The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms

Lorenz Cuno Klopfenstein
cuno.klopfenstein@uniurb.it

Saverio Delpriori
saverio.delpriori@uniurb.it

Silvia Malatini
silvia.malatini@uniurb.it

Alessandro Bogliolo
alessandro.bogliolo@uniurb.it

Department of Pure and Applied Sciences
Piazza della Repubblica 13, 61029 Urbino
University of Urbino "Carlo Bo" — Italy

ABSTRACT

This work documents the recent rise in popularity of messaging bots: chatterbot-like agents with simple, textual interfaces that allow users to access information, make use of services, or provide entertainment through online messaging platforms. Conversational interfaces have been often studied in their many facets, including natural language processing, artificial intelligence, human-computer interaction, and usability. In this work we analyze the recent trends in chatterbots and provide a survey of major messaging platforms, reviewing their support for bots and their distinguishing features. We then argue for what we call "Botplication", a bot interface paradigm that makes use of context, history, and structured conversation elements for input and output in order to provide a conversational user experience while overcoming the limitations of text-only interfaces.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces

Author Keywords

Conversational UI; mobile UI; botplication; bots; messaging

INTRODUCTION

The reach of third-party applications for mobile operating systems—usually marketed as "apps"—has grown dramatically over the last years. A decade after the launch of the *App Store* for iPhone in 2008, the number of apps available to smartphone users now reaches into the order of millions. Their reach, impact, and economic significance cannot be ignored.

However, recent statistics have shown that users frequently make use of a limited set of popular apps. Moreover, most smartphone owners are accustomed to installing nearly zero new apps on their devices on a monthly basis [8]. For app developers it has become more and more difficult to make their products visible in an already crowded application store, where competition is harsher than ever.

Instant messaging (IM) and social network applications consistently take up the top spots among the most used applications. IM apps in particular have shown tremendous growth over the last years, recently taking over the lead in terms of number of users, growth, and user engagement [35]. Messaging platforms are getting immense attention and—as their success depends on the network effect of their users—ferociously compete for market share.

Starting in 2014, many messaging systems have introduced support for so-called "bots": enhanced conversational agents that can chat with users, right inside the messaging app itself. These bots live inside a very familiar place: in a conversation thread, right next to private conversations with friends and relatives, which is increasingly the most used feature of a user's smartphone. Most users in fact use messaging apps several times a day and have a well-rounded understanding of their interface and manner of working.

Instead of trying to attract people to new apps, bots provide an incredibly convenient way for services and developers to engage with users where they already are, making use of the existing conversational paradigm. Even if the conversation follows the familiar and recognizable conventions of a messaging system, the exchange does not need to be text-based. Thanks to the richness of the development frameworks made available by many messaging platforms, bots can also exchange information using complex messages and UI primitives, which allow the conversation to be more efficient.

Bots are growing fast in number and many IM platforms have started offering *bot stores*, just like mobile OS platforms do for apps. In fact, the robustness of the messaging ecosystem—encompassing its existing user base and distribution channels—make bots the perfect bridgehead to transform instant messaging systems into software delivery platforms. This could be a lucrative endeavor for messaging apps, bringing them into competition with the mobile platforms on which they operate.

However, to the best of our knowledge, a thorough and detailed appraisal of this phenomena is still missing.

DIS 2017, June 10–14, 2017, Edinburgh, United Kingdom.

ISBN 978-1-4503-4922-2/17/06.

DOI: http://dx.doi.org/10.1145/3064663.3064672

Contribution

Our work examines this trend with a history of existing chatbots and a survey of popular IM platforms. We then propose the definition of "Botplications": conversational agents designed to follow a set of guidelines that can serve as functional replacements of apps. The distinguishing characteristics of such agents are identified, including differences and advantages over traditional applications. We then speculate about the future trends of these technologies and conversational interfaces.

HISTORY OF CONVERSATIONAL INTERFACES

The idea of building a computer—or better, a program—able to talk with humans, giving the illusion of a true human-to-human interaction, can be dated back to the '50s, when Alan Turing proposed his seminal "Imitation Game" [31]. Better known as the *Turing test*, it aims to determine if a machine can give the impression to other humans of being human itself. Today the Turing test is still used as a test for evaluating to which extent a program, a *bot*, is human-like: the Loebner Prize is annually assigned to the best computer system pretending to be human.

Chatterbots

One of the first examples of such a program was *ELIZA*, created by Weizenbaum in 1966 [34]. ELIZA did simply answer all the user's utterances with other vague questions, giving a rough impression of a Rogerian psychotherapist. Weizenbaum's first example of a so-called "chatterbot", and its crude attempts at fooling users into believing they were interacting with another human, laid the foundations for chatterbots and bots in the following 50 years, until today.

Another example of a popular and more recent chatbot is *ALICE* (Artificial Linguistic Internet Computer Entity), which won the Loebner prize in 2000, 2001, and 2004. It was developed by Wallace in 1995, inspired by ELIZA. ALICE relies on a simple pattern-matching algorithm: it uses pattern templates to represent input and output and makes use of recursive techniques to apply different rules [33]. The underlying intelligence is based on the Artificial Intelligence Markup Language (AIML), which makes it possible for developers to define the building blocks of the bot's knowledge [32].

Personification

From the initial attempts with ELIZA and ALICE, to the ambition of developing "intelligent" agents able to fully converse with humans using natural language, bots have been increasingly employed in many fields with a discrete success, yet with limited spread. They have been applied in e-commerce applications [17], like *Nicole*, a virtual assistant with customer service tasks, or *Anna* by IKEA; in education, like CHARLIE [23], a bot that lets students communicate with the online learning platform INES, or TQ-bot [10], dedicated to student tutoring and evaluation; information retrieval [26, 29], and e-government services [21]. Most of those bots are based on AIML and ALICE [27, 30].

