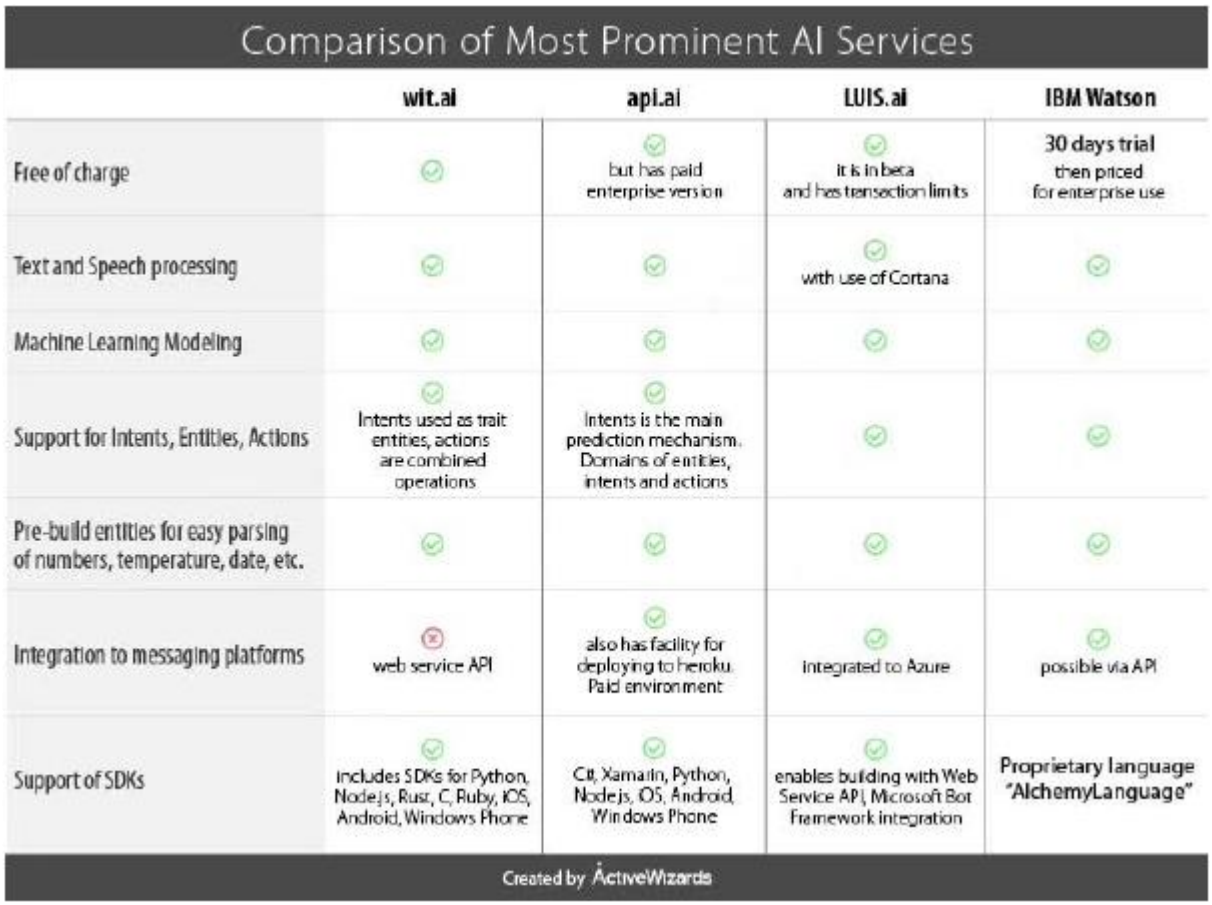
- Rasa NLU

虽然并非专门为聊天机器人的构建量身打造的框架，但 Rasa NLU 也是一种很方便的后端解决方案。Botkit 和 Microsoft Bot 可连接至各种消息平台，而 Rasa NLU 更像是一种自然语言处理服务，可以在用户本地环境提供处理功能。此外 Rasa 提供了 Python 接口，可将实体提取器直接集成于使用 Python 开发的应用程序。此外该技术还可作为服务在其他框架中运行，并暴露 REST API 端点。这些工具各自的利弊，可参阅下表。



卓越的 AI 服务：

Wit.ai、api.ai、LUIS.ai、IBM Watson 正如上文所述，AI 服务是一种可满足自然语言处理需求的云端解决方案，借此可开发 UI，通过机器学习技术理解语言中包含的实体。一起看看这个领域的几个重量级选手。



