UNIVERSITÀ DEGLI STUDI DI TRENTO

Department of Information Engineering and Computer Science

Bachelor Degree in
Computer Science

Final dissertation

Studying and Implementing Chatbots into Instant Messaging Applications

Studio e Implementazione di Chatbots in applicazioni di messaggistica istantanea

Supervisor
Prof. Giuseppe Riccardi

Co-Supervisor
Dr. Arindam Ghosh

Student
Davide Belli

Academic year 2016/2017

Acknowledgements

I want to take the opportunity to thank many people who have been by my side during these last three years of studies. Either if we met a few months ago, or have been reunited after a long time, or again we have been together for many years... You all contributed to making special my path across this bachelor degree, and, way more importantly, my life.

First and foremost, I want to infinitely thank my amazing family for supporting me through my entire academic career (and life). My mom Monica, for her unconditional and endless love. The one who can't wait for me to come back home for the weekend, and who is always there for me. My dad Marco with his romantic heart hidden inside the man of many talents he is. He never stopped pushing me towards doing my best and even more. You are the best parents in the world, and I know you won't ever let me down. I also want to mention my grandparents, who taught me precious lessons from different lives. Ida and Gianni, for all the times I ate at your place, with you telling me stories that reminded of ancestors and exoduses. Giovanna and Lorenzo, who I had lost for many years. Luckily, we have been reunited and I hope we'll finish looking at all those family photos.

I want to thank Prof. Giuseppe Riccardi, who gave me the opportunity to work on this thesis under his supervision, and Dr. Arindam Ghosh, for helping me with precious suggestions throughout the development and revision of my work. I also want to express my gratitude to all the Professors at the University of Trento who contributed in arising my interest, curiosity and expanding my knowledge about Computer Science.

Next, I want to thank the friends that helped in making the last 3 years much more fun. Starting with my roommate Pier, with whom I shared interests, fatigue, anxiety, fun, holidays and, obviously, my room. My friends at the Computer Science degree at the University of Trento. Gabriele, one of the cleverest guys I know, who will probably share with me the next adventure, this time as Master students. Sara, for keeping company with us and scratching my head during lesson breaks. Then, Nicola, who I've been knowing for 16 years now, and who decided to continue his studies in another country. We have few chances to meet during the year, but I'm always looking forward to them. My group of friends from a lifetime: P, Gabri, Mira, and again Pier. Times changed and we can't meet altogether so often, especially since one of us is forced to its bed and another one in Argentina. I hope we will carry on doing crazy stuff just as at the High School. I never got bored with you. Fabio and Damiano, the other two black-haired guys of the trio. Meeting with you almost every week and having the opportunity to share deeper thoughts and bad jokes is a sign of a great friendship. Finally, all the awesome people from the "Family", the friends at the Grest and my High School class. Many other friends shall enter this list for shaping me as the person I am now, but space and time are running out. I wish you all the best in pursuing and succeeding in your personal goals, and I hope to hear from you sooner or later.

Contents

Summary

In the last years, an exponential increase in computing data and communication volumes have been the consequence of a relevant technological advancement. Many tasks that previously required great computational power can now be accomplished with a relative ease. Moreover, Artificial Intelligence have been receiving a great interest from developers and companies, as it would provide new, powerful ways to solve complex work, eventually overcoming human intelligence.

Among those tasks, one of the hardest challenges is to create computer systems capable to converse with human in an intelligent way via auditory or textual methods. This software, named chatbots or chatter bots[25], is still at low capability levels, but it's expected to face more and more improvement over the next decade.

Chatbots can be applied to different fields for a large range of tasks. The first of them is e-commerce, which is already experiencing a radical introduction of chatbots to satisfy and answer basic user requests. Amazon and Apple published their own bots on Messenger and Telegram to give users a quick way to find products. Many other areas such as healthcare, customer support, communication and entertainment can also be improved by integrating chatbots. Yet, current chatbots are more dumb than smart: they can just recognize a few amounts of intents and doesn't consider context or past information about the user, resulting in frequent errors or verbose requests for detailed explanations. Anyway, there is a branch of chatbots is more intelligent: Virtual Assistants. Starting with Apple's Siri in 2012, companies started to add Assistants to their devices to help users performing activities. Google Now and Microsoft's Cortana followed this trend in the next years, and companies have continuously been improving capabilities and intelligence of their software to overcame competitors. Recently, industries figured out that Assistants can help people in different scenarios. From this intuition, Amazon Echo and Google Home were developed, powered respectively on Alexa and Google Assistant intelligences. Now, people can interact with these state-of-the-art chatbots to retrieve latest news, book flight, buy items on e-commerce websites, interact with devices around the house and much more.

Many computer scientists believe we are standing just in front of the gap leading to a 4th paradigm of customer technologies. That means we are shifting from the Mobile era in which everyone has been installing apps on their device to get services and tools, to the Chatbot era, in which operations would be performed by natural language instructions (vocal or textual). This change appears to be very drastic at first glance, but let's look at what happened in the past. In the 70s, no one would have ever thought computers all over the world could become real-time connected. Similarly, by the end of the past century, mobile phones when nothing but heavy devices few people were willing to bring around. Yet, trend changed quickly and both users and developers soon realized the great capabilities of those rudimental ancestors that now have become cores in everyday technology.

Returning our focus to Chatbots, we will discuss in this thesis how companies are pushing the boundaries of bot capabilities, how different types of chatbots can fulfil tasks related to different areas, and which pros and drawbacks characterize the most used state-of-the-art technologies. As I mentioned early, there is a great slice of chatbots with limited comprehension abilities on the market. The reason is that it is pretty simple to develop small bots able to interact with users following an if-then-else workflow. We will also look at the most popular platforms available to develop such software and how to deploy them on popular messaging applications.

Finally, as the main part of my work, I took into consideration one of many scenarios in which chatbots could improve user experience and worked to implement a working software to demonstrate those improvements. The use case chosen is the one in which two people are using an instant messaging application to organize to go eating somewhere around or to go to the cinema for a movie. Looking at commonly used technologies, the average user would use a messaging app to communicate with its

friend, and he would switch to a browser or to other apps to gather the required information about restaurants or movies. A chatbot user wouldn't have to open other apps, but still would have to switch to an individual conversation with a chatbot and then he would share one link at a time with his mate. When ideating my software, and Android App, I decided to work on overcoming these drawbacks by having an assistant integrated in each user-to-user chat.

The actual implementation started from creating a messaging app from scratch and then building the natural language understanding system for the assistant to be integrated with the app. I used different software, platforms and frameworks to develop and test the app. The backend part is based on a MariaDB SQL database accessible via a PHP interface. It was initially deployed on a local XAMPP server and later on moved to a public domain using the Heroku platform. The Android client was coded using Android Studio as IDE, making use of many interesting classes to implement different features. It also uses some SDK libraries such as Facebook and Google ones. GitHub was used to version the code for backend and frontend. In the last chapter, we will go through different steps in the ideation, development, testing and evaluation, with particular interest on the architecture, the user interaction and relevant implementation details.

1 Chatbots: all you need to know

1.1 A brief history of Chatbots

Chatbot software has come a long way, from early computer programs like ELIZA to modern assistants like Alexa[52]. In the 60s, Joseph Weizenbaum at the MIT developed the first Natural Language Processing-based Conversational Agent. It was called ELIZA, and it was meant to demonstrate the superficiality of conversation between man and machine. Although it was based on a pattern matching system that substituted keywords in predefined sentences, many people were convinced of ELIZA's intelligence and understanding. The program was also one of the firsts to pass the Turing Test. The test involves two people and a computer running the tested software. One person types questions in a computer, receiving answers from either the other person or the software. The goal for the first person is to correctly recognize whether he is chatting with the human or the machine.

In the following years, other chatter bots were created for academic purposes. PARRY, developed at Stanford University, could add beliefs and judgments to conversations, showing progress with respect to ELIZA. An important incentive to chatbots improvement was granted from 1990 when the Loebner prize was launched. It is an annual competition still running nowadays, awarding prizes to the most human-like chatbot of the current year. Many computer projects were started exactly on this purpose like A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) and Albert One.

Definitely, developers began to move from academic projects to practical implication of chatbots in real life contexts[50]. The telecommunication is one of the areas that first witnessed the introduction of these technologies. As simple Automatic Speech Recognition functionalities were being developed in the 90s, companies began to integrate them in telephonic systems. Those NLU components performed basic operations such as recognizing isolated words to be matched to corresponding tasks and identifying digitital utterances. Early devices allowed the user to dial numbers by speaking and permitted call routing. Of relevant interest is the Voice Recognition Call Processing launched in March 1992 by AT&T. It could detect human speaking allowing commands like "I want to make a calling card call, please". The system was based on a vocabulary of five words only, and it permitted the automation of 3 million calls per day, with a success ratio of over 99

As chatbots grew in vocabularies and capabilities, they expanded to other fields. Many businesses decided to implement chatbots to provide real-time assistance to customers, without requiring the need of human intervention in a conversation. For example, US Sgt. Star is a chatbot in the Army's website

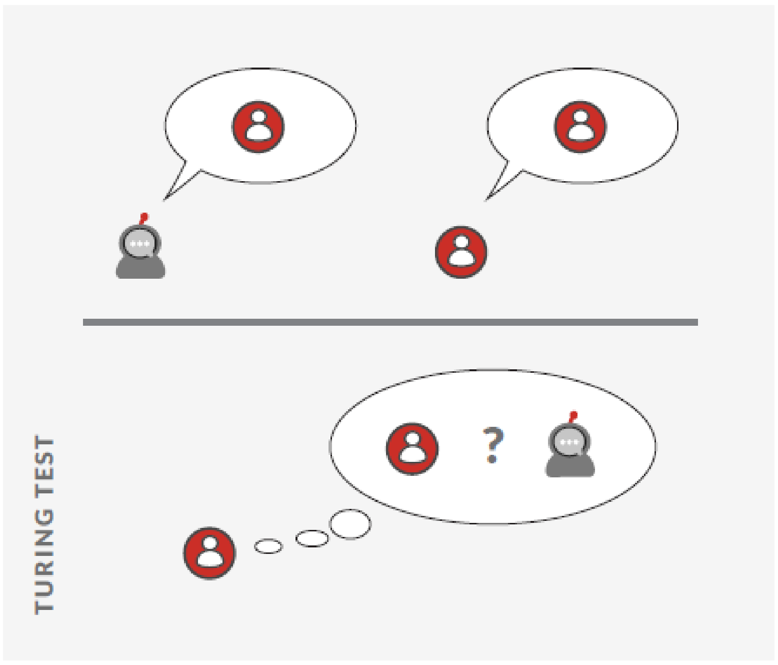


Fig. 1.1 – The Turing Test.

ready to answer common questions about joining the Army. In the same way, airlines companies have chatbots to process the purchase of tickets and to help users with general inquiries. Electronics brands can develop bots to help customers troubleshooting issues with devices and products, healthcare services can provide a "family doctor chatbot" for minor emergencies or first aid suggestions. Compared to humans, these machines are more reliable, quickly accessible, cheaper and easily updatable.

The latest, and currently state-of-the-art, applications of chatter bots are personal assistants. This kind of Conversational Agents goes deeper than providing or retrieving information from users. The challenge for this software is to have intelligent conversations with users, understanding necessities from any context and being able to perform multiple operations also interacting with different hardware and programs. At the same time, it should have the cleverness to learn from users, maintain memories, infer complex reasoning. Briefly, it should simulate a human brain[58]. All the major industries are putting huge effort and investing heavily in this direction, and although many of the mentioned capabilities are still very limited, great progress has been made in the last few years.

Finally, as mentioned in the summary, a shift in the interfaces paradigm is thought to continue in the next years. If that is the case, web searches based on keywords and indexing will slowly leave the path to voice queries build upon context and tasks. Software and smart devices will implement voice interface in addition to graphical interfaces on computer and smartphones, especially resulting in an improvement in control over IoT devices. Probably, in few years, people will be used to turn on and off light by voice, and buttons on the walls will disappear. Same thing will happen for appliances, heating and multimedia systems.

In this chapter, we will go into deeper details about chatbots applications, distinguishing among different categories and purposes and comparing examples of existing Conversational Agents and related software[78].

1.2 Different categories of Chatbot Technologies

Hundreds of thousands of chatbots are already out there. Some are very complex and are developed by the best teams of major tech companies, others are rudimental experiments created in few minutes by beginners[30].

There are different aspects that can be accounted to categorize bots. First, they can be classified into one or more domains, depending on their purpose[88]. Examples of domains are e-commerce, healthcare, entertainment, news gathering, navigation, etc. Bots also differ in the ways and goals for which users interact with them. That is the role they play into human-to-chatbot relations, such as shopping assistant, healthcare helper, professional assistant. Finally, chatbots have a set of actions or functionalities they can perform. Those skills represent what they can do to provide a service to users. Examples are recommending, shopping, dialoguing, playing music, teaching, etc. Some chatbots are designed for marketing purposes like helping a brand in reaching new customers, others provide support to already acquired customers. Some companies develop bots by themselves, others prefer to outsource the task to external vendors[59].

How do companies approach chatbots integration in their business? Most of them opt for the creation of chatbots deployable on messaging platforms such as Skype, Messenger, and Telegram but also Slack, Kik and WeChat. Among this kind of bots, the simpler ones tend to provide predefined labels for users to choose from, in order to guide them through the process flow. This option gives users the advantage of quick responses without typing and avoids out-of-domain queries. Cleverer chatbots allow users to write free-text sentences. Even if it makes the dialogue more human-like, bots often are not able to understand people's intentions. Some chatbots, usually the older ones, are offered to the user through website application. Obviously, ready-to-use mobile apps are preferable in comparison to browser-based platforms. In addition, World-leader industries like Microsoft and Apple are usually working on complex chatbots to embed on their Operative Systems or in standalone application and devices. Often, these bots have skills that span across various domain to accomplish different goals. They are called Intelligent Virtual Assistants.

In this section, examples for bots are presented to consider and explain the main domains of use, also pointing out roles and functionalities for each of them.

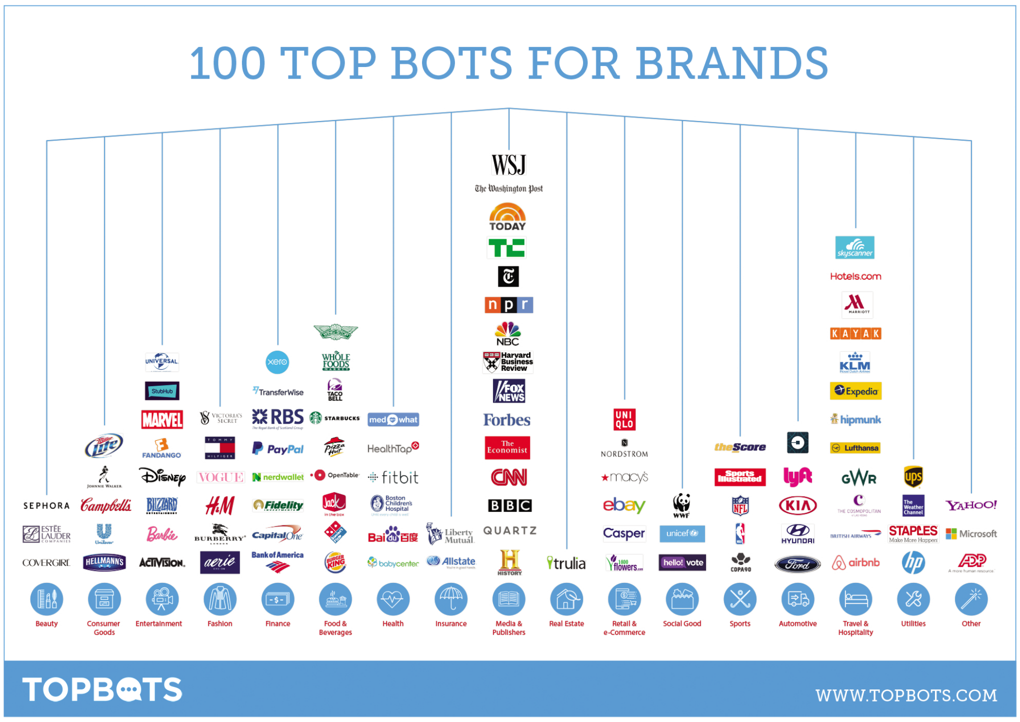


Fig. 1.2 – Top 100 Bots classified per domain.

1.2.1 Shopping and E-Commerce

The last decade witnessed a sharp increase in online commerce. People switched from visiting groceries and clothing shops to visiting websites to buy online. As we mentioned earlier, this is one of the fields that first saw the integration of chatbots, as it provided a direct income to software houses.