Along with chatterbots, supported by the evolution of Automatic Speech Recognition (ASR) systems in the '80s, Spoken Dialog Systems (SDS) started drawing the attention of academics and the industry [22]. The conversation did move from a textual to a spoken channel, as a presumably easier and more natural interface for humans. ATIS [13] was a telephone-based flight reservation system, funded by DARPA, developed in 1990, in the USA; in the same years, SUNDIAL was developed in Europe [25], with the same aim of giving information about flights via telephone. The latter system was able to understand between 1.000 and 2.000 words in four different languages. Mercury was a similar, more recent, undertaking in this field [28]. These systems addressed several difficult technological issues, including speech recognition, understanding the user's requests, and giving the system the means of offering satisfactory answers.

A further step toward the "personification" of intelligent systems has been achieved with the development of Embodied Conversational Agents (ECA) [22], in the late '90s. Animated characters, with human features, able to simulate emotions with facial expressions and gestures, have been employed in different fields to interact with humans. They seem to be perceived as more trustworthy and agreeable, and are thus employed with the hope of being more readily accepted in everyday life applications than simple textual or spoken interfaces. Cassell et al. developed REA (Real Estate Agent) [4], a humanoid expert in real estates, that interacted with users and could sense them by means of cameras. Kopp et al. give another example of an ECA that has been used as a museum guide, in order to engage visitors and to interactively chat with them. In a real world study it was observed that users were inclined to use human-like communication strategies and that they perceived the agent as being sociable [19].

Instant messaging

After the success of browser chats at the beginning of the '90s, like IRC (Internet Relay Chat), the end of that decade has seen the spread of instant messaging services, such as AOL Instant Messaging (known as AIM), Yahoo! Messenger, and MSN Messenger. Many of these platforms allowed users to add bots to their contact lists as if they were real people: they could then exchange simple text messages and receive different kind of information [17]. A famous example of such a bot was *SmarterChild*, that could converse about breaking news and the weather.

Virtual Private Assistants

Only few years ago, "smart assistants" have started drawing the attention of the greater public. According to McTear et al., different reasons have influenced this new success of conversational interfaces [22]: the progresses in artificial intelligence assistive technologies, like speech and image recognition; the emergence of the semantic web; the increasing availability of connectivity and improvements in device hardware; and the renewed interest of big technology players. Apple's *Siri* (generally considered to be the the first public voice-enabled Virtual Private Assistant), Microsoft's *Cortana*, *Google Now* and *Google Assistant*, Amazon *Alexa*, and Samsung *S Voice* are the main actors of the last five years in the field of conversational interfaces.

Most notably, the reasons of the growing success of VPAs could also be explained by the many differences in respect to older personal assistants. First of all, smartphones have become pervasive in everyday life and are perceived as more personal than any other device. The same goes with the assistants, which are always present for the user. Modern assistants are perceived as more flexible: they are not limited to a very narrow task, but are able to interact with a plethora of applications, internal and external to the device [3]. The interaction is more "human-like", using simple, yet impressive tricks: the effort made to provide direct answers, in integrated and often spoken dialogs; improvements to the inference of the user's intents and the correction of ambiguities; better interpretations of the input's semantics; the capability of answering sassy questions, giving the illusion of a synthetic and likable personality. All these factors have probably made the luck of modern VPAs and the failure of older ones, like for instance *Wildfire*, a VPA of the mid '90s, multi-modal and phone based, but with poor human-like interaction capabilities [15].

Even though basic services of the aforementioned personal assistants are somewhat similar (including web search capabilities, event planning, voice calls and messaging, music playback, personalized notifications, weather information retrieval, etc.), some peculiar aspects can differentiate them. Siri and Cortana are the more similar ones, acting as personal assistants with a well defined "personality" that transpires in many of their answers. Google Now is also similar to these first two, but lacks a well defined personality and pulls information directly from the user's online Google Account, possibly raising some privacy questions. Samsung S Voice is probably the more "classical" one among the others and may not be perceived as personal like the other ones. Lastly, Amazon Alexa is going in a somewhat different direction: developed for Amazon Echo, a smart speaker, it brings the Internet of Things (IoT) closer to everyday life and it can be integrated with different other devices and services, especially in regard to home automation and entertainment.

It is also worth mentioning the IBM statistical responding agent called "Watson" [9], which stepped into the limelight in 2010 for participating in the "Jeopardy" television program. In that occasion Watson beat both his two human competitors only relying on an off-line database of unstructured contents. In more recent years, IBM further developed Watson, turning it into a powerful Artificial Intelligence (AI) assistant employed in health-care and custom-care scenarios. Watson can also be exploited as a cloud-based cognitive services of composable AI building blocks [37]. Developers can train the Watson AI to answer to answer questions posed in plain natural language about particular intent, and use it to build applications like chatbots.

The Rise of Bots

In the last couple of years a new approach to conversational interfaces has been reclaiming prominence: an apparent return to the classic *chatterbot* texting interfaces has been observed, with the main difference that these new bots have gained additional capabilities and now "live" in the cloud.

Starting in 2014, many online messaging systems (like Kik, Telegram, and WeChat) have opened up to third-party developers, offering the means to building bots and programmatically exchanging messages with users through their platforms. Application Programming Interfaces (APIs) for bots expose high level services (messaging, payments, bot directory, authentication, etc.) and UI elements (buttons, locations, images, etc.) giving developers the possibility of implementing innovative services through a conversational user experience.

Services offered by these new bots are of a higher level than the ones offered by their predecessors. Bots often feature access to other services that have utility in everyday life, such as ordering food, managing an e-commerce purchase, booking restaurants, ordering a cab, and so on. Examples include health care bots such as *Nombot* [11], a bot helping users to track their daily food consumption on Telegram; educational bots, like the one by Chaniago et al. that let parents track their children's presence at school [5]; educational help, like *MOOCBuddy* that proposes MOOC courses over Messenger [14].

It is also of note that the possibility of building more helpful and practical bots is in good part due to the increased availability of service APIs open to third-parties and the rise of the "platform" business model [7]. The recent decade has seen a growing attention to *open data* as well, which is increasingly made available by governments or administrative entities, also providing a useful foundation for many bots.