Comparison of Most Prominent AI Services

	wit.ai	api.ai	LUIS.ai	IBM Watson
Free of charge	✓	✓ but has paid enterprise version	✓ it is in beta and has transaction limits	30 days trial then priced for enterprise use
Text and Speech processing	✓	✓	✓ with use of Cortana	✓
Machine Learning Modeling	✓	✓	✓	✓
Support for Intents, Entities, Actions	✓ Intents used as trait entities, actions are combined operations	✓ Intents is the main prediction mechanism. Domains of entities, intents and actions	✓	✓
Pre-build entities for easy parsing of numbers, temperature, date, etc.	✓	✓	✓	✓
Integration to messaging platforms	✗ web service API	✓ also has facility for deploying to heroku. Paid environment	✓ integrated to Azure	✓ possible via API
Support of SDKs	✓ includes SDKs for Python, Node.js, Rust, C, Ruby, iOS, Android, Windows Phone	✓ C#, Xamarin, Python, Node.js, iOS, Android, Windows Phone	✓ enables building with Web Service API, Microsoft Bot Framework integration	Proprietary language "AlchemyLanguage"

Created by ActiveWizards

◆ Wit.ai

根据官网介绍，Wit.ai 平台可以让开发者轻松地构建可以通过语音或文字方式交流的应用程序。该技术最近已被 Facebook 收购，并已纳入 Facebook 的 Bot API 中。Wit.ai 可帮助用户无需考虑内部逻辑，轻松构建机器人的行为，并可通过多种语言实 Wit.ai 中，用于定义行为的关键概念叫做“故事”。举例来说，这些故事代表着各种可能产生的对话所对应的基本骨架。一个“故事”可包含一组相关意图，而意图本身则是用户定义的实体，其中包含了未被用于定义整个流程的各类特性。

随后即可通过样本训练用于自然语言处理的机器学习模型，这是一种很好的做法。只需要告诉 Wit.ai 需要接受怎样的输入，当用户发送了类似的请求后，即可酌情做出响应。

Wit.ai 提供了一套强大的语言实体理解机制。此外还有一个很棒的功能，该平台可以为实体分配角色，借此可进一步促进服务器端的处理工作。为了与服务器端交互，Wit.ai 还实现了 “Bot sends” 命令，并可通过与 Webhook 的集成实现自定义机器人操作，例如调用 API。

◆ Api.ai

Api.ai 是另一个可以帮我们构建支持自然语言处理的机器人的平台。与 Wit.ai 严重依赖意图进行预测的做法不同，Api.ai 由两个重要概念：意图和上下文情境（Wit.ai 也有上下文情境，但用途有所差异）。“意图” 充当了桥梁的作用，可将用户的请求与对应的操作连接在一起；而以字符串值形式呈现的 “上下文情境” 则可用于区分请求以及存在各种细微偏差的意图。

Api.ai 与 Wit.ai 的基本工作流也有所差异。当 Api.ai 接到用户请求后，首先会将其与意图相匹配（如果没有找到匹配的意图，则直接使用默认意图），随后调用相应的操作。意图的匹配过程可以通过上下文情境列表加以限制，借此确保始终具备可匹配的意图（匹配可以产生或删除上下文情境），借此创建出类似于包含不同状态应用程序的工作流。

与 Wit.ai 类似，Api.ai 也可以提取意图。更重要的是，该平台自带一个 Slot-filling 系统的实现(Slot-filling 系统是一种方法,可用于从用户处请求信息，并将其与已获取实体进行关联并继续向用户索取缺失的实体，借此从用户提供的信息中提取实体)。

服务器端逻辑同样包含 Webhook，而 Api.ai 基本上是在调用各种服务并获得响应。这里有个重点需要注意，服务器端代码可以修改上下文情境，因此会影响预测流的结果。

◆ Microsoft Language Understanding Intelligent Service

LUIS 差不多算是 AI 服务领域的一个新选手，由微软在 2016 年 Microsoft Build 大会上发布。与竞争对手类似，LUIS 提供了实体识别和训练系统，可以通过具备层次结构的实体区分细分的含义（有些类似 Wit.ai 中的“角色”）。

LUIS 需要通过意图预测所要执行的操作，并使用了与 Api.ai 相同的逻辑。训练过程同样是通过 UI 进行的，因此用户可以用灵活的方式加以训练。用户请求日志可以帮助用户方便地做出解释并对解释进行纠正，借此进一步训练模型。