Many chatbots are available on messaging platforms. One of the best ones is chatShopper[2]. It is available on Facebook Messenger and was elected by users of chatbottle.co as the best chatbot for E-Commerce in 2017. Its role is "shopping assistant". To interact with it, the user can ask for a brand's product or for a specific category of items, such as "Lacoste Sneakers". The chatbot has the ability to recognize the specified product and look for it in featured shopping sites like Zalando. Then, it recommends items by providing links for additional details and giving the opportunity to add them to a personal wishlist. But what makes this bot emerge among competitors, is the integration of an image classifier. Users can send pictures as a message and have chatShopper recognize relevant items, looking for them in the same way as written queries. Brands like H&M, Victoria's Secret and Aerie also released chatbots to recommend customers about fashion styles.

A different thing is Amazon's Alexa, which must be taken into special consideration while approaching e-commerce domain[20]. As it is developed by the top retailer industry in the world, Alexa must excel in this task. And it does. It has more skills than any other shopping assistant. First, it can understand voice utterances in an almost flawless way through the Echo device. Moreover, it doesn't stop to recommending items, but it can directly purchase them. There is no need for external platforms to access details and insert payment details. However, Alexa is not only a shopping assistant. It is a Personal Assistant, and it has many more skills and functionalities ranging across many domains. It will be described in more details in the next section, in comparison with other similar chatbots.

1.2.2 Customer Care

Customer care, in many cases, can be collapsed to a classification problem such as labeled tree descent. As a problem and its relative solution are known, they can be inserted into the knowledge tree to be available for querying. Switching from human-based support to automated chatbot assistance can grant great advantages to companies. Costs are reduced by having a single instance of the app running to satisfy every customer, instead of investing time and money into training many employees. Moreover, it is easy to update and upgrade knowledge and NLU system of the service. Possible unknown requests can be redirected to a human employee to have better comprehension.

Let's dig into some real examples. Often, customer care domain is bonded with other domains, such as healthcare, travel or banking. We've previously mentioned Sgt Star providing information on behalf of the US Army. People can also meet Ann[14], a virtual assistant from Aetna, company offering insurance and healthcare services in the US. Ann helps users in accessing the website, registering or recovering lost passwords. Just the same support as the one obtained by calling customer services, but 24/7. This agent is browser-based and can be found on Aetna's website.

An example of customer services bound with travel assistance is the chatbot deployed by Icelandair[38]. The company, which flies people to and from Iceland, already has a fully operative chatbot on its Facebook page, reaching over 400,000 customers on Messenger. When the conversation begins, the agent can take two different roles upon request: "Flight Finder" and "Customer Assistant". If the option "Book a flight" is selected, it will guide the user through the research of specific flight opportunities and the successive reservation of the chosen one. Otherwise, the bot is ready to help the user providing assistance about a specific trip itinerary.

1.2.3 Banking and Finance

Kay[18] is a bank-account manager developed for Kasisto' customers. It provides a whole range of services to users. Kay can tell information about their accounts, help make payments, search for transactions and answer questions. The bot is available on different messaging platforms plus normal SMS. It's made of the latest artificial intelligence technology, allowing human-like conversations without awkward buttons or rigorous prompt syntax. Kasisto ensures privacy and security are their priorities[19]. The company affirms they are not storing any personal data except for name and email, granting secure transaction by encrypting data and allowing users to quickly block access to their account in case they lose their smartphone.

There are also personal solutions to manage spending and income information. Tiny Bank[3] is chatbot on Messenger that takes the role of "wallet manager". It that allows users to store transaction details about income and expenses, subdividing them into categories. They can even review their balance over time, asking the bot to plot graphs and charts for visual representation.

Bank of America unveiled in late 2016 their latest product, an artificial intelligence called Erica[81] built in a mobile app. Its purpose is to assist customers in improving their spending habits and making financial reviews based on their income and expenses.

1.2.4 Medicine and Healthcare

What if you are feeling sick but you don't have the time to visit a doctor? HealthTap[87] is there for you: a medic chatbot in your pocket. The chatbot is ready to analyze each case, pointing out what you might be suffering from and how to treat it. If that's not enough, users can submit questions via chat to more than 100,000 doctors in the U.S. or even get in contact by video. Developers also ensure that all information provided by users remains confidential and anonymous.

Florence[4], on the other way, is a Healthcare Assistant. It has many skills ready to help people. It can remind users to take pills or medicines at a fixed time and it is able to tell information about pathologies, cures, first aid, etc. Florence can also be used to track weight, heart rate or blood sugar, setting goals and checking reports to visualize progress.

1.2.5 Productivity

There are many ways in which bots can boost productivity. HpPrintBot[84], as the name says, is an assistant located between the user and the printer, providing high-level control on functionalities such as sending documents to the device and managing the printing queue.

X.ai[11] also belongs to the productivity domain, with its own agenda manager called Amy. The chatbot is able to schedule meetings for you and add them to your personal calendar. The interaction with Amy is made by email, and she takes care of organizing meetings with colleagues and friends based on the preferences you previously told her. Participants are invited formally and the personal calendar is updated upon acceptance.

Zoom.ai[49] is a professional agent able to provide a wider range of services, from scheduling meetings to setting reminders, from providing FAQs accessible to different departments of a company, to scheduling Ubers and flights. Its purpose is to improve employees' productivity minimizing distractions from operational tasks, in favor of higher-value labor.

1.2.6 News and Information

Most of the major newsagent companies decided to expand their channels of communication with chatbots. Information providers such as CNN, BBC, Fox, TechCrunch and many others are distributing their "News-provider" bots on popular platforms. Their goal is to facilitate how people access news from medias. Often, chatbots provide quick access to information prompting buttons and labels to choose from categories.

Some chatbots focus on sharing detailed information on a specific topic. At DoNotPay.co.uk, people can check out the first Robot Lawyer[10] to assist them with juridical inquiries. It can talk to users to understand their problems, answering questions and generating documents when needed. For instance, it can deal with compensation requests for flight delays and appeals for unfair parking tickets.

1.2.7 Food and Restaurants

This category may also be seen as a subdomain of the broader E-commerce domain, as people can order food online and have it shipped home. In 2016, Domino's Pizza launched their food-ordering chatbot, Dom[68]. Dom can chat with users on Messenger, listing the whole menu to receive pizza orders. It can also memorize recent preferences, enabling users to quickly reorder already bought pizzas. With the chatbot, it is also possible to track the current order, verifying its current state of delivery. Although payment is only available by cash, as Messenger released transaction capabilities in chatbots in late 2016, Domino's Pizza ensured they are currently working to integrate this functionality in Dom. Also Burger King, Starbucks and Taco Bell released their own food-ordering bots.

Other chatbots belonging to the Food and Restaurant domain have the purpose of finding, rating or suggesting places to eat. "Sure"[13], another Messenger chatbot, is your trusted restaurant advisor. Every time you feel hungry, just tell Sure what kind of food you would like to eat. It will select and suggest best restaurants for that category nearby your current location, based on the number of Instagram hashtags for that location.

1.2.8 Navigation and Travelling

Travel agents, also, decided to dive into chatbot communication. Skyscanner, one of the top flight-finders on the market, released a useful bot[74] for the most common instant messaging applications.

It is a "flight comparator", with the goal of checking best offers across different websites, returning best options to the user. The bot performs well when prompted with easy sentences and short replies, but can't really handle long, structured phrases. This applies to most chatbots out there: the state of current technologies isn't sufficient to provide a complete and complex experience to users, correctly handling requests and understanding context.

Other travel bots are also available on the market. Uber, a "Taxi Finder" makes it possible to call a taxi in any city where the service is provided. The chatbot[76] provides choices for different vehicles or drivers among which the users can choose.

To book for rooms or hotels, then, Expedia Bot is available. People can directly make a reservation, and in case a traveler should have additional requests, he can ask for human support directly from Skype or Amazon Echo.

Latest news are Alexa's skills from Hyundai[5] and Ford[8]. The companies developed bots for people to interact with their car from home to check its status and control it from remote. Voice commands allow users to lock, unlock, start and stop their car. They can also set the cabine temperature, check gas range, battery state and tire pressures.

1.2.9 Multimedia (Musics and Movies)

Many chatbots are ready to help out finding and playing multimedia contents. Xong[12] is the chatbot version of Shazam, a popular music recognizer app. It can recognize songs by receiving short audio snippets of the track. As a result, Xong will provide details about the song, as well as shortcuts to Youtube's video, Spotify and Apple Music items. Xong also manages a list of preferred songs, and let users share tracks with friends on Messenger.

If you are more interested in watching a good film, Movie Finder[7] is there for you. This chatbot is able to suggest movies based on IMDb ratings and categories. To do that, it will ask some questions to create a user profile, and then it will let him choose among many categories.

1.2.10 Entertainment

Some chatbots aren't meant to satisfy necessities or complete tasks. The first bots like Eliza and Parry had an academic purpose, and their only goal was to mimic human's ability to conversate. Agents of the same type have been developed since nowadays, some for academic purposes, others for pure entertainment. In 1998, Cleverbot[26] was released purely aiming at entertaining people through conversations. It is still active on its website and on other platform apps, and it held hundreds of millions of dialogues in its lifetime. Unlike some other agents, it directly learns from human input, finding keywords and responding based on searches on previously saved conversations.

On March 23, 2016, Microsoft also released its own AI chatbot, Tay[47], which stands for "thinking about you". It was modeled to mimic a 19 years old American girl, and it was deployed as a Twitter user, capable of reading and replying tweets and messages from other users. Tay could learn in an unsupervised way from those Twitter message. Sadly, its strength point soon became its main problem: Tay learned how to behave from racist and sexually-charged messages the Twitter user-base sent it. 16 hours and 96,000 tweets later, Microsoft suspended the account. A similar issue happened to IBM's Watson some years before when it started swearing after learning from Urban Dictionary.

Other chatbots are built to entertain people in different ways. Telegram saw a widespread in recent months of chatbots able to tell jokes, create funny GIFs or memes. Meme Creator is one of them. It lets users create their memes in seconds by sending an image and text to caption. The chatbot merges the content returning the edited picture.

Toys industries approached chatbots technologies too. Barbie gave life to its latest doll[15] so that children could talk to her.

1.3 Personal Assistants in depth

Personal Assistants, or Personal Agents, or again Virtual Assistants, are advanced chatbots with the purpose of helping users with a whole range of common tasks. Their final goal is to help users with tasks from different domains, replacing websites and applications in terms of rapidity and effort. If you are a global enterprise in technologies and computers, there are great chances you are developing your own personal assistant. Amazon, Apple, Facebook, Google, and Microsoft are there, followed by dozens of other companies. Amazon was the first to commercialize its Intelligent Assistant, Alexa, on large scale. Its Echo Dot smart speaker was the best-selling product on winter holidays, 2016, with around 6 million devices sold. While Alexa has been entering millions of houses all over the world, the other companies out there are trying to develop and release other solutions to compete with the previously unchallenged Amazon.

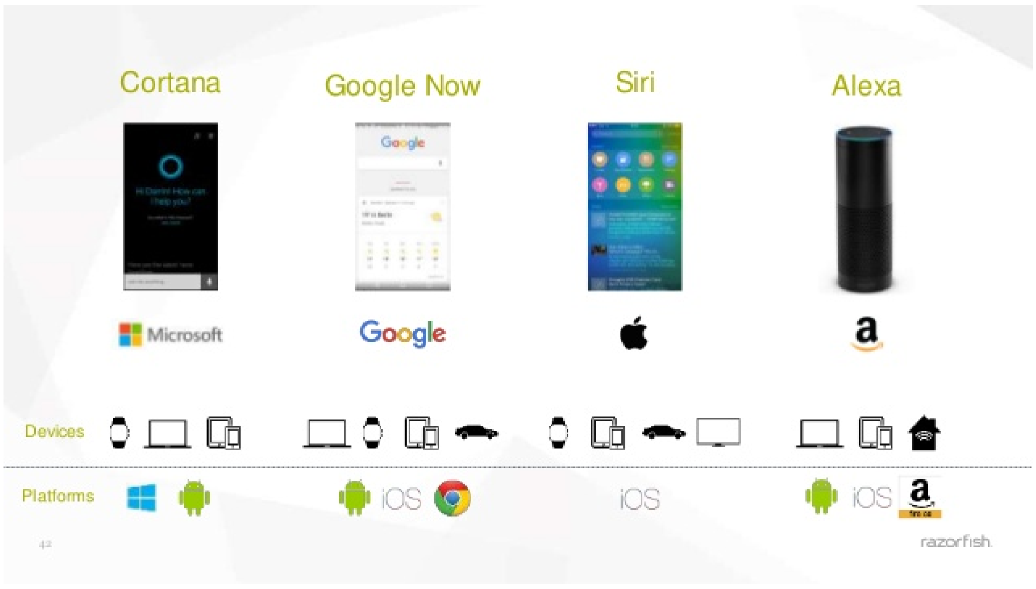


Fig. 1.3 – Which platforms and devices do popular Assistants support?

1.3.1 Main four competitors

Amazon Echo has been the first Virtual Assistant device to witness unprecedented global spread[86]. It was launched in 2014 and publicly released in June 2015 across the US. One of the reasons it became so popular, is that the company didn't promote it as a world-changer device, but rather as a speaker with some smart command on it. As people started buying it, Amazon worked on great improvements adding capabilities and more voice commands to the assistant. Moreover, Amazon gave the possibility to external companies to develop their own apps to be integrated with Alexa. They are called skills[53], and as of now, there are thousands of them available on the market. With the BBC Skill, people can quickly ask for daily news while having breakfast. Amex, from American Express, offers a new way to make payments, check the balance and much more with the power of voice. Developers can also access the new Smart Home Skill API to configure cloud-controlled services to adjust lighting and thermostat devices. In addition, Amazon provides the Alexa app for Android smartphones and iPhones. It makes it possible to connect to the Echo device from remote to set preferences and to call your Echo device which is at home. Users can also use the app to get a visual representation of the information provided vocally. This may be helpful, for example, to check paths and locations on Maps. Amazon is aiming for a deeper distribution of Alexa onto different software. At Consumer Electronic Show 2017, LG unveiled a new fridge[24] powered by Alexa. Other than common tasks like playing music and buying goods, it can reveal with a touchscreen display what is inside and warn if products are close to expiring. A more ambitious project is the one developed in collaboration with Ford. The two companies decided to include Alexa's cloud-based voice service into future vehicles[21]. This would be an important step into creating smart cars which, in addition to driving autonomously, would be controlled by voice. For the moment, the purpose of this upgrade is to help drivers staying more focused on driving, by providing hands-free interaction with the car for common tasks. Users can easily ask the car for some music or can set air-conditioning and GPS navigator. Moreover, the car can connect to you Echo device at home to turn on the lights or raise the thermostat temperature while you are away. On the other side, users can lock the car or start it wherever it is, directly from their home.

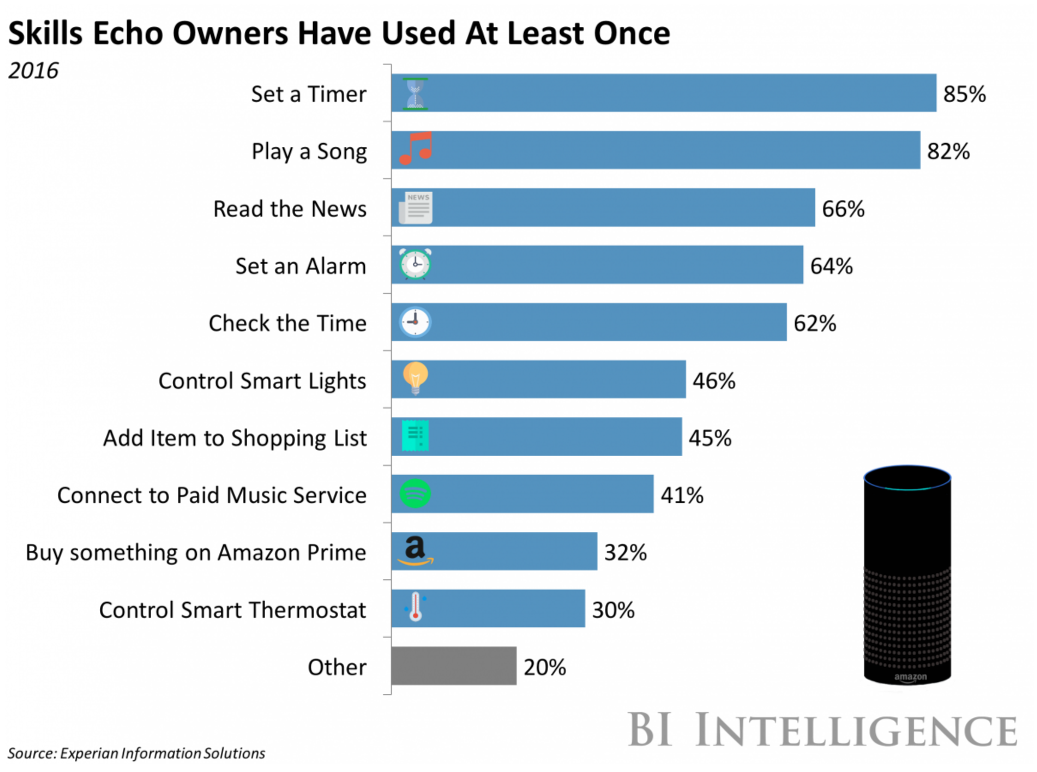


Fig. 1.4 – Alexa's most used Skills.