Since data and services are more and more accessible through programmatic interfaces, and given that bots often offer a simpler development platform than apps in terms of development and maintenance efforts, the task of offering access to such services through a conversation interface is very approachable.

Many of the major messaging platforms lately have started offering *bot directories*: repositories similar to app stores, listing the available bots that can be accessed through the platform.

Beyond "Chatbots"

In light of the recent resurgence in popularity of chatbots, we argue for a new and more significant taxonomy of autonomous conversational agents.

Firstly, the very nature of the term "chatbot" sets certain expectations about the agent's interactions: users may assume that a chatbot will be able to "chat" with them using natural language, but only a subset of autonomous agents are designed to simulate natural conversations. Agents that are not will inevitably fall short in their understanding of humans, who have been generally shown to tend to impatience when interacting with software agents [16, 3].

In fact, the "chat" verb trivializes the potential of bots, leading back to chatterbots as silly novelties with which to exchange messages on a command-line, mimicking human idiosyncrasies without any further purpose, just like with the aforementioned ELIZA chatterbot. Instead, as we discuss in the next section, bots have the potential to be a powerful and efficient software platform, only incidentally accessible through text messages.

BOTPLICATIONS

Among the plethora of new and old conversational interfaces, we identify a category of conversational agents that, having been designed according to principles of simplicity and effectiveness, can serve as functional replacements of mobile applications.

We call them "Botplications".

A Botplication is an agent that is endowed with a conversational interface accessible through a messaging platform, which provides access to data, services, or enables the user to perform a specific task.

In particular, a Botplication is generally characterized by the following distinguishing features.

Thread as app

Botplications are stepping stones in the evolution from an *app-centric* mobile OS experience—where the whole user experience of the device is concentrated in mostly independent applications that serve as an enclave of unique services and functions—to a *thread-centric* experience. That is, a user experience where services and information are provided as streams of messages and notifications, presented using a coherent and consistent set of interface paradigms.

Messaging threads are independent conversations that enclose personal relationships. Exchanging messages is in fact the primary method for users of a mobile OS to keep up relations with other people, in a very natural and intimate way. These personal conversations and relations can be naturally extended to services and businesses.

Each one of these threads is an entity that can send updates and notifications to the user, even in multiple parallel conversations, while taking advantage of the built-in facilities of the messaging system.

Among these facilities, for instance, the ability of users to retain total control over each single thread: they can chose to reply, mute the conversation, or even to permanently delete the thread. Also, threads have the capacity of keeping track of read/unread status and message drafts on multiple devices or platforms. The user has the ability to search for a message across all open threads, instead of having to remember in which particular app or service the information is hidden. Incoming messages on any thread are notified to the user in a familiar way.

Most modern messaging apps are in fact presented as a threaded "inbox", automatically grouping messages from the same sender and surfacing recently updated conversations that may be of interest. Instead of having new information dispersed across a number of isolated apps, each with its own custom interaction modes, users can rely on the fact that frequently or recently used services are automatically promoted to a visible position.

A user's "inbox" acts as a replacement for the app-centric homescreen of a mobile OS, where the most recent threads serve the same purpose of a dynamic list of favorite apps.

Conversation threads make it easy to provide integrated tools and services that make it easy to accomplish regular tasks, but in a recognizable and familiar place: a personal relationship developed through the exchange of text messages.

History awareness

Just like mobile apps, Botplications are designed to solve a specific and circumscribed issue. However, unlike most apps, Botplications inherently keep an exhaustive chronological log of past interactions with the user in their thread.

This ingrained feature of a thread of messages allows the user to explore past information in a familiar way, by scrolling through a timeline or by using built-in temporal search. Users can approach the service with more confidence, since past state appears frozen in previous messages and data does not unexpectedly vanish.

History can only change with familiar and explicit user actions, deep-seated in muscle memory for the majority of users used to receive and send text messages. New messages are appended to the history in a predictable and well-known way, while past interactions can be deleted through explicit action of the user.

History also serves as guidance to users, since past commands (and the results they generated) can be easily used as a template for future requests.

Also, past history provides the context in which all future interactions can be evaluated. Information collected by a Botplication *should* be maintained and used in order to streamline requests, skipping questions and automatically disambiguating between different choices if possible. A conversation is a natural and effective way to collect personal information, interests, purpose, and preferences of the user, all of which can be employed in order to improve the quality and accuracy of the service.

Enhanced UI

Despite the fact that Botplications derive from chatterbot-like conversational interfaces, their UI does not *have* to consist of mere plain text messages.

Modern messaging platforms support a variety of messages, including pictures, "stickers" (preset or custom images that convey emotion), videos, and audio. Most platforms also allow the transmission of packaged data, as in the case of geo-locations or contact information, in addition to generic data files. While these platforms of course do not offer the graphical capabilities of apps, it is important to make use of the features available and to exploit the conventions of the messaging platform. For instance, instead of printing out a raw URL, some platforms may display links as nicely formatted cards with a preview.

Moreover, on many messaging platforms even plain text messages can be enriched with a limited subset of rich text formatting. Bold, italic, and embedded links are the most common formatting options, available through markup languages such as a subset of HTML or Markdown. Unicode-encoded *emoji* characters can also be used to efficiently transmit additional meaning or to convey emotion.

Typing should be reduced to a minimum: ideally it should be limited to very short and concise commands and replies, of few characters. A fitting comparison can be done with UNIX commands, which are an example in terseness, as they were designed to efficiently work over teletypewriters—the primeval example of text-based interfaces—and reward users with powerful functionalities through very little typing.

Even if it can be argued that younger generations are getting more accustomed to typing on touch-based soft keyboards, the practice of shortening common words or using abbreviations in everyday messaging by itself demonstrates why full-fledged and verbose conversations should be avoided if possible. Interactions with Botplications should follow linear conversation routes and avoid complicated branching or multiple complex dialogs: anything that cannot be accessed through a couple of taps on the screen will become tiresome to most users. In a user study by Azenkot et al. the average text entry for all participants on a popular mobile OS keyboard was of 41.01 Words per Minute (WPM) [1]. Also, the theoretical maximum typing rate predicted by MacKenzie et al. was of 43.2 WPM for expert users on QWERTY keyboards [20], which is of course far lower to what can be achieved by expert typists on a physical keyboard [12]. Text interactions for mobile users should therefore be kept short and precise.