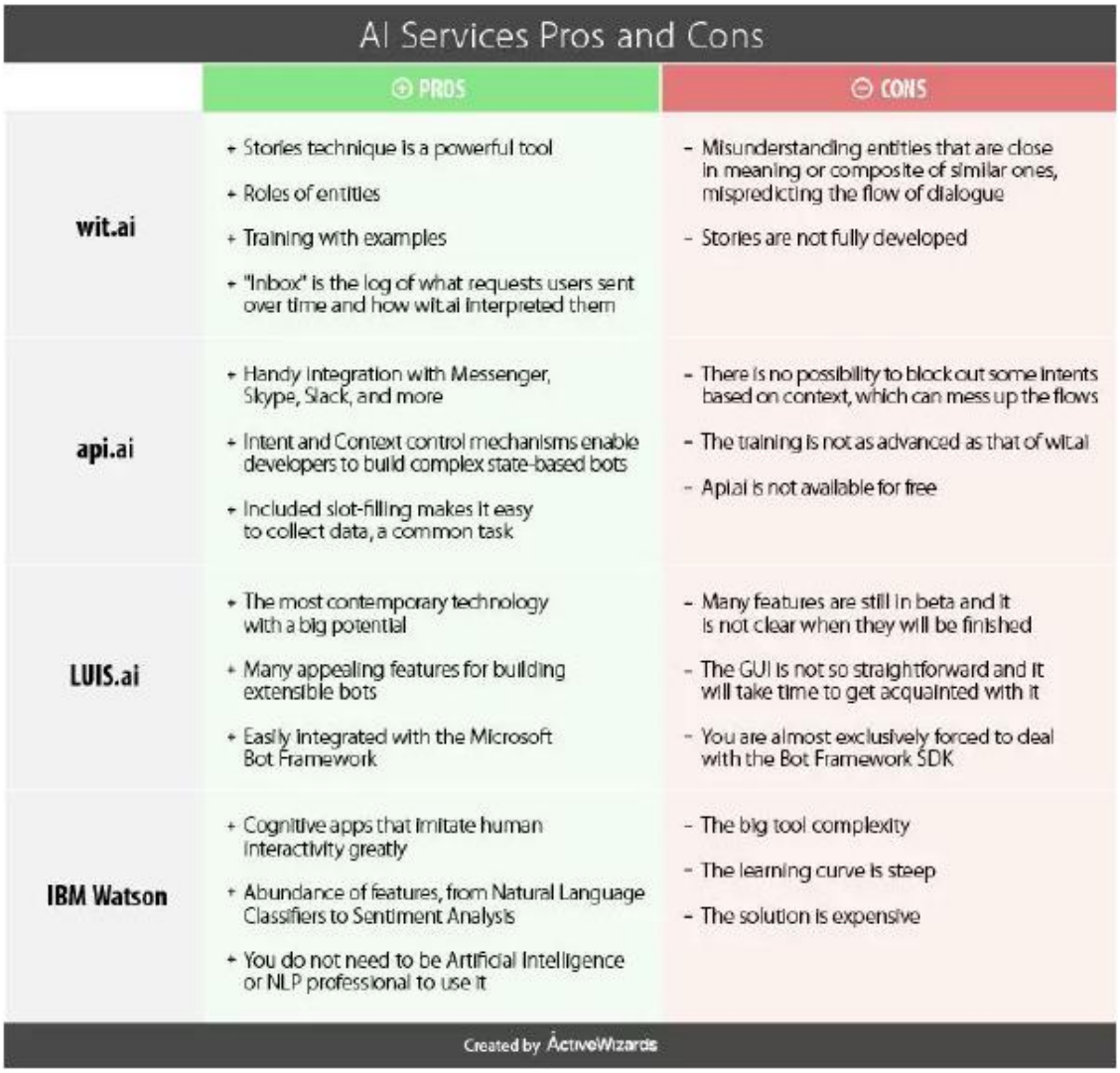
LUIS 同样具备操作履行能力，可收集所需意图和上下文情境，进而执行一系列连锁操作。该平台目前还是测试版，仅可用于简单的测试。此外该平台还有另一个测试版功能值得我们注意，可以通过专门设计的对话支持帮助我们对相关请求进行整理，并对各种问题进行分组，借此通过机器人为用户提供更简洁的对话。

◆ IBM Watson

IBM Watson 是一种认知云服务，可实现包括自然语言处理、语音识别、情绪分析、对话在内的各种功能。可能有人还记得（或者已经忘了）IBM Watson 曾作为一种具备认知能力的超级计算机系统，在电视问答节目 Jeopardy 中战胜过人类。是否纳闷它们之间有没有什么关系？没错，是有一些关系，IBM 将那台超级计算机的强大能力带到了云端，创造出一个可以帮助我们完成各种任务的庞大平台。

然而该平台的不足之处在于，整个平台提供了太多功能，可能会让用户疑惑，进而需要花费大量时间来决定自己到底该用哪些功能。而就算确定了要使用的功能，随后可能也需要投入大量资源，才能顺利开始使用。

此外该服务还非常贵（每个 API 调用约 2 美分），因此可能并不适合初学者。但对于有着广泛并且明确用例的企业环境，IBM Watson 依然是一种不错的解决方案。



结论

聊天机器人正在飞速崛起并依然在不断完善，更加贴近最终用户，能与更多现有技术集成。通过深入理解其本质、功能以及运行原理，可以帮助我们开发更有效的机器人，借此改善产品的营销、广告宣传以及整体消费者体验。

目前我们可以通过多种平台创建聊天机器人，这让人激动，但同时也会让人感觉畏缩，但每种平台都是针对不同用例设计的，在机器人开发过程中包含了不同的角色。随着深度学习技术的进一步发展，我们也同样期待对话式 AI 在不愿的未来也能利用这种技术，向着通过图灵测试的最终目标迈出更大的步伐。

阅读英文原文:

A Comparative Analysis of ChatBots APIs
https://medium.com/activewizards-machine-learning-company/a-comparative-analysis-of-chatbots-apis-f9d240263e1d