Google started its competition versus Amazon in November 2016, when it released Google Home device. It is a relatively similar piece of hardware that contains an improved Google Assistant[57]. Even though Google entered the market of personal assistants one year later, it can leverage on many strength points to overcome this gap. First, the searching company has the best machine learning background among competitors. Years of web search development improved the same tasks virtual assistants want to satisfy: providing answers to questions. Also, voice queries through Android devices took part into increasing Google knowledge about Voice understanding. Second, billions of Android devices could possibly be used to download the Google Assistant. For example, the company already integrated it in Google Allo[33] smart messaging app and into the new Google Pixel phone. Google Assistant is also supported on Tablets, Smart TVs and Android Wearable. Regarding the Home device, it supports external features like Alexa's Skills, with the name of Voice Actions. To conclude, Google is surely going to invest huge efforts into this area, as it will be the future of search tasks replacing web queries.

Microsoft also joined the race for Virtual Assistants. Unlike competitors, however, it decided to focus on computer platforms instead of mobile or external devices. Their virtual assistant is called Cortana, and it's preinstalled on every Windows 10 computer. The main reason for this choice is the fact that over 90% of computers worldwide run Windows OS[29]. One of the main drawbacks, however, is that the average user will not keep its PC on all day long. This means Cortana will not be able to compete against Amazon Echo and Google Home for interacting in a house context. In fact, Microsoft Managers stated they are working towards improving Cortana as a business, professional voice assistant. Their focus is on productivity, and as part of this objective, they are trying to integrate a lot of professional services under Cortana's supervision. Calendars and emails can be checked by linking Microsoft Exchange and Outlook software, while a deeper interaction and search with documents can be achieved with Microsoft Office. Finally, a big set of data has recently been obtained by acquiring LinkedIn in December 2016[66]. The transaction cost $26 billion and gives the opportunity to access a huge graph of information about network, careers, and professional interests of people. Again, this demonstrates Microsoft interest in business voice assistants. On the same trend, Microsoft announced the plan to expand Cortana onto Nissan and BMW cars and plans to preview its Connected Vehicle Platform in late 2017[85]. As General Manager Gavin Jancke said, the goal is that: "Your car becomes an extension of your office". Anyway, the company still aims to challenge competitors. For this purpose, it invested in many chatbots experiments such as China's Xiaoice. Xiaoice is a chatbot specialized in "chit-chats". That stands for random conversations without a real purpose but entertainment and fun, such as asking silly questions. Even though it would appear to have few practical implication, these studies help in creating more human-like and funny virtual assistants. In fact, many users stop using personal assistants because they feel too cold, mere machines. Other experiments have been carried on by Microsoft, like the xenophobic Tay and its better successor, Zo. For the moment, however, it feels like Microsoft is a bit behind competitors (as better explained in the next section), and Cortana acts more like a shortcut for Bing then a real assistant.

Among leading companies, Apple was the first to release its own virtual assistant on a large scale. Siri[45] was introduced on October 4, 2011. At first, Siri was acclaimed very positively for its ease and utility. At the time, people had never interacted with technologies so human-likely. However, users all over the world started complaining as many English accents weren't recognized by Siri. Apple went through many upgrades during the following years, and currently, more than 20 languages are supported. To compare it with other assistants: Google Assistant supports 7 languages, Cortana reaches 8 languages and Siri is only available in English, for the moment. Siri is available on most Apple devices: iPad, iPod, iPhone, but also Mac, AppleTV, and AppleWatch. Even though a great number of users can be reached in this way, using Siri is still an expensive choice, as Apple doesn't provide cheap devices like Amazon Echo and Google Home. Rumors say Apple is working on integrating Siri in a smart speaker, but no official statement has been made until now. If that's true, a great number of people could switch to this new device, as Amazon's and Google's speakers don't allow streaming of music for Apple users. At its Worldwide Developer Conference held on June 5th-9th, 2017, Apple publicly announced Apple Business Chat[22]. ABC is a platform to improve chat integration on iMessage with many new features, also from external services. For the moment, only developers are allowed to work on the platform for testing purposes, while the public released is stated to happen in 2018. At the conference, some video preview showcased many functionalities[80]. The customer can directly pay businesses with the Apple Pay integration and then send their location to instantly start shipment processes. They can also build models to analyze customer's messages. This could be used to save contents, interests, and orders mentioned in previous conversations improving the user experience in further dialogues. Other features include calendar integration, time pickers for appointments, list pickers to share options with customers, and QR code shortcuts. Apple also announced an association with LivePerson, Nuance, Genesys, and SalesForce to help companies in connecting Business Chat with their customer support.

1.3.2 Comparing Skills

Popular virtual assistants have many common things and some other different aspects. Is there a way to state which is the best? Probably not, as each one of them is built to satisfy slightly different requirements. However, we can try to list various popular tasks and see how the four assistants perform on each of them[60][65]. 13 categories are accounted for this evaluation: email, messaging, calendar, music, travel, weather, food delivery, sport, social, translation, basic tasks, general knowledge, and assistant personality.

When it comes to reading emails, Siri performed better than competitors. While all of them were able to send a new mail, only Siri could read the last mail. Google could only display it, Cortana did a Bing search and Alexa couldn't recognize the question.

With regards to messaging, Siri could both send messages and read inbox, while Google Assistant wasn't able to read messages. Cortana obviously can't access SMS on phones, just like Alexa. Surprisingly, only Alexa could read tweets on Twitter (again using a third-party Skill). The other three assistants are only capable of creating new tweets.

If the user asks for tomorrow's schedules, all personal assistants are able to answer correctly. Only Siri, however, understands the following connection with: "what about the rest of the week?". Again, Siri can smoothly insert new schedules and modify them, but the others struggle in modifying entries.

Alexa impressed when prompted with music requests. It could easily access Spotify, while the others wanted the user to continue on their own software: Groove Music, Apple Music, and Google

Play Music. "Play some new music" looked like a hard task for them. Alexa came closest again by reproducing music sorted by last songs added to the library; Siri just played the shuffled library; Cortana, again, went for Bing searches, while Google preferred "New" by Paul McCartney.

It's common to use phones to find best routes to some location, for example, "How do I get to Povo, Trento?". When asking each assistant to satisfy this task, Google Assistant automatically opened Google Maps with the correct path. Siri made it a little more verbose by requiring a vocal confirmation. Cortana did a Bing search for the path, while Alexa could only tell information without providing any map on the mobile app. Similarly, Google stood out among opponents when asked to book a particular flight. It provided the results (still not being able to book directly), while Siri and Cortana again performed Bing searches. Alexa can track and search flights using Kayak Skill.

All four virtual assistants scored well when interrogated about tomorrow's weather. Cortana and Google Assistant, however, seem to better keep context, in this case, being the only two able to get the follow-up question "What about Boston?".

Siri can make reservations at restaurants. The other three fell again into web searches. However, only Google Assistants could order delivery food.

The query "What's happening near me?" seems to be a tough one. Alexa, by downloading StubHub Skill, let you know about shows in the nearby. A similar content is provided from Cortana, with a widget for shows and sport events. Sadly, Apple and Google stuck to Bing Searches and Eventbrite shortcuts.

Translations are not easy tasks. Requesting for the Spanish equivalence to a sentence, Google Assistant and Cortana could help, and could even pronounce the translation, unlike Siri and Alexa.

Next up, the assistants were tested on basic functionalities. All four could open apps, set and cancel alarms and create reminders. The only lack was for Google Assistant, which couldn't change the screen timeout delay.

Then it was the time for some general knowledge questions. Here, Alexa performed the best. Cortana couldn't say aloud the distance from the sun to the earth but simply replied with a graphic, while Google Assistant couldn't answer the square root of pi. Upon asking "What's the news?", Alexa and Google Assistant replied flawlessly, while Siri surprisingly replied, "Here's some news for 'What's the news?'". Cortana, on the other side, gave the dictionary definition of "news".

Finally, their personality was put to test. All of them could tell "relatively funny" jokes. When they were asked to play some games, Siri and Alexa couldn't help. Cortana seemed to only know a game about finding names of films after being given some clues. Google Assistant won here, offering various quiz games, Rubik's Cube and more.

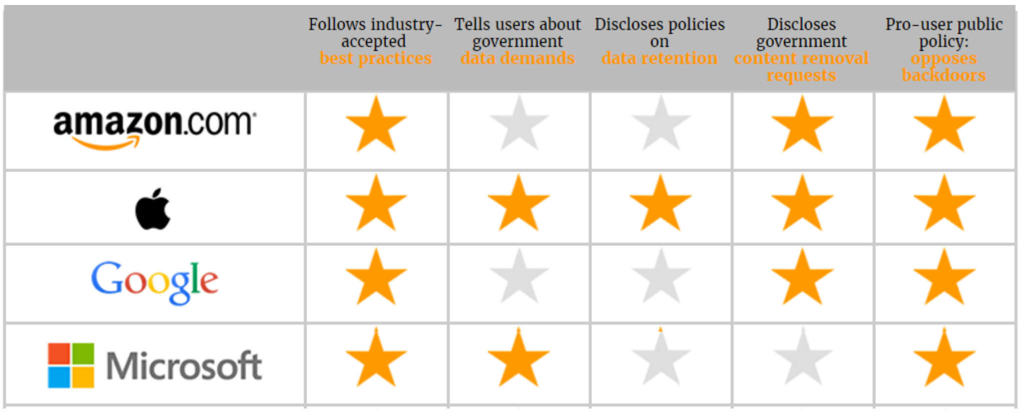
To conclude, it is obvious that Virtual Assistants are pretty far from perfection. Maybe, the one with higher potentiality could be Google Assistant. For the moment, it can satisfy a higher number of requests than the others, closely followed by Siri. Alexa is good for some tasks, like playing music, shopping online and controlling home devices. Cortana, simply, is a Bing shortcut, but Microsoft Managers are aware of this and say Cortana is just scratching the surface of its future potentialities. Anyway, each of them is poorly connected to some services that are strictly necessary, often resulting in web searches for unknown queries (especially with Cortana's annoying Bing searches). Furthermore, many operations are slow and require many passages, making them slower than browser-based ones. In addition, it is very easy that misplaced words or badly pronounced utterances lead to complete incomprehension from the software. Finally, Assistants usage imply many Privacy issues that will be deeply discussed in the next section.

1.3.3 Privacy and Security issues

Personal Assistants are designed to learn about our interests and preferences. That's their nature. Without saving information about us, they couldn't provide customized experiences to users. The problem is to define when, what, and how do they stop listening to users. To provide best performances, all the agents need to be constantly awake, so that they can listen to utterances and understand when users are calling for their help. The problem is Natural Language Understanding Systems are located in the Cloud. Although companies state all the information is treated anonymously, there's no evidence they won't store personal data for aimed advertising or worse purposes. What if a user reads aloud sensible information like credit cards details? Let's see in detail what the major tech industries have to say about this topic, and how do can people deal with this relevant issue.

Under the Privacy Policy section of Microsoft's website[28], the company states all the data collected by Cortana is stored on the cloud to grant the best user experience, but none of them is used to target ads to users. Microsoft distinguishes between signed in and signed out users. When a person is signed out on the device, Cortana collects information about speech to improve their speech recognition services. It also collects the history of searches and data about the device, such as status, settings, and installed apps. If a user is logged in, a profile is kept with personal details plus information from other apps, such as songs on Groove Music or favorite places on Maps. A Notebook is also available and shared real-time with all connected devices. By default, Cortana stops here. However, users can give permissions to other information to personalize even more the experience (and, of course, send even more data to Microsoft). They can approve location access to get contextual news based on actual position. It is also possible to allow Cortana to read messages, email and calendar appointments to set reminders and quickly save new events. Even though it's possible to manually tune all the permissions, having all of them set to off would just make Cortana a Bing 2.0. A positive aspect is that Microsoft gives the possibility to users to delete all information previously connected to themselves. Yet, it is unknown if data is entirely wiped out, or if is just disconnected to personal details, but still linkable to the original user trough some machine learning.

There are concerns about how Echo accesses conversations in the home. It may be also possible to trace who is at home by matching footstep sounds or music playing, once Alexa has a detailed profile of inhabitants. Amazon, like Microsoft, stated the device only saves speeches uttered after the "Wake Word" and store them to improve its assistants' abilities[16]. Again, like Cortana's users, people can tune permissions for the location from the app. It is also possible to retrieve the full list of voice recordings stored in the past and manually delete each of them. There have been many cases in which Echo was used as crime evidence. The company fielded 813 subpoenas for information about its customers, and in 542 cases, it gave all details requested by a legal process. The same trend applies to 25 search warrants and 13 court orders.



	Follows industry-accepted best practices	Tells users about government data demands	Discloses policies on data retention	Discloses government content removal requests	Pro-user public policy: opposes backdoors
amazon.com	★	☆	☆	★	★
	★	★	★	★	★
Google	★	☆	☆	★	★
Microsoft	★	★	☆	☆	★

Fig. 1.5 – "Who has your back?" Protecting your data from government requests

Regarding this topic, the Electronic Frontier Foundation documents the practices of major Internet companies and service providers, judging their public policies and highlighting best practices[67]. In recent years, companies started sharing annual reports about government data requests, like the one mentioned above about Amazon. The chart depicted illustrates behavior in five main situations, and the only company reaching five stars out of five is Apple, while the others just achieved three. Also looking at Siri's Privacy Policies, it's evident as Apple greatly overcomes competitors in security[46]. What you pronounce is paired with a random identifier associated with your device, while all of your personal data isn't connected to any voice recordings. Users can also reset identifiers to start a new Siri experience without personalization, or they can bind identifiers to other devices such as Apple Watch. Messages, emails, and calendar stay on your phone locally, nothing is stored in the Cloud.

When some content like event's location or user's position needs to be shared with external servers, they are protected by rotating identifiers that can't be matched to your account.

Google Assistant is available both on Google Home and Android devices. By default, on smartphones, it has many privileges. It can see contacts, calendar, and storage in general, plus all the information saved on the Google Account. That's a pretty deep knowledge of the user! Upon permission request, Google Assistant can see what is on your screen plus the whole history of locations[34]. On Google Home, things are slightly different[35]. Data collection isn't permitted by default, but by granting it, people let Google share it with third party services by their own privacy policies. Initially, Google stated that that information would be stored only for a short time on data-centers and then deleted, but apparently, they changed their mind. Moreover, even though users can delete history through the website, some service-related information is kept to "prevent spam and improve services". On the positive side, Google grants that its Assistant doesn't record all conversations. It only listens for few seconds to catch the hot word "Ok Google". Those snippets are deleted right after unsuccessful detection. When the hot word is recognized, the colored LEDs on top of Google Home light up, telling the users it started recording.

To sum up, Personal Assistants have much space of improvement. On one side, they need to learn to keep track of complex contexts, to be more flexible in understanding utterances and to connect in better way services and devices. On the other side, they must grant privacy and security to users, with transparency on how personal data is managed and preventing unauthorized access from the web. The goal is to improve and balance all those aspects to provide a new functional and safe way to live technologies.

1.3.4 Other Personal Assistants

Other than the four Personal Agents discussed in the previous section, there are tens of projects already on the market or in development, each with its own goals and strengths.