Many messaging platforms also feature advanced structured message forms, which can further enhance the flow of conversation. Structured messages may, for instance, contain buttons for preset "canned" replies, show different alternatives in a rich representation, or show a list of available commands. Examples of such messages are shown in the next section. The advantages of structured messages are manifold: (1) They constrain the conversation into a limited number of expected outcomes, reducing the possibility of users feeling trapped in a dead end where they have to "guess" their way out. (2) They push the user to use the service, suggesting how the conversation can continue. They also reduce the need for the users to "explore" the interface, making it easier to learn and use. (3) Buttons and quick replies reduce interactions to a single tap instead of requiring complex typing. (4) The service can be implemented more easily.

Botplications try to fill the gap between plain conversational interfaces—which are inherently inefficient to use and offer little way in terms of user experience customization—and the world of the full graphical interface.

Limited use of Natural Language Processing (NLP)

Given the aforementioned rising interest in Virtual Private Assistants (VPAs), voice appears to be a natural fit for conversational interfaces. However, to the best of our knowledge, existing bots on messaging platforms avoid voice processing and choose digital text as the most direct and unambiguous form of communication, possibly adopting NLP systems in order to extract commands and intent from the user's messages. Even if natural language understanding is getting progressively more advanced [3], in many scenarios complex dialogs break down because of simple misunderstandings, because of unexpected turns of phrase (human language shows incredible variation and rarely follows a script), or because the user's context cannot be fully elicited from the conversation [22, 2].

Botplications should not attempt to provide the complex experience of a full VPA. Instead, in keeping true to the principles of simplicity and effectiveness, Botplications should almost completely avoid natural language where possible. At the scale of a single bot, going after AI is mostly excessive and counterproductive, when the same results can be obtained with simple text commands using a limited structured language.

Botplications should in fact not pretend to be human: except in cases where this is desirable (e.g., for customer support or as a barrier before an actual conversations with a human operator is activated), it should be clear to users that they are talking to a machine. Even if artificial delays or "is typing" indicators can be used in order to make the conversation more familiar to the user, faking human responses risks to increase the perceived distance between service and user instead of decreasing it. Texting a computer that doesn't understand what the user is saying can be a frustrating experience, in particular when the computer hides its failures inside a dialog that is artificially kept "natural" and "human-like". This hides failure points in the conversation and makes the user feel less in control of the interaction.

However, this does not imply that bots shouldn't show a personality or take advantage of humor and emotional responses to provide a charming and likable interaction with users.

Botplications should rely on the limited—but accurate—interaction tools the messaging platform makes available, while NLP frameworks can optionally be employed to accommodate unforeseen user requests. AI and deep human-like dialogs are red herrings in the current development of conversational interfaces: Botplications should be about accessing services efficiently, a command-line-like interface to cloud-based APIs, not talkers.

Message self-consistency

Each single message sent by a Botplication should contain the full context of the conversation and should embody what a single UI screen represents for mobile apps. Users should not have to browse through their conversation history in order to figure out what they are attempting to do and what the service is expecting. Each message has an atomic meaning and stands on its own.

The scope of each message must be clear, its intent must be explicit, and what action must be taken by the user—if any—must be explicit and unequivocal. Indeed, a message delivered by a Botplication should be conceptually seen as a *micro application*, while the conversation is a timeline of past application screens. As mentioned before, structured forms of messages, which include buttons or commands, should be used where appropriate.

Some messaging platforms have, in fact, the ability to alter messages after they have been delivered. In that case, messages can be changed based on the availability of new data or other changes in state, giving the impression of a living view on the service.

Table 1: Features of major messaging platforms that support bots (as of October 2016).

Platform	MAU†	Groups	Mentions	Message types	Buttons	Carousel	Quick reply	Payment
Messenger	800 M			Picture, video, file, voice.	✓	✓	✓	✓
	Persistent menus, several message templates (Airline trip, Buy, Receipt, Web link, etc.).							
WeChat	700 M		✓	Picture, video, sticker, voice, location.			✓‡	✓
	Deep-links through QR codes, Rich media and Music messages. Integrated web views§.							
Skype	300 M			Picture, video.	✓	✓		
	Several message templates (Hero image, Thumbnail, Receipt, Sign-in, etc.), phone call support.							
Line	220 M	✓		Picture, video, sticker, voice, location.	✓	✓		
	Imagemap message template (picture with multiple hot-spots).							
Telegram	100 M	✓	✓	Picture, video, sticker, file, voice, location.			✓	
	Persistent commands. Deep-links to conversations.							
Kik	80 M	✓	✓	Picture, video, sticker, voice.			✓	
	Kik code identifiers, browser integration via Javascript.							
Slack	$\gtrapprox$ 3 M	✓	✓	File.	✓			

† Monthly Active Users, based on most recent quarterly report published at the end of 2016. Numbers for Slack are an approximation based on known Daily Active Users.
‡ In the form of custom defined menus shown in the conversation UI.
§ WeUI, HTML/Javascript library that provides web-based UI elements coherent with the WeChat app: https://weui.io/0.4.x/.

Guided conversation

An important part of an application's user experience is based on user guidance and, likewise, the same care should be applied when designing conversational interfaces through text messages. In fact, because of the free-form nature of the medium, it is easy for bot users to get lost and not to be certain of what commands or what exact syntax is required to perform the desired action.

A successful Botplication guides the user through a task in order to avoid this impasse. The service proactively suggests actions that are likely to follow up after the current interaction, offers alternative choices when needed, and generally offers a framework in which user interactions feel reliable. This can be achieved using the same UI enhancements mentioned before, that is through the use of buttons, formatted messages, or built-in menus that offer interface guard rails to the conversation.

Also, notice that when starting the first interaction with a bot, many messaging platforms offer a way to show a welcome message to the user. The design of the onboarding experience must take into account the initial user guidance and ensure that all functionalities are readily available.

TECHNOLOGICAL SURVEY

In this section the most popular messaging platforms that support bots through their APIs will be taken into exam, describing distinguishing features of each one.

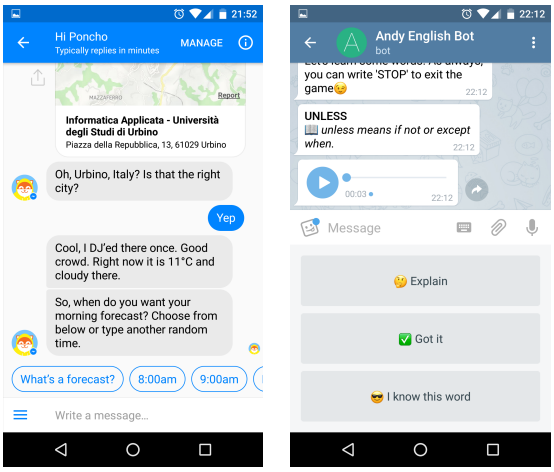
All of the platforms mentioned allow third-party developers to register an identity for a virtual agent and to programmatically receive messages from any user of the platform, either by accessing an API end-point (*pull* mode) or by being called back by the platform itself using a "web-hook" (*push* mode). All platforms, in any mode, make use of the HTTP protocol.

The *Kik* messenger service launched in 2010 and introduced bots in 2014. On April 2016 the platform launched a "bot-shop", which includes a directory of available service bots. *WeChat* is a popular IM system in China, first launched in 2011, that always included more features than simple messaging. The platform includes support for "Official accounts", which can provide services to users. It includes a payment system that allows the exchange of monetary gifts, known as "red envelopes". *Telegram* published its bot support API on July 2015, further expanding it with the 2.0 API in April 2016. *Line* also launched its first messenger API in 2015, while Facebook's *Messenger* and *Skype* joined the game only later, including support for bots in April 2016. The former also added support for payments in September 2016.

Given the rising interest in bots, most platforms are rapidly iterating and evolving their APIs and services to third-party developers. This survey is updated to March 2017.

In Table 1 the platforms considered in this study are listed, firstly reporting the amount of *Monthly Active Users* (MAU), which gives an approximation of the relative popularity of each system. Furthermore, the table shows whether the platform supports *group messaging* (i.e., if one or more bots can be added to a group conversation between real users) and *mentions* (i.e., if bots can be "called into" a conversation using a special combination of characters). Both of these features allow bots to be used in order to perform specific tasks for a group of users instead of only one. The table also reports the different *message types* supported by the platform (includ-

ing pictures, voice, video, stickers, and structured messages, discussed in the next section) and other significant features.



(a) (b)

Figure 1: Features that allow users to pick suggested replies. (a) *Quick replies* on Messenger. (b) *Button keyboard* on Telegram.

Interface features

Messaging platforms distinguish themselves not only for their underlying technical aspects, but also because of the different user interface elements they offer to bot developers and, ultimately, to the end-users. While sending and receiving text and basic multimedia messages is a common feature, more structured messages are available only on some platforms and often differ in key aspects.

Many platforms offer a way to suggest canned replies, which can be sent with a single tap. For instance, on Messenger bots may display a selection of *quick replies* that appear on the bottom of the conversation and remain valid for one interaction, as shown in Figure 1a. On Telegram and Kik, bots have the ability to replace the keyboard with suggested responses, as shown in Figure 1b.

While these preset replies can change in the course of the conversation, other UI features can immutably be added to the chat. In Figure 2a a list of *commands* are shown. On Telegram, like on Slack, commands are characterized by starting with a '/' character and are always available to the user to perform tasks in a command line-like experience. A similar system, but with a hierarchical structure, is shown in Figure 2b: WeChat allows developers to add a fixed menu to the chat interface, showing a maximum of 3 first-level options and several second-level ones. The same system has been also adopted by Messenger in March 2017. Commands and menus alike give direct access to a bot's main features.

Other UI features are not bound to the overall conversation, but are instead tied to a specific message. For instance, *carousel* messages shown in Figure 3a make it possible to include multiple rich cards, provided with an image, a description, and a button, and make them horizontally scrollable to the user.

In Figure 3b a message with *embedded buttons* is shown: in this case the actions provided to the user are not persistent throughout the conversation, but are limited to one single interaction. Both message formats allow bot developers to show available alternatives to the user, providing a well defined path for the conversation.

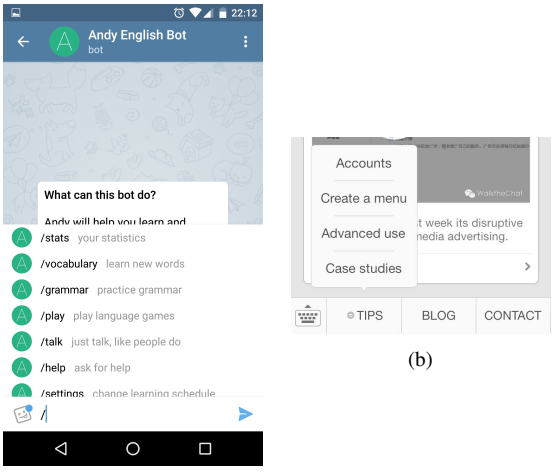
Several other message formats are available to specific platforms, among which messages including a "Buy" link for shopping-oriented bots, messages with music support in WeChat, and flight travel itinerary messages on Messenger.

In terms of making bots interoperate with external systems, it is noteworthy that several messaging platforms have included support for QR Codes or variations thereof. On WeChat, code scanning can be used internally to the system in order to send payments of a preset value to a given contact, which makes the platform compelling for sellers of physical goods as well. On the other hand, Skype, Telegram, and Messenger allow web sites and apps to launch conversations through the use of a specially marked URL. Such URLs include a custom data payload that is sent to the bot in order to track user entry points or to invoke specific actions. This mechanism—also known as "deep linking"—can be used to embed bots in innovative and complex workflows [18].