Among all of them, some emerge for special features they pack. In 2015, Facebook announced it was working on its own Virtual Assistant, M. Despite other major companies heavily used machine learning techniques for their assistants, Facebook proceeded in a different way. In fact, M was trained combining machine learning and human problem-solving activities[71]. The company allowed few people to use M for testing. As the users messaged their requests, the AI tried to parse them and answer based on its knowledge. When M wasn't able to catch the question, it fell back to a team of humans which manually replied the users. By doing so, M was learning how to deal with the processed query, thus being able to satisfy it the next time. Facebook proceeded this way for a couple of years, coming up with an improved version of their assistant. In April 2017, the first version of M was released publicly in the US as "M Suggestions" part of Facebook Messenger[64]. In contrast with popular assistants, M Suggestions is meant to be added in existing chats to help people chatting (as its name "suggests"), instead of existing in individual human-to-chatbot conversations. When the user is messaging with a friend, the text is parsed by the AI which comes up with emoji, GIFs, quick-links or information, depending on the intent detected. For example, saying "pay me 20€" will pop up a link to a payment functionality built in Messenger. Although Facebook is starting to connect the app to third party services such as Uber and Lyft, they agree M is still in an early phase and is missing many relevant functionalities on which developers are working. That is evident when M is compared to Allo's Google Assistant. One of the features it could integrate is the one of suggesting other chatbots be added to the conversation. The company, in fact, has recently announced major improvements on their chatbot platform on Messenger, such as the option to add them in human-to-human or group chats[63]. Anyway, updates have been promised by Facebook for the next months, when hopefully people from other regions will be allowed to upgrade to M.

Going back to 2010, Apple acquired Siri, maker of the popular Virtual Assistant and founded (among the others) by Dag Kittlaus, Chris Brigham, and Adam Cheyer. Two years later, the trio left Apple to start developing a new Assistant called Viv, something like a Siri 2.0. In May 2016, the potentialities of their new agent were demonstrated in a world premiere, at a TechCrunch conference. Again, their company was acquired, this time by Samsung, with a transaction of over $200 million[75]. For the moment, Samsung devices offer Bixby assistant, but in further versions, Viv will take an important part in improving the company's personal agent. Compared to its "predecessor" Siri, Viv platform's is open to external plug-in to easily interact with the assistant. It is also able to handle more complex queries[79]. How did they make it possible? In their World Premiere, the founders showed how Viv works. Its way of reasoning, they say, is something different from other assistants. For each problem presented, the AI writes the program to find the best solution. It is like it generates automatically its own code. Moreover, the assistant combines the results for any query with a personal database containing specific information for every user. If someone asks: "What's the best available seat on Virgin 351 next Wednesday?", Viv not only searches in an external service (Travelport) available seats, but it combines them with a backend (SeatGuru.com) that provides information on seats for each plane, and finally, it filters results for personal preferences, like aisle seats or extra legroom. Because Samsung owns a relevant slice the smartphone market, introducing a new assistant could be a game-changing outbreak in the competition for the best Personal Assistant, also because it will directly face Google Assistant on the same devices.

The third and last Virtual Assistant took into consideration for its originality is Lucida[83]. It is a project founded with the name of "Sirius" by the University of Michigan and has the interesting feature of being Open Source and prone to modularity. In the crowd of private agents owned by big companies, it is unclear how data is stored and elaborated. With that in mind, Sirius project began as a research prototype on hardware platforms for Intelligent Personal Assistants. Its initial purpose was to facilitate benchmarking and crafting highly optimized algorithms to handle huge workload of Cloud Platforms. Those datacenters, in fact, are reaching their computational limits due to the huge tasks carried on by Machine Learning services. As this goal was achieved, the project evolved into Lucida, an AI aiming to modularity and extensibility. That was pursued by creating open source modules for main Personal Assistants features, giving developers the opportunity to modify or replace them. Those core functionalities are the Automatic Speech Recognition (ASR), Image Matching (IMM), and Question-Answering System (QA). By default, Lucida can understand in queries in the form of speech and images and answer them in natural language. For example, when it is presented with a picture of the Leaning Tower of Pisa and questioned about its height, the Assistant can identify the image, understand the request and retrieve the answer from its backend. What if a researcher develops a new DNN for Image Recognition trained on a particular category of images (e.g. a car model recognizer)? He can now just replace Lucida's IMM component with his own, taking advantage of the rest of backend services provided while improving the assistant's abilities on that specific task. From September 2016, Lucida is available as an open platform on the web and source code on GitHub[41].

1.4 Platforms to develop basic chatbots

Until now, a detailed discussion has been carried on regarding cutting edge chatbots like personal assistants. Obviously, they are developed by a large team of experts and rarely (like for Lucida), projects are open source and usable by other developers. What are, then, the available solutions for individuals or small teams to quickly develop chatbots? Luckily, many platforms are available to help to create simple bots in a short time. The outcome will not be that performant, but sometimes it will be enough to accomplish simple tasks. Most of the chatbots available on Telegram, Messenger, and Skype are built in similar ways. There are a lot of framework and websites to fulfill this task. In this paper, a comparison is done among four of the most popular services available: Wit.ai, Api.ai, Watson, and LUIS.ai[51][55].

1.4.1 Wit.ai from Facebook

In April 2016, Facebook released Facebook Bot Engine. It is a wrapper built to deploy bots on Messenger, and it is based on Wit.ai technology. Wit.ai is a startup created in 2013 and acquired two years later by Zuckerberg's company[62]. The platform they developed in these years provides a server on the cloud when developers can create their own apps for basic Natural Language Processing tasks. The app receives requests as textual strings or voice utterances and proceeds in extracting entities, intents, and sentiments from them. Users can adopt predefined entities such as date, time, phone, greetings, or create their own ones. User-defined entities are based on keywords recognition. They can be input in the app as a list, or users can provide free text samples with expected results. On this labeled dataset, Wit.ai applies machine learning techniques to learn to extrapolate relevant intents and entities. This technique is more performant in the case of large numbers of detectable features but also requires larger training datasets to for more precise results. To interact with the app from a frontend or a server, developers just need to send to the app endpoint HTTP requests (GET or POST) carrying the message as payload and specifying the private app key as an additional parameter. Wit.ai provides extensive documentation of easy comprehension[1]. I used a Wit.ai to develop SmartChats, and in the next chapter, I will discuss in more details how I created the NLP on the platform.

1.4.2 Api.ai from Google

In late 2016, Google acquired Api.ai[70]. Same story as Facebook and Wit.ai, every tech company wants its own chatbot platform. It is based on entities and intents and works similarly to Wit.ai. However, Google's platform provides a wider range of option to integrate the app into other services[23]. The app can be set to call external webhooks when a certain intent is matched, to get the additional content and return a complete answer to the users. For example, in few minutes anyone can configure a simple service to provide weather information defining the intent and correlated entities such as time and place, plus an external backend to get the requested forecasts. Moreover, Api.ai let users easily integrate the bot in Messenger, Skype, Slack, Google Assistant (via Actions on Google), Telegram, and also Cortana and Alexa. Finally, it offers a useful set of Prebuilt Agents such as vehicle interactions, flight, hotel and restaurant reservation, language translation and traffic information. Each one of theme comes with a complete structure of entities and intents. Initially, I began working at an Api.ai app as NLP service for my Android App. The problem was some features I needed were going to be deprecated or modified soon as part Google's acquisition process. For this reason, I decided to switch to Wit.ai.

1.4.3 Watson from IBM

Watson is an artificial intelligence in development by IBM since 2005. Its initial goal was to beat champions of the quiz show "Jeopardy!". In order to do this, the bot would need to understand questions and fetch answers in a matter of seconds, something impossible to do at those times. Three years later Watson reached its objective[73]. IBM developers, however, continued to carry on the project with the goal of "having computers interacting with humans across a range of applications". In 2013, IBM made Watson's API available to software providers to build apps with its capabilities. Those features not only include NLP functionalities (Watson Conversation) like Facebook and Google counterparties, but also many other powerful functionalities such as tone and sentiment analyzers (Watson Alchemy Language) together with high-level and abstract concepts detection[6]. All services as provided on IBM Bluemix cloud. Watson is targeted to existing enterprises, as developers need to pay to use Watson tools.

1.4.4 LUIS.ai from Microsoft

Microsoft Language Understanding Intelligent Service (LUIS)[40], was introduced in 2016, together with Microsoft Bot Framework[43] and Skype Bots. LUIS.ai can be compared to Wit.ai, being more limited in capabilities than Api.ai. Currently, users are limited to using at most ten entities per application. Along with other platforms, is based on entities and intents, providing some predefined ones, and is available as an app on the web with its own endpoints. LUIS apps can be easily integrated via Microsoft Bot Framework by deploying them in .NET, Node.js or Azure Bot Services.

1.5 To sum up: Today's Chatbots in a nutshell

As we moved through this chapter, it emerged as the current state-of-the-art of chatbot technologies, although improving at a fast pace, still offers limited possibilities. Bots on messaging applications struggle to converse without falling into default "I couldn't catch your intentions" dead ends. The platforms available to the developer community are not enough to build artificial intelligences able to provide a captivating experience to users. That is because the resulting bots are only able to follow if-then-else flows in conversations and their knowledge is based on basic keyword spotting or pattern matching. Therefore, many follow-up questions such as "What about Milan?" after the query "What's the weather in Rome?" are missed by AIs. Human dialogue cannot exist without a minimum of context understanding and memorization[77]. This implies being able to recognize when and how to communicate with the user, lowering the volume when he is in the library or postponing the communication when the user is speaking to someone else.

Then, being that chatbots aim to act like humans, one of their greatest limitations is the inability to perceive sentiments and moods. They must start handling emotions to have a deeper understanding of their input and a better way to express responses. Finally, while few tasks such e-commerce and restaurant reservations have been studied and implemented thoroughly in recent years, a great vastness of functionalities is still not caught by chatbots. And because of the incredible complexity of human cognition, developers should find more performant ways to create bots that learn to learn by themselves.

2 SmartChats, a recommender chatbot inside an Instant Messaging App

What is SmartChats? SmartChats is a mobile application that I developed across April and May 2017. Its purpose is to study, implement and simulate how Chatbots and Virtual Assistants can merge into existing platforms such as instant messaging apps to improve the user experience in performing common tasks. In this particular case, the role of the chatbot is to recommend movies and restaurants to users in a smart and non-invasive way. The app is developed for Android OS.

2.1 Use case scenarios and Initial Ideas

How can a chatbot help people by suggesting restaurant ad movie screenings? Let's dig into the current ways of organizing to meet up for a dinner or to go to the cinema. Usually, people would start by launching a mobile app to text to his friend. Next, he would need to switch to external apps or a browser to get information about restaurants (and in similar way movies) nearby. Then, he should get back to the messaging app, providing a "meaningless" link to his friend, or just naming the place. The other participant, again, would look for more information on the web to confirm the choice or propose an alternative. It would continue this way until a decision would be taken. While existing chatbots provide, sometimes, a quicker way to get information without switching application, they force you to switch among conversations, and responses are not directly available to both users unless they are shared. Returning to the initial question, SmartChats leverage on removing the effort needed to take and share information with people. The software combines the pros of chatbots with the ones of instant messaging apps. On one side, information is provided quickly just by means of Natural Language Understanding. On the other side, messaging apps let users stay in touch with their friends in real time. The combination of both would be a chatbot integrated into messaging chats between people. The chatbot would be able to detect requests and suggesting information directly to both participants.

Some specific scenarios were plotted and studied to simulate how the app would work. One of them was also used for user-testing purposes, as explained in the following section about Testing and Evaluation. The written plot can be found in Appendix A.4.2. The scenario was also simulated and recorded to show off how users can interact with the app. The video is available at `https://youtu.be/h46r0S0rxdA?t=47s`. Briefly, all the scenarios involve two people chatting on their smartphones. What is common among them is how the suggestion feature built in the app works. The suggestion is only triggered when messages refer to restaurants or movies. For example, when the message "Would you like to hang out for a pizza?" is sent, the NLP system detects that the context of the conversation has changed to "restaurant domain". As a result, the app will come up with some suggestions regarding the food domain. What differs among the various scenarios depicted is how, when and where users are interacting with the app. Let's see some examples. Looking at the location, users can be in the same city, or in different places, they can be looking for close restaurants or restaurants in different regions. Users can be interested in movie shows aired tonight at the cinema, or in the next days, weeks, months. They may want to compare screenings across different dates. They can be typing in different languages (Italian or English, for example). Users may talk explicitly about restaurants, or they can be referring to pizzas, hamburgers, lunch, dinner, or generally speaking about "eating something". Users may want to receive more information about the restaurant on its website or plan a trip using Google Maps. They may also be interested in contacting the owners by phone or email. In addition, users may want to easily share the relevant information with their friend. Those are many scenarios and actions I thought about when planning SmartChats functionalities.

Once the main idea was defined, I started tackling key aspects of usage, to improve the user experience while thinking of different scenarios. In this paragraph, I will explain how I identified and solved relevant issues about user interaction with the chatbot, finally coming up with the actual design and workflow realized for SmartChats. Firstly, I thought about having a chatbot placed as a middle layer between conversations. It should have been able to detect both users' preferences, resulting in coherent suggestions. However, this implementation would have destroyed the human-to-human conversation. To have a more spontaneous experience, I chose to keep conversations based on normal user-to-user chats. On top of them, a chatbot would control and recognize user-intents, eventually intervening to provide suggestions. The problem with having a chatbot inside a conversation is that it should know when it is its turn to talk. Obviously, if the chatbot would suggest something every time it detects an intent, the chat would have been filled with spam. A way to tackle this problem could have been requesting its intervention with keywords or special commands such as "/suggest". I preferred to implement it in another way to speed up the interaction. When an intent is detected in a message, a non-invasive button appears aside the text, making it quicker to require suggestions instead of typing the command. Once more, I had to think of how to present the recommendation to users. Common chatbots show results as a list of images plus descriptions and shortcuts to actions. I thought that it would be better to give the chance to the user to interact in a more complex way with the results, for example ordering them in another way, repeating the query for another location or day, etc. For this reason, the suggested locations or movies are shown in another full-screen window. Even though suggestion results aren't inserted directly in the chat as normal chatbots do, they are always available from the non-invasive button. Also, if a user finds an item especially interesting, he can share it with its friend as a permanent message in the chat. These are the core functionalities I thought before developing the app. A detailed explanation along with screenshots and use flow is provided in successive sections.

2.2 Related Work

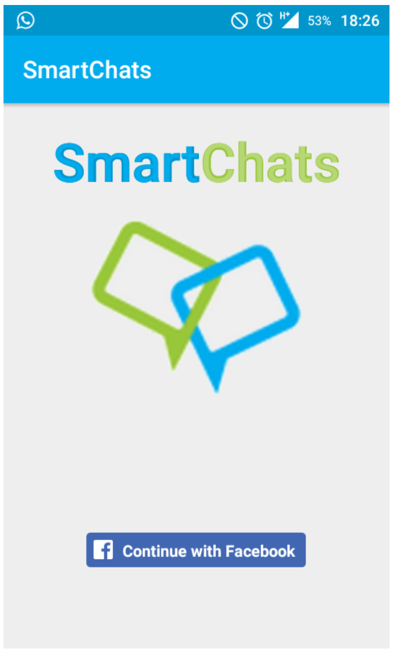
While many companies such as Facebook, Skype, WeChat, and Telegram are working towards a deeper integration of chatbots into their messaging platforms[61], existing bots are still bounded to a human-to-machine relation with the user. In April 2017, Facebook introduced Chat Extensions[56], that allows the user to call bots' functionalities directly from existing conversations. The only app existing on the market that provides the functionalities implemented by SmartChats is Google Allo. Google Allo[33] was released in September 2016, and it is an instant messaging app featuring Google Assistant AI. Users can directly type to the assistant or request its help from other conversations, with the command "@Google". Google Allo has many of the most important features available on other Messaging Apps, and it has all the requirements to deal with competitors. Facebook's M Suggestions[71] has, also, a word on satisfying SmartChats' goals. Although the software is limited to users from the US only, it can similarly join existing chats to help out users with its services. In the next months, Facebook developers should implement the integration of external services in M Suggestions. This could dramatically improve M's capabilities into Messenger, allowing it to contest Google Allo as the smartest messaging platform available.

2.3 User Manual

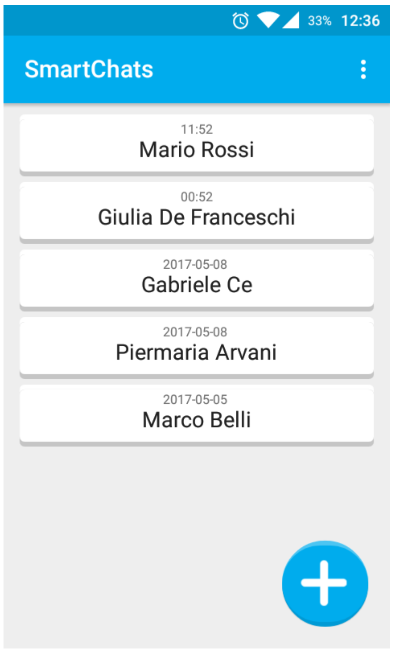
The app is not available through the Play Store, but it can be installed from a lightweight APK file[54]. To install the app, permissions to use Internet and GPS Location (not compulsory) are requested. When SmartChats is launched for the first time, it shows up the Login/Registration screen (See Figure 2.1(a)). The user needs to login with a Facebook account to be able to use the app. SmartChats asks Facebook permissions to retrieve user's email, name and friend list. After the first login on a device, further logins are automatic, directly loading the homepage.