Advantages of Bots for users

Instant availability

Bots do not need to be downloaded and installed: they are immediately up and running as soon as a conversation is started inside the messaging app. Device storage capacity, installation, and complex configuration steps are not required. This makes bots fast and lightweight, if compared to traditional mobile apps.



(a) (b)

Figure 2: Structured commands available during the whole conversation. (a) *Commands* on Telegram. (b) *Custom defined menu* (with an open second-level menu) on WeChat.

"Instant Apps", a technology for Android still in development, will bring a similar experience to apps as well, making them instantly available to users without installations[1].

Gentle learning curve

Texting has been used since the dawn of mobile phones and this makes it quite natural for users of any skill level. Provided that the bot provides adequate guidance through its features, learning how to interact with it is never an outlandish experience.

Since all bots on the same messaging platform rely on the same UI building-blocks (such as buttons, carousels, quick-replies, and so forth, down to the basic text messages), all bots share a common UI vocabulary. Knowledge in the use of one bot can easily be transfered to the usage of other bots.

Notifications

Many mobile applications implement their own notification system, used as a mean to re-engage inactive users. Instant messaging applications however already include an efficient and functional push notification system, which is available by default without any additional implementation effort.

According to statistics by Localytics[2], more than 50% of users find push notifications annoying. However, receiving notifications through the messaging apps could be perceived as being less invasive.

Social graph and contacts

Users are already accustomed to voluntarily sharing and storing contact information in messaging apps. In fact, as soon as a bot conversation is initiated, the bot automatically receives the user's basic personal data. When participating in a group conversation with multiple users, bots also gain access to the contact information of all participants.

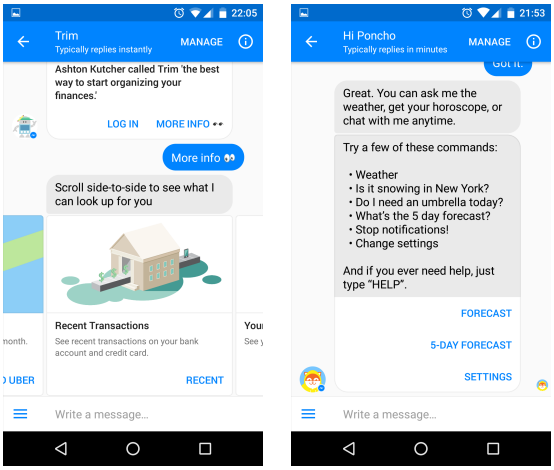
Moreover, all contents generated by an interaction with the bot exist in the form of messages, which can be directly forwarded and shared with friends. Thus contact to the bot can be spread through a user's social graph, without leaving the messaging app or installing new applications.

Platform independence

Bots live inside instant messaging applications without having to worry about which mobile platform they are running on. This makes bots independent from the underlying host operating system: each bot is inherently available on all operating systems the messaging app supports, without any graphical or functional adjustment.

On the contrary, native mobile applications must be adapted or rewritten for each mobile operating system, often with great effort. Users sometimes perceive the same application as being different if they are using it on different operating systems, which can frustrate user engagement. Also, mobile applications must be frequently updated in order to keep up with upgrades to the host system and its features.

[1] http://developer.android.com/topic/instant-apps/
[2] http:/info.localytics.com/blog/the-inside-view-how-consumers-really-feel-about-push-notifications
Accessed: 2016-10-10.



(a) (b)

Figure 3: Structured message templates in Messenger. (a) *Carousel* presenting multiple choices through horizontal scrolling. (b) *Message buttons* embedded in a message.

Authentication

Usually, for each new application or service, users need to register and go through the steps of creating a new account. This is a known obstacle to user onboarding, who notably dislike long registration processes and often do not remember new account credentials.

In contrast, user authentication is not needed in bots. The hosting messaging platform already provides and guarantees the user's identity reliably. Users are uniquely identified by default and they do not have to create additional accounts and passwords in order to interact with the service.

Payment support

Some of the analyzed platforms include payment services, integrated in the messaging system. Often users will have already connected their credit cards or bank accounts to such services, allowing bots to use the existing payment methods in a safe and reliable manner, for both user and developer. In contrast, for each brand new app requiring payment capabilities, users have to re-connect their payment accounts and have to trust the newly installed app.

Discoverability

As seen previously, discoverability is a very real problem for mobile application on application stores: fierce competition and very small economic margins often make it very hard to compete and gain visibility. The same issues exist for bots. Even if many messaging platforms offer "bot stores" with a selection of available bots, emerging and gaining a solid user base remains one of the main development issues. However, bots have some advantages considering that they can be easily integrated in group conversations or shared just like any other contact.

Asynchronicity

Exchanging instant messages is an asynchronous task: after sending a message, users do not have to wait for a reply. Con-

versation threads store the context, letting user free of leaving a conversation and going back to it later, picking up the dialog from the very last interaction. Multiple conversations can be easily carried forward in parallel.

Limited data requirements
Downloading a new app can have a sizable impact on a limited mobile data plan, whereas starting to use a new bot only requires to initiate a new thread in an existing messaging app, with a negligible impact in terms of data traffic. The data size of exchanged messages in an instant messaging app is limited and usually more predictable. This makes bots very attractive for users with limited data plans or in countries with less developed technological infrastructure [24]. In fact, Facebook recently developed "Lite" versions of its applications, specifically targeted at emerging markets, where limited data plans are more common [36].

Advantages of Bots for developers

Communication reliability
Instant messaging applications are naturally designed for fast and efficient message delivery. They are capable of dealing with all kinds of network issues, automatically retrying transmissions and ensuring that the message is either delivered or that failure is handled adequately or correctly reported to the user. Security and privacy of transmitted messages are also guaranteed by many platforms.

On the other hand, mobile applications must implement most of these networking features independently, incurring in the likely risk of bugs or security issues. These issues are multiplied in case of multi-platform applications.

Fast iteration
Since a bot's logic is implemented server-side and no code must be deployed onto user devices, deployment is effectively effortless. Similarly to what happens for web pages or web apps, bot updates are also almost immediately propagated to all users. This form of deployment also completely avoids issues with app stores, for instance due to rejections or delays during application review. This allows for fast and uncomplicated development iterations.

Lack of version fragmentation
The immediate propagation of updates to all users also means that no user risks staying stuck with an old version of a mobile application: just like web apps, bots are always updated to the latest version.

Limited design efforts
Bots heavily rely on the instant messaging application UI and usually have quite limited possibilities of graphical customization. This reduces the interface design time, limiting it to minor customizations of interface elements and focusing on a rich user experience using few simple building-blocks.

Development ease
As seen in the previous sections, several important and demanding services are already implemented by the messaging platform. Services such as user authentication or payments normally require major efforts and attention by mobile app developers. Bots however already include many of these services, which are ready to use through developer APIs.

Moreover, having a generally smaller API surface—which can be used through any development platform and language—developing a bot is generally cheaper and less time demanding than the development of a mobile application.

DISCUSSION

As previously examined, the mobile device ecosystem has been characterized by the popularity of mobile apps and app stores as a software distribution platform in the last decade. Recent statistics have however shown that users tend to rely almost exclusively on a very restricted set of consolidated, preferred apps, instead of trying out new ones, leading the number of new downloaded apps nearly to zero in the short-medium period [8]. Moreover, a 2014 report has also shown the surprising overtake of online messaging platforms usage over social network apps [35].

Bots on instant messaging platforms have recently started gaining widespread interest. As discussed above, not only do bots rely on very popular messaging apps, they also eschew many of the difficulties of mobile app development and distribution. In fact, bots can increasingly be considered as a novel approach to the software distribution problem.

From this point of view, bots show many similarities with mobile web apps, as a distribution system competing with native mobile applications [6]:

1. they allow the development of multi-platform services using a commonly available platform (messaging apps or the web),
2. they are partly limited in terms of what features of the mobile device they have access to (this is particularly true for bots, which however have a different usage paradigm with different expectations from users),
3. the "look and feel" is different from a native app (however, while the user experience of mobile web apps may be different from what users are accustomed to, bots offer a very familiar interface that feels native to the messaging app at least),
4. both approaches favor developers in terms of development, distribution, and maintenance efforts, however bots also ensure a high ease of use to end-users.

These advantages notwithstanding, there are different drawbacks to bots that must be taken into consideration when evaluating their merits as a software platform.

In the first place, not every application is well suited for conversational interfaces. There are several tasks that are inherently more convenient to be tackled by means of a dedicated app, with access to local computational resources and data storage, instead of a remote bot. Other tasks simply work better with a rich visual interactions that goes beyond what a conversational interface can offer.

Even if bot development is very approachable by developers and offers many advanced features at essentially very little effort, a bot is tightly bound to its messaging platform, which may lead to design constraints. There are, very noticeably, inherent limits to the customization of UI elements, the look and feel, and finer details of the user experience.

Bots require an active Internet connection in order to work (in contrast to modern mobile web apps, which may provide an offline experience). This can create discontinuities in service availability and negatively affect the bot's perceived reliability, in case of lacking network coverage or limited data plans, especially in growing markets [24].

Despite the appearance of "bot directories", discoverability is still a crucial issue that must be tackled both by developers and bot platforms. Also, the risk of an overpopulation of bots that hinders the discovery of new services, like for app stores, is equally likely.

Many recent efforts in the field of bots have made use of natural language processing in order to create a generic conversational interface, supporting natural human-like interactions. There is however an inherent distinction between conversational virtual assistants (such as Amazon *Alexa* or Apple's *Siri*), which are well-suited to interactions through spoken natural language, and service-oriented bots, that currently benefit from structured input. While the first kind of bots may provide a more convenient, and in some cases instinctive, interface, the latter can focus on providing a more efficient and lean experience. Also, NLP capabilities are presently not always sufficient to fulfill natural-feeling conversations and they still impose a learning curve on users [3, 2].

Even though some of these issues can be addressed and mitigated in the near future, there are inherent limitations to what bots can offer, which limit how far they can serve as a replacement to native and web apps.

The relative novelty of bots must also be taken into account: the first roots for the spread of bots have been put down, but we are still in the very early days of this new era. A "killer application" for bot platforms, which goes "viral" and fully justifies the rising interest by clearly showing the full potential of bots, is yet to be seen.

Conclusions

In this work the spread of the new generation of bots on messaging platforms has been discussed, describing how the concept of "bot" has changed from its very first stages to the present days.

A definition of *Botplications* has been given, as a conversational agent living on existing messaging platforms, that follows a set of principles of simplicity and purposefulness in providing access to services and data.

Features and limitations of the most prominent messaging platforms have been presented and compared. A technological survey, depicting in detail the peculiar available user interface mechanisms, and advantages of bots, for both users and developers, were discussed in detail.

In conclusion, even if bots will not substitute the whole mobile application ecosystem in the near future, close attention must be paid to the "rise of bots", as a new software platform for delivering services and data to users.

REFERENCES

1. Shiri Azenkot and Shumin Zhai. 2012. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. ACM, 251–260.
2. Emanuele Bastianelli, Daniele Nardi, Luigia Carlucci Aiello, Fabrizio Giacomelli, and Nicolamaria Manes. 2016. Speaky for robots: the development of vocal interfaces for robotic applications. *Applied Intelligence* 44, 1 (2016), 43–66. DOI: http://dx.doi.org/10.1007/s10489-015-0695-5
3. Jerome R Bellegarda. 2014. Spoken language understanding for natural interaction: The Siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 3–14.
4. Justine Cassell, Timothy Bickmore, Mark Billinghurst, Lee Campbell, Kenny Chang, Hannes Vilhjálmsson, and Hao Yan. 1999. Embodiment in conversational interfaces: Rea. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 520–527.
5. Muhammad Benny Chaniago and Apri Junaidi. 2016. Student Presence Using RFID and Telegram Messenger Application. 8th Widyatama International Seminar on Sustainability (WISS 2016), Widyatama University and IEEE.
6. Andre Charland and Brian Leroux. 2011. Mobile Application Development: Web vs. Native. *Commun. ACM* 54, 5 (May 2011), 49–53. DOI: http://dx.doi.org/10.1145/1941487.1941504
7. Sangeet Paul Choudary, Marshall W Van Alstyne, and Geoffrey G Parker. 2016. Platform revolution: How networked markets are transforming the economy–and how to make them work for you. (2016).
8. comScore. 2014. The U.S. Mobile App Report. URL https://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report. (2014). Accessed: 2016-10-10.
9. David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, and others. 2010. Building Watson: An overview of the DeepQA project. *AI magazine* 31, 3 (2010), 59–79.
10. FA Fonte, Juan Carlos, Burguillo Rial, and Martín Llamas Nistal. 2009. TQ-Bot: an AIML-based tutor and evaluator bot. *Journal of Universal Computer Science* 15, 7 (2009), 1486–1495.
11. Bettina Graf, Maike Krüger, Felix Müller, Alexander Ruhland, and Andrea Zech. 2015. Nombot: simplify food