In the Homepage (See Figure 2.1(b)), the user is shown a list of its existing chats sorted in inverse chronological order, from the most recent to the older. To open a chat the user has to tap on its name. He can also create a new chat with a friend by clicking the "+" button on the bottom right of the screen. The user can log out at any time from a device from the homepage's menu. Chats and information about the account are still backed up on the server, and the user can access from another device without losing any personal data. In the menu, there is also the option to get help in interacting with the GUI. A 3-seconds-lasting message will appear on the screen, informing the user about what he can do in the current activity. The "help" option in the menu is available in any activity in SmartChats, except for the Login one.

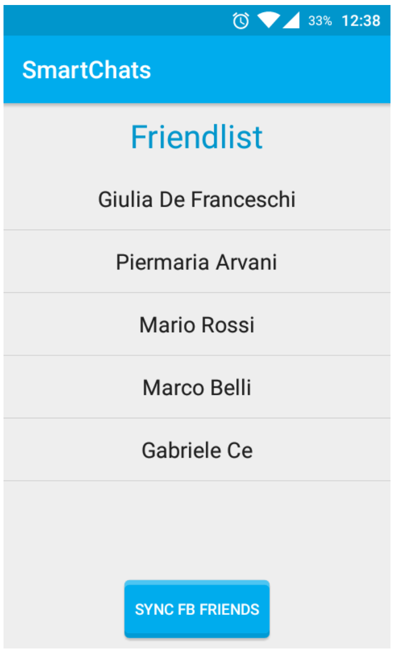
When the user clicks on the "+" button, he will see his SmartChats friendlist (See Figure 2.1(c)). This list contains all Facebook friends which installed the app. It can be synchronized manually with the "Sync" button, to check if any friend has recently installed the app. When a friend is chosen by tapping on it, its chat will load on the screen. If the user and his friend have never chatted before, the chat will be created and will now be visible from the chat list on the Homepage.



(a) Login Screen



(b) Homepage



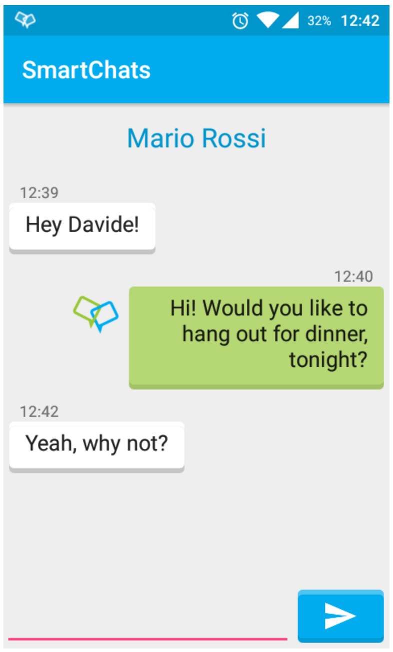
(c) Friend List

The chat activity (See Figure 2.1(d)) shows up a list of messages between two users ordered by time. User's messages are shown on the right inside a green bubble, while his friend's messages are left-aligned on a white background. Both users can tap on a message to see at which time it was sent. At the top of the page stands the name of the correspondant for the current chat. At the bottom, the user can type a message in an empty field and send it with the button next to it. Users can not send empty messages. When a message is sent, a push notification is created and received by the target user. He will see it at the top of the screen represented by SmartChats' icon and containing a preview of the message's content. Tapping on the notification will launch the app. When a notification is received while the user has already the chat activity open, it will refresh to automatically collect and show the new message.

Now, let's see how SmartChats' assistant is triggered while chatting. When a message is sent, the app analyzes its content to look for references to cinema and restaurants activities. If one of them is recognized, a small button with SmartChats' logo will appear aside that message. The user can click that button at any time to access the Suggestion Activity related to the detected task.

The suggestion page differs between restaurant and cinema intents. If a restaurant intent is detected, the Suggestion Activity will contain a list of restaurants (See Figure 2.1(e)), each of them with a map preview of its location and the address. Here, the user can quickly find closest restaurants by clicking on the GPS button. If localization is already enabled for the device, the suggestion page will sort restaurants based on their distance to the current location. The current address will be printed out in the text field on top of the page and the distance in km will also appear aside each restaurant's address. If the device doesn't have GPS enabled, the app will open up device settings to turn it on. To prevent the user from draining its smartphone battery, SmartChats tells the user when the current location is detected so that it can turn off location. While developing the app, I noticed a problem in the workflow of finding an interesting restaurant. What if a user is in Milan and would like to find places near its home in Trento? As an answer to this question, I added another way to find restaurants. The user can manually type an address in the text field (which by default is set to Trento). The app automatically matches the location to its GPS coordinates, repeating the sorting query for that position. The user, finally, can tap on the map preview to open the location on Google Maps, or he can click on the name/address to load a detailed Restaurant Page.

What if the cinema intent is detected? The Suggestion Page for Movies (See Figure 2.1(f)) will show a list of movies which are available at the cinemas for the current date. Each movie is described by its poster, name, category, length and a short description of the plot. Again, when a movie item is chosen from the list, the Movie Page with more details will pop up.



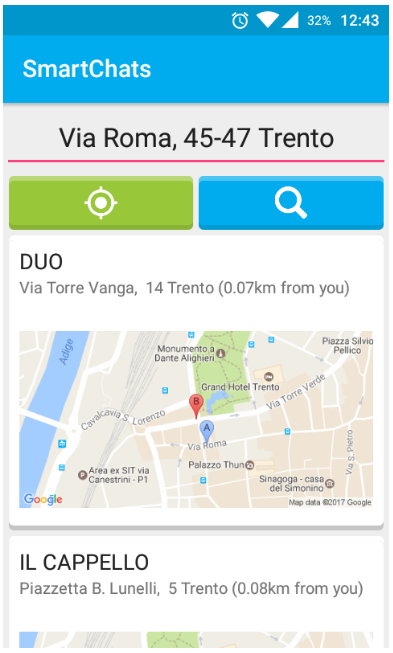
Mario Rossi

12:39
Hey Davide!

12:40
Hi! Would you like to hang out for dinner, tonight?

12:42
Yeah, why not?

(d) Chat Screen

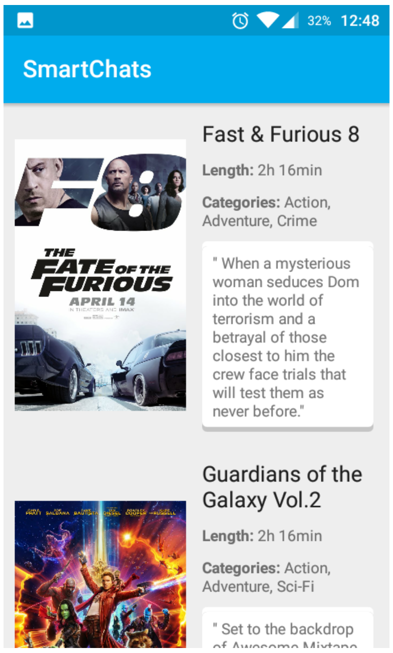


Via Roma, 45-47 Trento

DUO
Via Torre Vanga, 14 Trento (0.07km from you)

IL CAPPELLO
Piazzetta B. Lunelli, 5 Trento (0.08km from you)

(e) Restaurant Suggestions screen



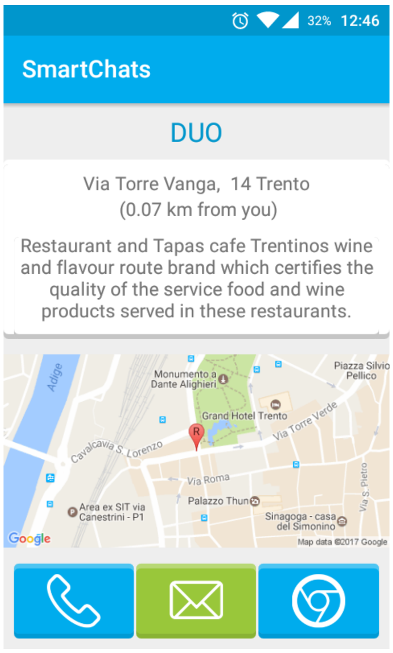
Fast & Furious 8
Length: 2h 16min
Categories: Action, Adventure, Crime
" When a mysterious woman seduces Dom into the world of terrorism and a betrayal of those closest to him the crew face trials that will test them as never before."

Guardians of the Galaxy Vol.2
Length: 2h 16min
Categories: Action, Adventure, Sci-Fi
" Set to the backdrop

(f) Movie Suggestions screen

The Restaurant Page (See Figure 2.1(g)) contains many information and shortcuts for the chosen restaurant. Its name, address, distance and some lines of descriptions are available on top. Beneath, a larger preview of the map is a shortcut to launch Google Maps application. At the bottom of this page, three more buttons are available. By clicking the first one, the phone dealer will open, containing the phone number to contact the restaurant. The second button allows the user to send an email through an installed app (such as Gmail) to the restaurant's customer service. The last button leads to the website page of the restaurant, loaded on the default browser. Finally, the user can share the restaurant as a message to its friend. To do this, he needs to swipe up on the screen. The suggestion message (See Figure 2.1(h)) contains an invitation to check out the restaurant, as long with its name, address and map preview. Tapping on this suggestion message will open up the relative Restaurant Page (here, further sharing by swiping up is prevented).

In the same way as restaurant's case, the Movie Page (See Figure 2.1(i)) shows more information about the chosen film. Other than the content present on the suggestion page, screenings for the

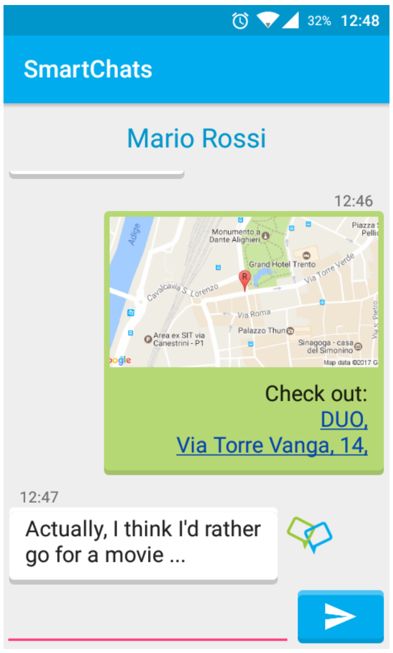


DUO

Via Torre Vanga, 14 Trento
(0.07 km from you)

Restaurant and Tapas cafe Trentinos wine and flavour route brand which certifies the quality of the service food and wine products served in these restaurants.

(g) Restaurant Page



Mario Rossi

12:46

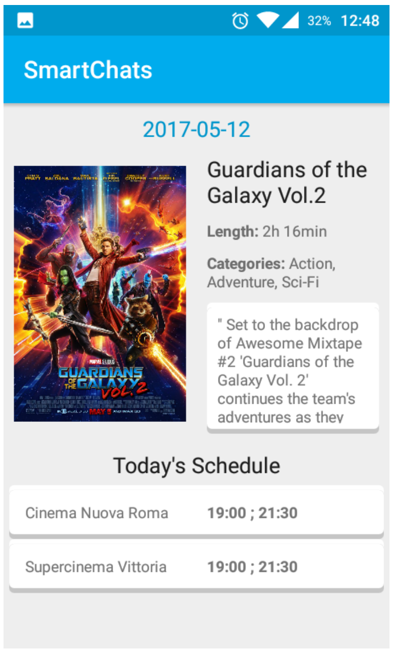
Check out:
DUO,
Via Torre Vanga, 14,

12:47

Actually, I think I'd rather go for a movie ...

(h) Suggesting a Restaurant

movie are shown on the bottom of the screen. Here you can find screening times for every cinema which is projecting the movie on a certain date. By default, screenings are referred to the current date. The user can then swipe left and right on the screen to respectively check screenings on the day before and after the current date, in the range of a week. The date will be printed out on top of the page. It is important to know that if SmartChats detects a temporal reference in the message containing the cinema intent, it will automatically load screenings for that date instead of today. Just like restaurants, users can share screenings to their friends by swiping upwards on the smartphone. The suggestion message (See Figure 2.1(j)) contains a poster preview together with name and date of the screenings. Tapping on it will load the Movie Page with screenings referred to that particular date.



2017-05-12

Guardians of the Galaxy Vol.2

Length: 2h 16min

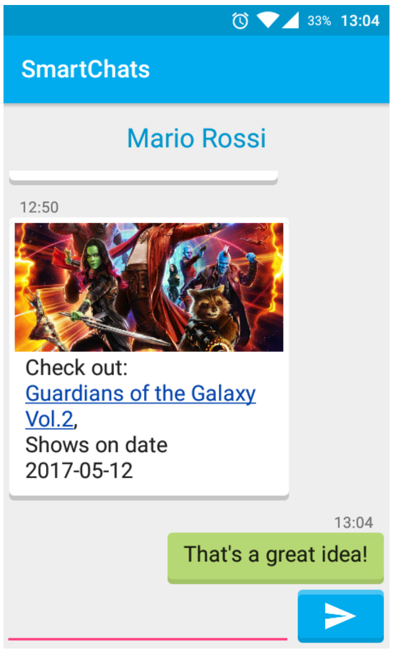
Categories: Action, Adventure, Sci-Fi

" Set to the backdrop of Awesome Mixtape #2 'Guardians of the Galaxy Vol. 2' continues the team's adventures as they

Today's Schedule

Cinema Nuova Roma	19:00 ; 21:30
Supercinema Vittoria	19:00 ; 21:30

(i) Movie Page



Mario Rossi

12:50

Check out:
Guardians of the Galaxy Vol.2,
Shows on date
2017-05-12

13:04

That's a great idea!

(j) Suggesting a Movie Screening

2.4 Architecture and Model

SmartChats software is based on an Android client application working as a frontend to users. The client is connected to many backend services to provide different functionalities. The main backend is built using PHP endpoints deployed on a Heroku public Server. A MariaDB SQL database is also installed in the Heroku app as an add-on. While the users interact with the client app, useful data is retrieved from and saved to the database.

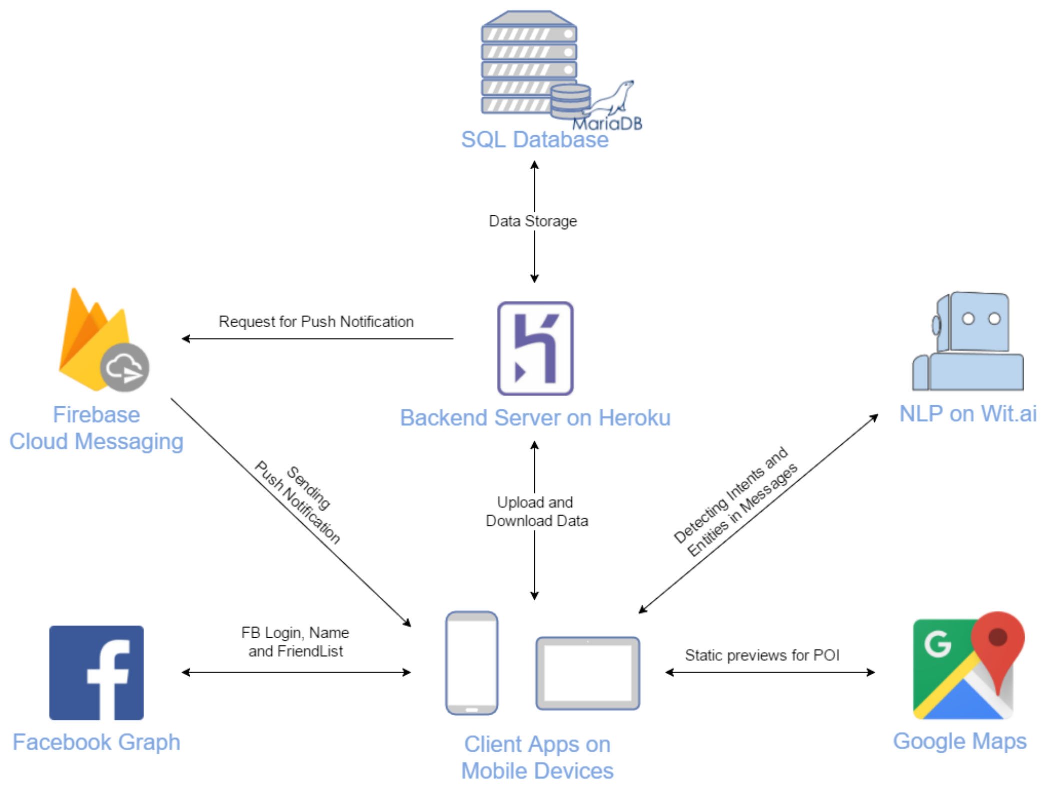


Fig. 2.1 – SmartChats Architecture schema

The MVC pattern is adopted in this software. Java entities are used to represent relevant data such as Users, Messages, and Restaurants. As the user interacts with the graphical interface, entities are created and modified to encapsulate the operations. Those objects are then used to update information saved in corresponding tables in the database. In the same way entities store objects received from database's fetch queries. HTTP Post requests and responses are used to exchange information between frontend and backend. In the client application, this is managed by the Java Class "Volley", which provides methods to create requests, set parameters, send them to an endpoint with an asynchronous process and process the eventual response. JSON is the format used to get and set content in the body of Post requests. There are other backends services obtained from third parties providers. Facebook is used to perform logins. After the first login, information about name, surname, and list of friends is obtained through requests to the Facebook Graph. To integrate Facebook Services into my app, I had to open a Facebook Developer Account to create an app and obtain a private key to be set in the Android project[31]. Also, I included Facebook SDK into my project's dependencies to access other functionalities, such the ready-to-use "Login with Facebook" button. A useful endpoint provided by Google Maps[36] allowed me to request for a preview of locations in the form of static images. Information about coordinates, zoom level and pinpoints label are sent as HTTP Get parameters. Another important backend service integrated into SmartChats is Firebase Cloud Message[44]. Just like with Facebook, a developer account is needed to bind the app with FCM functionalities. I used FCM to manage creation and reception of push notifications. The Android app doesn't directly request FCM to create a notification. This task is executed from the PHP backend using the cURL library[27]. When a notification intent is sent along with the related content and target device, FCM takes care of creating it and sending it to the targeted device. Firebase SDK provides services to have the receiving device catching the notification. These services are extended in my app to manage and create custom notifications which are shown to the user. The last service connected with SmartChats is Wit.ai. Wit.ai is the platform from Facebook we have talked about in the first chapter. I used it to create a Natural Language Processing system to analyze messages for intents and entities. Again, this service is bound to SmartChats using a private key, which is required for HTTP connections.

2.5 Backends

2.5.1 Database and Heroku Server

SmartChats' main backend is hosted on Heroku. Heroku[37] is a Platform-as-a-Services supporting many programming languages on the cloud. I decided to prefer a PHP interface to a Node.js (which I already worked with) one because I wanted to experiment with a different language for my backend. At first, I created an emulated local server with XAMPP[69]. After a consistent part of my frontend was ready for public testing, I moved to the public platform. It is very easy and straightforward to configure settings, modify files and deploy them on the cloud by git-like commands from the cmd[32]. For what concerns my database, I chose MariaDB[42], which is an improved version of MySQL. The tables are presented in the Table diagram below, with related references and foreign keys. To create the database together with the Heroku app, I had to install it as an add-on (JawsDB[39]), saving its MySQL URI as a variable in the Heroku environment to easily read it from the PHP interface. I looked for available datasets for actual restaurants, cinemas' screenings and other POIs to provide real content to the users. Sadly, the only available dataset I could find was the one about Trentino POIs, published by Trentino Open Data (and it also came with issues such as using commas both as separators and inside data strings). Then, I had to simulate the dataset containing the movie screenings. To achieve this, I wrote a short python script to generate a CSV file containing id, date, time, and cinema for some movie's screenings over 2017.

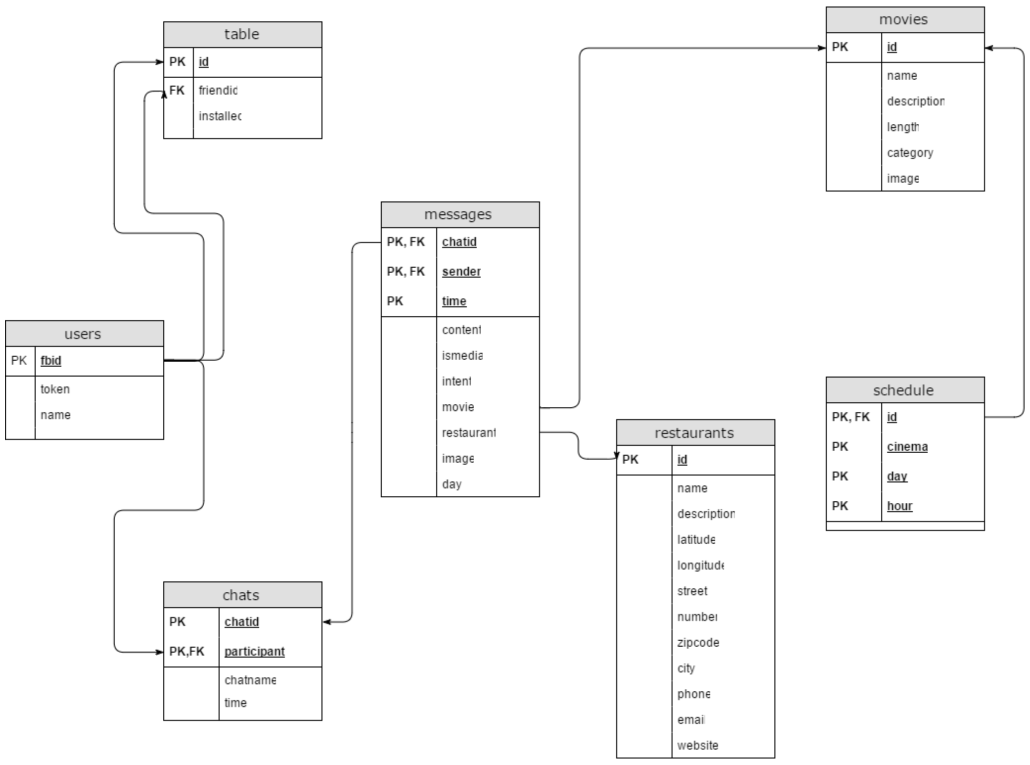
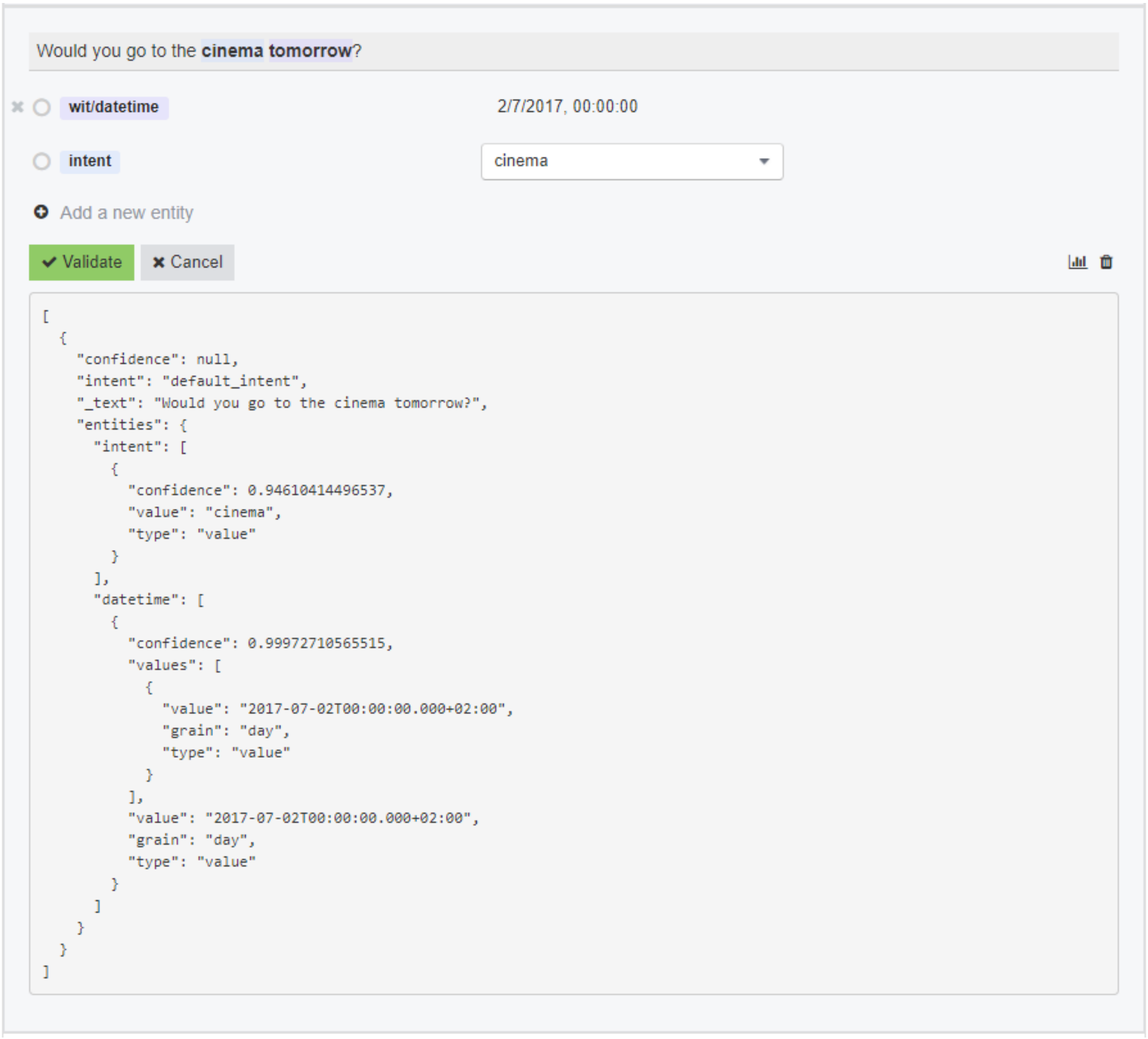


Fig. 2.2 – Database tables' structure

The PHP extension mysqli (MySQL improved) was used to work with the DB. It provides many methods to connect, prepare statements, execute them and retrieve the results. Moreover, it sanitizes automatically SQL injection attempts thanks to adopting prepared statements and parameterized queries[17]. There are four main types of PHP files in my backend. The first and shortest one is the Config.php file, which stores variables such as host address, username, and password to access the database. DbConnect.php manages the creation of a mysqli connection with the database, used to execute every query. DbOperation.php contains all the functions that run every query on the database and return the results, taking care of projections, ordering or counting when needed. Finally, all the other files represent the available endpoints of the backend. When the client app sends an HTTP Request to one of those endpoints, the relative PHP file is executed, getting the eventual content sent as JSON, calling the functions to store or get data from the database and creating the HTTP response as another JSON object. The returned content may carry selected rows for a select query or execution results for insert, delete, and update queries. The Github repository for the Backend server is available at: `https://github.com/DavideSK3/MyChatbot_php_backend`

2.5.2 Wit.ai App

To implement the Natural Language Understanding for my app, I used Wit.ai. I created a web application accessible through some endpoints that is able to analyze sentences to extract relevant information.



```
[
  {
    "confidence": null,
    "intent": "default_intent",
    "_text": "Would you go to the cinema tomorrow?",
    "entities": {
      "intent": [
        {
          "confidence": 0.94610414496537,
          "value": "cinema",
          "type": "value"
        }
      ],
      "datetime": [
        {
          "confidence": 0.99972710565515,
          "values": [
            {
              "value": "2017-07-02T00:00:00.000+02:00",
              "grain": "day",
              "type": "value"
            }
          ],
          "value": "2017-07-02T00:00:00.000+02:00",
          "grain": "day",
          "type": "value"
        }
      ]
    }
  }
]
```

Fig. 2.3 – Keyword Recognition and JSON structure in message parsing.

Wit.ai NLU system[1] is based on two kinds of objects: intents and entities. While entities are words which refer to a specific category (such as location, car brand or type of pizza), intents refer to goals and intentions expressed by the user in a sentence. My Wit.ai application can recognize two main entity categories: one related to the task or action users want to do, and the other representing temporal information. The first categorizes users intentions, which are classified into either cinema or restaurant. The other entity uses a predefined datatype in Wit.ai that is datetime, to recognize time references in phrases. Being datetime an already implemented entity, I didn't have to work on it. While for the entity expressing intents, I had to choose a way to implement it.

The platform provides two different strategies to process textual or voice utterances. Although the pattern-based one is the most powerful, it requires large datasets to gain accuracy in its prediction. That is why I chose keyword detection, which still performs good in a small domain, to analyze SmartChats' messages. To start teaching the Wit.ai application, I had to provide some training sentences in which I highlighted keywords referring to restaurant or cinema intents. For example, in the sentence "Would you catch a movie tomorrow?" I selected "movie" as a keyword for the cinema intent. I could also add some synonyms to expand the keywords vocabularies related to both topics. I included in the vocabulary both English and Italian keywords, although the datetime predefined entity only understands English words. When some example sentences are provided, Wit.ai should run some machine learning techniques to generalize its understanding to correctly process new phrases.

To have the Wit.ai application analyzing a sentence, SmartChats client sends an HTTP Get request, containing the message to be processed and the app private key as parameters. The response's body is a JSON object (See Figure 2.3) containing the results for the query. If one or more entities are found, they will be paired with a confidence ratio expressing the probability assigned by the NLU for the correctness of its evaluation. Finally, Wit.ai provides a page showing the logs for all received requests. Here, the developer can confirm the prediction of Wit.ai understanding, eventually correcting it when errors are encountered.

2.6 Frontend Mobile Application

I will not go into deep details in how the 5700 lines of code for the Frontend Mobile App were realized, but I want to provide some examples of salient features of SmartChats and how they were implemented. Code snippets for mentioned features are available in the Appendix A.2 as long as the file hierarchy for the Android Studio project A.1. The Github repository containing the Frontend application is available at `https://github.com/DavideSK3/MyChatbot_android_frontend`. You can also find the latest versions of the App at the /apk directory. The layouts for each Activity where defined using XML files. Similarly, XML was used to define some recurrent shapes, colours, strings and dimensions. Java classes were used to manage entities, adapters, activities creation, services and other useful mechanics. Let's discuss some samples from those Java Classes.

To insert the "Login with Facebook" procedure in SmartChats, I used Facebook SDK which contains, in particular, a layout element and a Java Class to implement the login button. The permissions to be requested at Login time are then set into the button object. In my case, other than default permissions, I asked to get the list of friends. Upon clicking the button, an access to Facebook Graph is made and a FacebookCallback is created to manage results by overriding three methods. Those methods allow distinguishing among Success, Cancel and Error situations. In the last two cases, the user is notified of the access failure. Otherwise, a LoginResult instance is used to catch the results received. It contains the information connected with the login process. In particular, we get the UserId and the Token for that user. Both of them are stored in a customized SharedPreferences instance, because they are needed for each further access to the user's Facebook profile (for example, to get the Name or Friend list). The UserId is permanent and it is stored in SmartChats' database to identify the user. The token is not stored because it can change at any time. After this procedure, the next method is called to get the user's name from Facebook and to proceed to the homepage. See Figure A.2

To manage HTTP requests inside the Java Android application, I chose the Volley SDK[48]. By using its methods and classes, I could manage and send requests generated by any activity using a single external class (MyVolley) The StringRequest class is used to create HTTP request objects which are then added to MyVolley's RequestQueue to be processed. Each StringRequest contains information about the type of request (GET or POST), the endpoint address, Listeners to catch successes or errors and a method to set the eventual parameters in the body as a Java Map structure. In the case reported in Figure A.3, complex JSON objects are parsed in the case of successful responses. If an error occurs, for example, if internet connection is turned off, the issue is caught and the user is informed of possible solutions.

The SharedPreferences class provides a general framework that allows to save and retrieve persistent key-value pairs of primitive data types. In SmartChats, it was used to store some recurrently useful data such as IDs and Tokens for Facebook, Firebase and Wit.ai services. The dictionary is managed by a custom SharedPrefManager class, which contains a static instance of SharedPreferences accessible through Get and Set methods. Being that the method to instantiate the object is synchronized, it is impossible to have two different instances even if the method is called more times in the same moment. The keys are defined as final Strings attributes. See figure A.4

Being that long tasks can make the app waiting for a long time, the OS can decide to stop it resulting in unexpected crashes. That is why Asynchronous tasks are needed to perform long operations. Volley requests are implemented in such way. Another case in which Async Tasks are required is the one of downloading images. In figure A.5 you can see how SmartChats create new threads to perform those downloads. I created a class extending Java AsyncTask defining the method doInBackground to define what should be done in the new thread, and the method onPostExecute to return the picture when the download is completed.