tracking. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 360–363.

12. Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. 2004. A Reduced QWERTY Keyboard for Mobile Text Entry. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. ACM, New York, NY, USA, 1429–1432. DOI: http://dx.doi.org/10.1145/985921.986082
13. Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*. 96–101.
14. Adrian Iftene and Jean Vanderdonckt. 2016. MOOCBuddy: a chatbot for personalized learning with MOOCs. In *RoCHI–International Conference on Human-Computer Interaction*. 91.
15. Philippe Jeanrenaud, Greg Cockroft, and Allard VanderHeidjen. 1999. A multimodal, multilingual telephone application: the wildfire electronic assistant.. In *EUROSPEECH*.
16. Henry Kautz, Bart Selman, Michael Coen, Steven Ketchpel, and Chris Ramming. 1994. An experiment in the design of software agents. In *AAAI*. 438–443.
17. Ian R Kerr. 2004. Bots, babes and the californication of commerce. *University of Ottawa Law and Technology Journal* 1, 1-2 (2004), 20004.
18. Lorenz Cuno Klopfenstein and Alessandro Bogliolo. 2017. The Quiz-Master Bot: a persistent augmented quiz delivered through online messaging. In *INTED2017 Proceedings (11th International Technology, Education and Development Conference)*. IATED, 9806–9811. DOI: http://dx.doi.org/10.21125/inted.2017.2328
19. Stefan Kopp, Lars Gesellensetter, Nicole C Krämer, and Ipke Wachsmuth. 2005. A conversational agent as museum guide–design and evaluation of a real-world application. In *International Workshop on Intelligent Virtual Agents*. Springer, 329–343.
20. I Scott MacKenzie, Shawn X Zhang, and R William Soukoreff. 1999. Text entry using soft keyboards. *Behaviour & Information technology* 18, 4 (1999), 235–244.
21. RP Mahapatra, Naresh Sharma, Aakash Trivedi, and Chitransh Aman. 2012. Adding interactive interface to E-Government systems using AIML based chatterbots. In *Software Engineering (CONSEG), 2012 CSI Sixth International Conference on*. IEEE, 1–6.
22. Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface*. Springer.
23. Fernando A Mikic, Juan C Burguillo, Martín Llamas, Daniel A Rodríguez, and Eduardo Rodríguez. 2009. CHARLIE: An AIML-based Chatterbot which Works as an Interface among INES and Humans. In *EAEEIE Annual Conference, 2009*. IEEE, 1–6.
24. San Murugesan. 2013. Mobile apps in Africa. *IT Professional* 15, 5 (2013), 8–11.
25. Jeremy Peckham. 1991. Speech Understanding and Dialogue over the telephone: an overview of the ESPRIT SUNDIAL project.. In *HLT*.
26. Antonella Santangelo, Agnese Augello, Antonio Gentile, Giovanni Pilato, and Salvatore Gaglio. 2006. A Chat-bot based Multimodal Virtual Guide for Cultural Heritage Tours.. In *PSC*. 114–120.
27. Md Shahriare Satu, Md Hasnat Parvez, and others. 2015. Review of integrated applications with AIML based chatbot. In *2015 International Conference on Computer and Information Engineering (ICCIE)*. IEEE, 87–90.
28. Stephanie Seneff and Joseph Polifroni. 2000. Dialogue Management in the Mercury Flight Reservation System. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3 (ANLP/NAACL-ConvSyst '00)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 11–16. DOI:http://dx.doi.org/10.3115/1117562.1117565
29. Abu Shawar, Eric Atwell, and Andrew Roberts. 2005. FAQchat as in Information Retrieval system. In *Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the 2nd Language and Technology Conference*. PoznanÌĄ: Wydawnictwo PoznanÌĄskie: with co-operation of Fundacja Uniwersytetu im. A. Mickiewicza, 274–278.
30. Bayan Abu Shawar and Eric Atwell. 2007. Chatbots: are they really useful?. In *LDV Forum*, Vol. 22. 29–49.
31. Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59, 236 (1950), 433–460.
32. Richard Wallace. 2003. The elements of AIML style. *Alice AI Foundation* (2003).
33. Richard S Wallace. 2009. The anatomy of ALICE. In *Parsing the Turing Test*. Springer, 181–210.
34. Joseph Weizenbaum. 1966. ELIZA - a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Commun. ACM* 9, 1 (Jan. 1966), 36–45. DOI: http://dx.doi.org/10.1145/365153.365168
35. Will McKitterick. 2015. *The Messaging App Report: How instant messaging can be monetized*. Technical Report.
36. Susan P. Wyche, Sarita Yardi Schoenebeck, and Andrea Forte. 2013. "Facebook is a Luxury": An Exploratory Study of Social Media Use in Rural Kenya. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*. ACM, New York, NY, USA, 33–44. DOI: http://dx.doi.org/10.1145/2441776.2441783
37. Mengting Yan, Paul Castro, Perry Cheng, and Vatche Ishakian. 2016. Building a Chatbot with Serverless Computing. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs*. ACM, 5.