To work with recurrent objects such as messages, restaurants, users, and movies, it is useful to design entities able to encapsulate each attribute. This allows a higher grade of abstraction. Content received from the backend is stored into lists of these entities. Then, the lists are used to load the graphical interface. This is especially useful to load ListAdapters. A layout representing one item of the list is repeated more time to fill the graphical interface with all the messages or restaurants. In Figure A.6, you can see how each line in the suggestion page is loaded with restaurant data. Name and address are loaded from strings, while the static preview of the image is downloaded with the method discussed in the previous paragraph providing the URL address customized with latitude and longitude. Finally, a ClickListener is added to the ImageView map to show directions on Google Maps when the preview is clicked.

I decided to adopt natural gestures like swiping on the screen to perform some actions like suggesting restaurants and movies or skipping to next and previous day's screenings. To implement this behavior, the app had to recognize the direction of swiping. I overrode the onFling method provided in the GestureDetector.SimpleOnGestureListener class. To check the swiping direction, I confronted the coordinates of the points where the touch event started and finished, distinguishing the 4 cases. I also fixed a velocity threshold and a distance threshold to establish if the events were to be considered touches (by default) or swipes. See Figure A.7.

Finally, I want to consider how I dealt with GPS polling[72] and requesting location permissions[82]. Enabling GPS is not compulsory to use the app, thus its operativity is checked only when the user touches the button to find closest restaurants. At that time, SmartChats controls whether permissions have been allowed from the user. If not, the askPermission method is going to distinguish among older versions of Android, in which the permission is requested at install time, and Android 6.0 and newer, where dangerous permissions must be accepted at runtime. In the latter case, a pop-up asks the user to allow using GPS and then initializes the LocationManager. If permissions were already enabled, the LocationManager is asked for the last known location. If this is not available yet (it may take some seconds for the device to get the coordinates), the user is informed to try again in few seconds. Otherwise, variable for latitude and longitude are updated with the latest poll. Then, setAddressFromLocation takes care of retrieving the closest address as a string to be shown in the interface. Finally, The list of restaurants is updated, sorting it from the closest to the farthest. See Figure A.8.

2.7 Requirements, Testing and Evaluation

Before, during, and after the development of my app, I used many tools and methods to test and evaluate software and ideas.

2.7.1 Initial Questionnaire

First, I started by creating an anonymous questionnaire using Google Modules to validate some aspects with the eventual users. Data visualization for the complete set of questions can be found in Appendix A.3. To publish the questionnaire, I posted in some university groups on Facebook. I collected 83 answers a week later.

The questionnaire was structured into 4 main parts. At the beginning, I requested age and occupation of interviewees to check the heterogeneity of the user sample. The great part of them was in the range 18 to 35 years old, with more students than employed. Then, I wanted to see if and how people knew and used chatbots and virtual assistants. A few percentage of them (10-15%) answered they frequently use those technologies. Most people said they are not used to interact with that software, or they stopped after a while (70 %). 18% of the interviewees didn't even know what chatbots are. From these results, I noticed that chatbot technologies don't already have a noticeable impact on people's lifestyles. At least here in Italy. I also asked people which platforms and chatbots in particular they preferred. Telegram is the most popular platform, along with Skype and Messenger. A great part of mentioned chatbots belonged to Entertainment domain. Some other bots were used to manage university groups, traveling (Skyscanner and Uber), and tracking expedition (Trackbot[9]). The positive aspects for which people liked these chatbots were their rapidity and ease of use. On the other side, most users encountered a lack of intelligence and ability to understand questions in the bots they tried.

In the next section, I wanted to study how people are used to spend their time using smartphones. To obtain this information, I asked how often did they use five different types of apps in a range from 1 to 5. The vast majority of users said they interact with their instant messaging apps every hour. To a lesser degree, interviewees access browser apps, and at fewer rate games, with an average of 3 out of 5. Yet again, chatbots and virtual assistants are used less than once a day by over 65% of people. This data was useful to demonstrate that people prefer to keep installed few apps they frequently use such as browsers and messaging software. That allowed me to confirm my idea of creating an assistant integrated into a messaging app rather than building an external software, that would probably be uninstalled after few days to clear up some space on the smartphone. To conclude, I explained the main concept of SmartChats to see if people would show interest in the features I proposed. 80% of interviewees said it may be an interesting improvement in messaging apps. I also left the opportunity to leave an email address to participate in further testing sessions of the app.

2.7.2 Testing, Evaluation and Refactoring

While developing the app I tested the graphical interface and app functionalities both using devices and emulators. Android Studio provides a built-in Virtual Device Manager to create emulators with specific resolutions and Android OS versions. I created an emulator with same specifications as a Nexus S with Android 5.1 for testing. I also used my Samsung S2+ running Cyanogenmod 12.1. When I completed coding the app, I continued testing with different smartphones and emulators. I tried running SmartChats on more than ten devices, including Nexus 5, Nexus 6, Galaxy S5, Samsung tablets and older smartphones with lower resolutions. The only problems I encountered were graphical appearance issues in larger devices as some items' dimensions and alignments aren't set to be proportional to the screen resolution.

After testing the Graphical Interface with different devices, I proceeded with Use Case and Scenarios testing. They were carried on with individual testing and pair testing with real users. Individual tests were assigned to volunteers collected from the first questionnaire. Before sending out copies of my app to unknown testers I decided to implement the recognition and management of different releases of SmartChats. I marked each APK build with a version ID so that I could check the ID from my backend to grant or deny access to the app in a customized way (for example, blocking the unknown testers at the end of the evaluation). Afterward, I contacted the volunteers via email providing instructions to download, install and test SmartChats. The individual testing consisted of a series of tasks to be accomplished, from logging in to messaging users and accessing suggestions. The user only had a general prompt on what to do, so that I could see flaws in the interaction between user and interface. Then, the user had to fill out a personalized questionnaire to report possible issues and doubts encountered during the process. Regarding the pair testing, I decided to create a scenario-based test. This kind of testing was carried on with people I knew so that I could directly see how they performed. The couple had to follow actions depicted in a shorts script, representing two people organizing to hang out for dinner and, later on, to see a movie. In the end, I had feedback from more than 10 participants. Both use case individual tests and scenario scripts for pair tests can be found in the Appendix A.4.

Finally, I analyzed the issues reported by interviewees, evaluating the severity for each of them and refactoring the app when needed. The problem encountered by most people was the lack of tips on how to interact with the Interface, leading to the inability to proceed with tasks in some circumstances. I decided to solve this issue by inserting the "Help" voice in the menu. Upon selection, it provides a brief visual tip on what he can do in the current screen. A tester also reported a problem with GPS permissions. In fact, I realized that GPS permissions are not given at install time from Android 6.0 and later. To solve this issue, a run-time request is now given to the user when he wants to get restaurant suggestions by current position. A third relevant bug was that, in certain cases, the back button led the user to unexpected activities, following the activity stack instead of the logical flow. I fixed this by overriding the onBackPressed method in such activities. This correction also prevents the user from returning to the login page without performing a logout.

2.8 Future Plans

To conclude the work on my app, I speculated on some new improvements that could be introduced in future versions of SmartChats. Group chats could be easily added thanks to how the chat table in the database was designed. Another relevant enhancement could be made by finding more public datasets. Unfortunately, I could not find many open services providing information about movies, events or sport structures but the one about Trentino POIs. If more public datasets would be available, SmartChats could be easily improved to scale its suggestion capabilities to different topics while also providing more detailed content, such as ratings for existing restaurants. Moreover, the NLP system could be improved to have a deeper understanding of user messages. For example, it could be taught to directly recognize names of existing movies and restaurants, thus allowing the app to provide shortcuts from the chat to the detail page.

3 Conclusion and Lesson Learnt

In the latest months, I had the chance to gain a detailed knowledge about state-of-the-art chatbot technologies while also learning how to ideate, plan and work at a project autonomously.

Before starting the development on SmartChats, I never had the chance to work on a project of these dimensions. During the latest years at the University of Trento, I had to work on assignments and projects but that happened in different ways. I could follow tracks and guidelines provided by professors, with projects bounded to a restricted domain (algorithms, database, web programming), with teammates to confront ideas and slides to retrieve information from. For my thesis work, however, I had to get informed by myself about something I never did before (Android programming), with just few useful tips from my supervisors. This made me acquire new important skills. First, I learned how to practically build a complete service, from ideating and managing a database to building a backend service interface and deploying it to a public platform. From creating an Android app to using external services and endpoints through HTTP requests to web platforms. Second, and equally important, I had the opportunity to work autonomously, overcoming the gap of knowledge with which I started my project and learning to focus deeply on problems, trying different solutions to solve them in the best ways. I also learned to apply the knowledge I acquired in other courses, such as Objective programming, Web programming, Database and Human Computer Interaction.

For what concerns the theoretical aspect of my work, I had the opportunity to work in the presence and supervisions of many experts in the field of NLU, like Prof. Giuseppe Riccardi, Dr. Arindam Ghosh, and other employees at the office. Thanks to them, I could take a glance at what NLU is and how it is going to have a huge impact on future technologies. Since all of the most powerful tech companies in the world are deeply investing in AI, and in particular, in Intelligent Assistants, there is no doubt chatbots will deeply bind to existing technologies in future years. In my small sample project, SmartChats, I wanted to demonstrate a way in which Natural Language Processing can improve the User Experience in something as popular as Instant Messaging. What is going to happen in the future is yet to know, but I am sure life-changing applications of chatbots will appear in the next years. Maybe we will talk with our personal robotic chef to bake our favorite cake. Maybe we will ask our car to drive us to work while debating on the latest news. Maybe privacy will not exist anymore, with all those always-listening devices everywhere ready to catch our dialogues and steal our information? The only certain thing is that the famous technological advancement will not stop soon, and no one has the power to control how to use it or abuse it but us, humans.

Bibliography

[1] Build your first app on wit.ai. `https://wit.ai/docs/quickstart`.

[2] chatshopper for facebook messenger. `https://chatbottle.co/bots/chatshopper-1`.

[3] Financial insights made simple. `https://tinybank.co/`.

[4] Florence (beta) your health assistant. `https://florence.chat/`.

[5] Hyundai blue link. `https://www.hyundaiusa.com/bluelink/index.aspx`.

[6] Ibm watson. `https://www.ibm.com/watson/developercloud/conversation.html`.

[7] Movie finder for facebook messenger. `https://chatbottle.co/bots/movie-finder-for-messenger`.

[8] Myford mobile. `https://www.amazon.com/Ford-Motor-Company-MyFord-Mobile/dp/B01N2L9H5J`.

[9] Trackbot. `https://botfactory.info/trackbot/`.

[10] The world's first robot lawyer. `http://www.donotpay.co.uk`.

[11] X.ai. `https://x.ai/`.

[12] Xong for facebook messenger. `https://chatbottle.co/bots/xong`.

[13] Your best friend when you get hangry. `surebot.io`.

[14] Aetna introduces 'ann,' a virtual online assistant to help members. `https://news.aetna.com/news-releases/aetna-introduces-ann-a-virtual-online-assistant-to-help-members/`, June 2010.

[15] Conversational toys – the latest trend in speech technology. `https://virtualagentchat.com/tag/hello-barbie/`, February 2015.

[16] Alexa privacy policy. `http://www.alexa.com/help/privacy`, November 2016.

[17] How can i prevent sql injection in php? `https://stackoverflow.com/a/60496`, December 2016.

[18] Mykay, your banking bot. `http://kasisto.com/mykai/`, 2016.

[19] Your security and privacy are our priority. we take them seriously. `http://kasisto.com/security/`, 2016.

[20] About placing orders with alexa. `https://www.amazon.com/gp/help/customer/display.html?nodeId=201807210`, 2017.

[21] Alexa in the car: Ford, amazon to provide access to shop, search and control smart home features on the road. `https://media.ford.com/content/fordmedia/fna/us/en/news/2017/01/04/alexa-car-ford-amazon-shop-search-home.html`, January 2017.

[22] Apple business chat. https://developer.apple.com/business-chat/, June 2017.

[23] Build conversation actions for the google assistant. https://api.ai/google-assistant/, 2017.

[24] Ces 2017: Lg fridge is powered by amazon's alexa. http://www.bbc.com/news/technology-38509167, January 2017.

[25] Chatbot. https://en.wikipedia.org/wiki/Chatbot, June 2017.

[26] Cleverbot. https://en.wikipedia.org/wiki/Cleverbot, June 2017.

[27] Client url library. http://php.net/manual/en/book.curl.php, 2017.

[28] Cortana privacy policy. https://privacy.microsoft.com/en-us/privacystatement#maincortanamodule, March 2017.

[29] Desktop operating system market share. https://www.netmarketshare.com/operating-system-market-share.aspx, May 2017.

[30] Dialog system. https://en.wikipedia.org/wiki/Dialog_system, June 2017.

[31] Facebook for developers. https://developers.facebook.com/docs/android/getting-started/, June 2017.

[32] Getting started on heroku with php. https://devcenter.heroku.com/articles/getting-started-with-php, 2017.

[33] Google allo. https://en.wikipedia.org/wiki/Google_Allo, June 2017.

[34] Google assistant privacy policy. https://support.google.com/assistant/answer/7126196?p=assistant_privacy, 2017.

[35] Google home privacy policy. https://support.google.com/googlehome/answer/7072285, 2017.

[36] Google static maps developer guide. https://developers.google.com/maps/documentation/static-maps/intro, June 2017.

[37] Heroku. https://en.wikipedia.org/wiki/Heroku, June 2017.

[38] Icelandair messenger bot. http://www.icelandair.us/messenger-bot/, 2017.

[39] Jawsdb maria on heroku. https://devcenter.heroku.com/articles/jawsdb-maria#provisioning-the-add-on, 2017.

[40] Language understanding intelligent service. https://www.luis.ai/home/index, 2017.

[41] Lucida repository. https://github.com/claritylab/lucida, 2017.

[42] Mariadb. https://en.wikipedia.org/wiki/MariaDB, June 2017.

[43] Microsoft bot framework. https://www.luis.ai/home/index, 2017.

[44] Set up a firebase cloud messaging client app on android. https://firebase.google.com/docs/cloud-messaging/android/client, June 2017.

[45] Siri. https://en.wikipedia.org/wiki/Siri, June 2017.

[46] Siri privacy policy. https://www.apple.com/privacy/approach-to-privacy/, 2017.

[47] Tay. https://en.wikipedia.org/wiki/Tay_(bot), June 2017.

[48] Transmitting network data using volley. https://developer.android.com/training/volley/index.html, 2017.

[49] You need an automated assistant for your work day. https://zoom.ai/, 2017.

[50] Julie A. Ask, Michael Facemire, and Andrew Hogan. The state of chatbots, forrester. http://info.247-inc.com/rs/074-HBW-141/images/Forrester-The-State-Of-Chatbots.pdf, October 2016.

[51] Pavlo Bashmakov. Advanced natural language processing tools for bot makers – luis, wit.ai, api.ai and others (updated). https://stanfy.com/blog/advanced-natural-language-processing-tools-for-bot-makers/, December 2016.

[52] Nicolas Bayerque. A short history of chatbots and artificial intelligence. https://venturebeat.com/2016/08/15/a-short-history-of-chatbots-and-artificial-intelligence, August 2016.

[53] Lee Bell. Amazon echo's best skills – and how to use them. http://www.wired.co.uk/article/amazon-echo-skills, June 2017.

[54] Davide Belli. Smartchats, apk repository on github. https://github.com/DavideSK3/MyChatbot_android_frontend/tree/master/apk, June 2017.

[55] Vadim Berman. Comparison of bot frameworks on the market. http://blogs.aspect.com/bot-framework-s-comparison/, June 2016.

[56] Matther Black. Live from f8, group bots with messenger chat extensions. https://chatbotsmagazine.com/live-from-f8-group-bots-with-messenger-chat-extensions-641a3d66b367, April 2017.

[57] Matt Burgess. Getting started with google home: the best things to do with personal assistant. http://www.wired.co.uk/article/voice-control-google-home-best-skills-apps, June 2017.

[58] Chayan Chakrabarti. Artificial conversations for chatter bots using knowledge representation, learning, and pragmatics. https://repository.unm.edu/bitstream/handle/1928/24299/ChakrabartiDissertation.pdf, May 2014.

[59] Carlyne Chan. 155 chatbots in this brand new landscape. where does your bot fit? https://venturebeat.com/2017/06/26/bot-analytics-platform-releases-new-chatbot-landscape/, June 2017.

[60] Brian X. Chen. Siri, alexa and other virtual assistants put to the test. https://www.nytimes.com/2016/01/28/technology/personaltech/siri-alexa-and-other-virtual-assistants-put-to-the-test.html, January 2016.

[61] Eleanor Cherry. Building a chat bot? the top messaging platforms. http://blog.exiconglobal.com/which-messaging-platform-should-i-use-for-my-chat-bot, February 2017.

[62] Josh Constine. Facebook acquires wit.ai to help its developers with speech recognition and voice interfaces. https://techcrunch.com/2015/01/05/facebook-wit-ai/, January 2015.

[63] Josh Constine. Facebook messenger launches group bots and bot discovery tab. https://techcrunch.com/2017/04/18/facebook-bot-discovery/, April 2017.

[64] Josh Constine. Facebook messenger's ai 'm' suggests features to use based on your convos. https://techcrunch.com/2017/04/06/facebook-messengers-ai-m-suggests-features-to-use-based-on-your-convos/, April 2017.

[65] Jeff Dunn. Siri, alexa, google assistant, and cortana through a marathon of tests to see who's winning the virtual assistant race. http://www.businessinsider.com/siri-vs-google-assistant-cortana-alexa-2016-11, November 2016.

[66] Grant Feller. This is the real reason microsoft bought linkedin. https://www.forbes.com/sites/grantfeller/2016/06/14/this-is-the-real-reason-microsoft-bought-linkedin, June 2016.

[67] Electronic Frontier Foundation. Who has your back? https://www.eff.org/who-has-your-back-government-data-requests-2015, 2015.

[68] Nikki Gilliland. Domino's introduces 'dom the pizza bot' for facebook messenger. https://www.econsultancy.com/blog/68184-domino-s-introduces-dom-the-pizza-bot-for-facebook-messenger, August 2016.

[69] Belal Khan. Firebase cloud messaging for android using php and mysql. https://www.simplifiedcoding.net/firebase-cloud-messaging-android/, November 2016.

[70] Greg Kumparak. Google acquires api.ai, a company helping developers build bots that aren't awful to talk to. https://techcrunch.com/2016/09/19/google-acquires-api-ai-a-company-helping-developers-build-bots-that-arent-awful-to-talk-to/, September 2016.

[71] Nicole Lee. Facebook's ai assistant is ready to hang out in messenger. https://www.engadget.com/2017/04/06/facebook-m-suggestions/, June 2017.

[72] m Joseph Kulandai. Get current location in android. http://javapapers.com/android/get-current-location-in-android/, May 2013.

[73] John Markoff. Computer wins on 'jeopardy!': Trivial, it's not. http://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html, February 2011.

[74] Catherine McGloin. How to search for flights with skyscanner's new facebook messenger bot. https://www.skyscanner.net/news/tools/skyscanner-facebook-messenger-bot/, May 2016.

[75] Matthew Panzarino. Samsung acquires viv, a next-gen ai assistant built by the creators of apple's siri. https://techcrunch.com/2016/10/05/samsung-acquires-viv-a-next-gen-ai-assistant-built-by-creators-of-apples-siri/, October 2016.

[76] Rahul. Say hello to uber on messenger. https://newsroom.uber.com/messengerlaunch/, December 2015.

[77] Giuseppe Riccardi. Towards healthcare personal agents. In *Proceedings of the 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research Including Business Opportunities and Challenges*, RFMIR '14, pages 53–56, New York, NY, USA, 2014. ACM.

[78] Giuseppe Riccardi. Deconstructing and debunking bot technology. http://latemar.science.unitn.it/segue_userFiles/2017Mobile/Bot-Lecture-r.pdf, May 2017.

[79] John H. Richardson. Viv will replace your smartphone with your fridge and then take over the world. http://www.esquire.com/lifestyle/a34630/viv-artificial-intelligence-0515/, April 2015.

[80] Brian Roemmele. What is apple business chat and why is it important? http://www.huffingtonpost.com/entry/what-is-apple-business-chat-and-why-is-it-important_us_59405afce4b04c03fa261644, June 2017.

[81] Harriet Taylor. Bank of america launches ai chatbot erica — here's what it does. http://www.cnbc.com/2016/10/24/bank-of-america-launches-ai-chatbot-erica--heres-what-it-does.html, October 2016.

[82] Clinton Teegarden. Runtime permissions: Best practices and how to gracefully handle permission removal. https://www.captechconsulting.com/blogs/runtime-permissions-best-practices-and-how-to-gracefully-handle-permission-removal, September 2015.

[83] Nikos Vaggalis. Lucida for personal artificial intelligence. http://www.i-programmer.info/news/105-artificial-intelligence/10043-lucida-personal-artificial-intelligence.html, September 2016.

[84] Warren Volkmann. Facebook announces hp print bot at f8. https://developers.hp.com/public/blog/facebook-announces-hp-print-bot-f8, February 2016.

[85] Matt Weinberger. Microsoft explains its plan to win the 'battle for the future' against amazon's alexa and google assistant. http://www.businessinsider.com/microsoft-cortana-vs-amazon-echo-2017-1, January 2017.

[86] Matt Weinberger. Why amazon's echo is totally dominating — and what google, microsoft, and apple have to do to catch up. http://www.businessinsider.com/amazon-echo-google-home-microsoft-cortana-apple-siri-2017-1, January 2017.

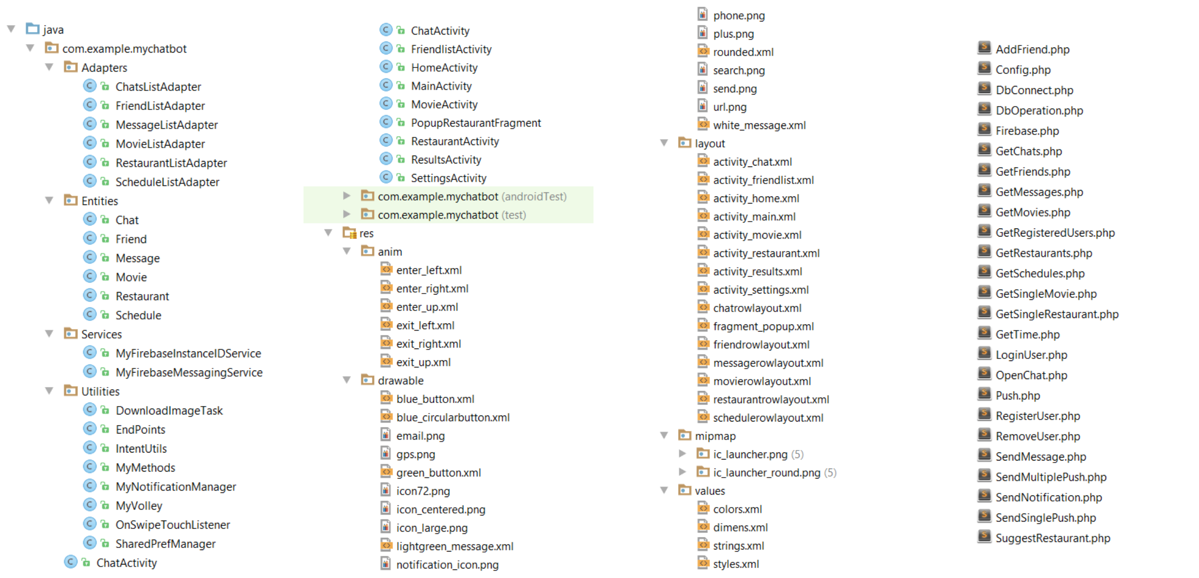
[87] Key Yeung. Healthtap launches facebook messenger bot that provides fast access to health care. https://venturebeat.com/2016/07/13/healthtap-launches-facebook-messenger-bot-that-provides-fast-access-to-healthcare/, June 2016.

[88] Adelyn Zhou. 100 best bots for brands & businesses. http://www.topbots.com/100-best-bots-brands-businesses/, March 2017.

Appendix

A.1 Project Hierarchy

Project Hierarchy for Frontend (Java Classes and Layout XML) and Backend (PHP Interface). Lines of Code: PHP interface: 1073 Java Classes: 4397 XML Layout: 1330 Total: 6800



```
java
  com.example.mychatbot
    Adapters
      ChatsListAdapter
      FriendListAdapter
      MessageListAdapter
      MovieListAdapter
      RestaurantListAdapter
      ScheduleListAdapter
    Entities
      Chat
      Friend
      Message
      Movie
      Restaurant
      Schedule
    Services
      MyFirebaseInstanceIDService
      MyFirebaseMessagingService
    Utilities
      DownloadImageTask
      EndPoints
      IntentUtils
      MyMethods
      MyNotificationManager
      MyVolley
      OnSwipeTouchListener
      SharedPrefManager
    ChatActivity
```

(a) Java Classes

```
    ChatActivity
    FriendlistActivity
    HomeActivity
    MainActivity
    MovieActivity
    PopupRestaurantFragment
    RestaurantActivity
    ResultsActivity
    SettingsActivity
  com.example.mychatbot (androidTest)
  com.example.mychatbot (test)
res
  anim
    enter_left.xml
    enter_right.xml
    enter_up.xml
    exit_left.xml
    exit_right.xml
    exit_up.xml
  drawable
    blue_button.xml
    blue_circularbutton.xml
    email.png
    gps.png
    green_button.xml
    icon72.png
    icon_centered.png
    icon_large.png
    lightgreen_message.xml
    notification_icon.png
```

(b) Java and XML

```
    phone.png
    plus.png
    rounded.xml
    search.png
    send.png
    url.png
    white_message.xml
  layout
    activity_chat.xml
    activity_friendlist.xml
    activity_home.xml
    activity_main.xml
    activity_movie.xml
    activity_restaurant.xml
    activity_results.xml
    activity_settings.xml
    chatrowlayout.xml
    fragment_popup.xml
    friendrowlayout.xml
    messagerowlayout.xml
    movierowlayout.xml
    restaurantrowlayout.xml
    schedulerowlayout.xml
  mipmap
    ic_launcher.png (5)
    ic_launcher_round.png (5)
  values
    colors.xml
    dimens.xml
    strings.xml
    styles.xml
```

(c) XML Layout

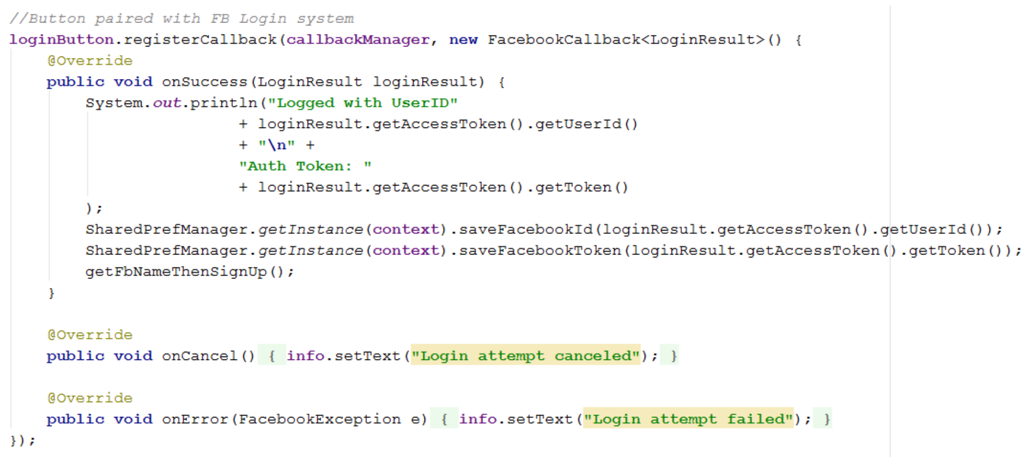
```
AddFriend.php
Config.php
DbConnect.php
DbOperation.php
Firebase.php
GetChats.php
GetFriends.php
GetMessages.php
GetMovies.php
GetRegisteredUsers.php
GetRestaurants.php
GetSchedules.php
GetSingleMovie.php
GetSingleRestaurant.php
GetTime.php
LoginUser.php
OpenChat.php
Push.php
RegisterUser.php
RemoveUser.php
SendMessage.php
SendMultiplePush.php
SendNotification.php
SendSinglePush.php
SuggestRestaurant.php
```

(d) PHP Endpoints

Fig. A.1 – Hierarchy

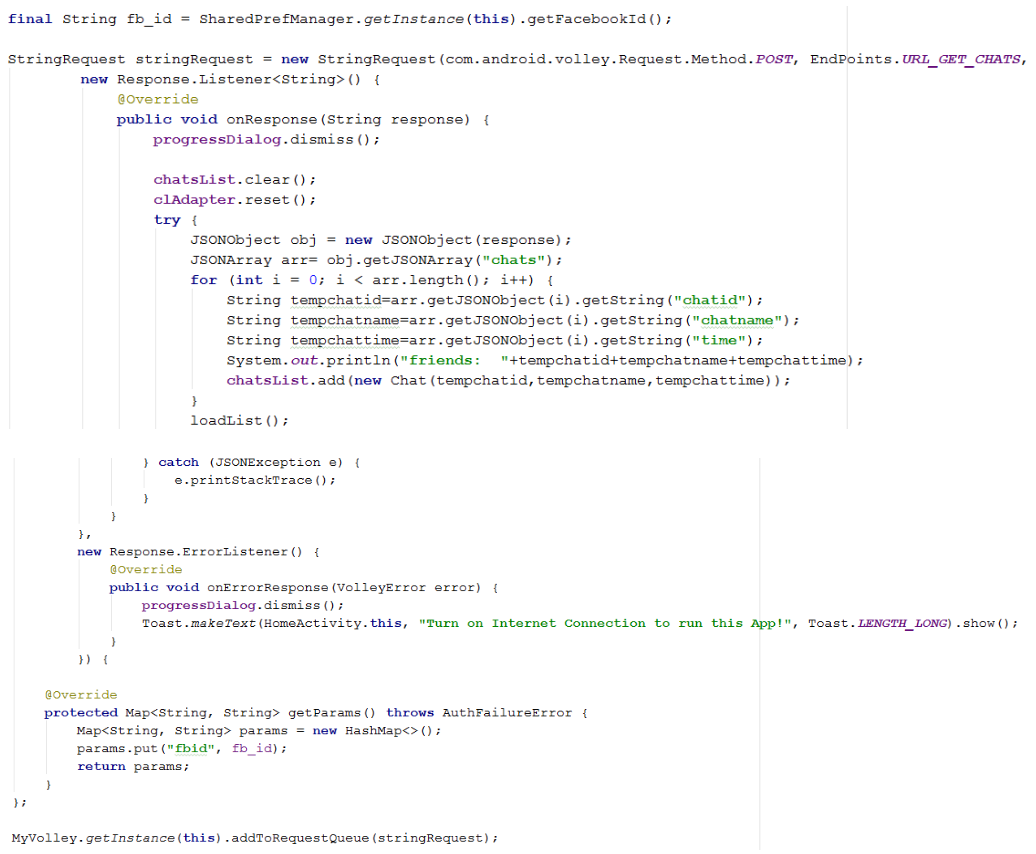
A.2 Code Snippets

Here are some code snippets to see how I implemented some relevant aspects of the Android application in Java.



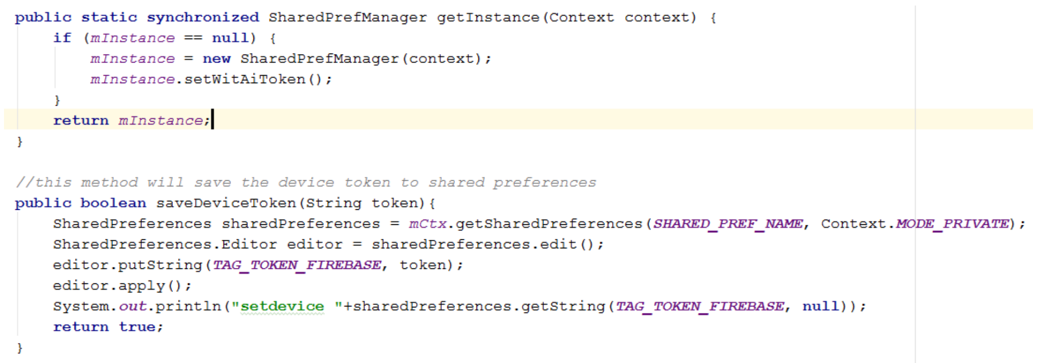
```java
//Button paired with FB Login system
loginButton.registerCallback(callbackManager, new FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult) {
        System.out.println("Logged with UserID"
                        + loginResult.getAccessToken().getUserId()
                        + "\n" +
                        "Auth Token: "
                        + loginResult.getAccessToken().getToken()
        );
        SharedPrefManager.getInstance(context).saveFacebookId(loginResult.getAccessToken().getUserId());
        SharedPrefManager.getInstance(context).saveFacebookToken(loginResult.getAccessToken().getToken());
        getFbNameThenSignUp();
    }

    @Override
    public void onCancel() { info.setText("Login attempt canceled"); }

    @Override
    public void onError(FacebookException e) { info.setText("Login attempt failed"); }
});
```

Fig. A.2 – Facebook Login Button



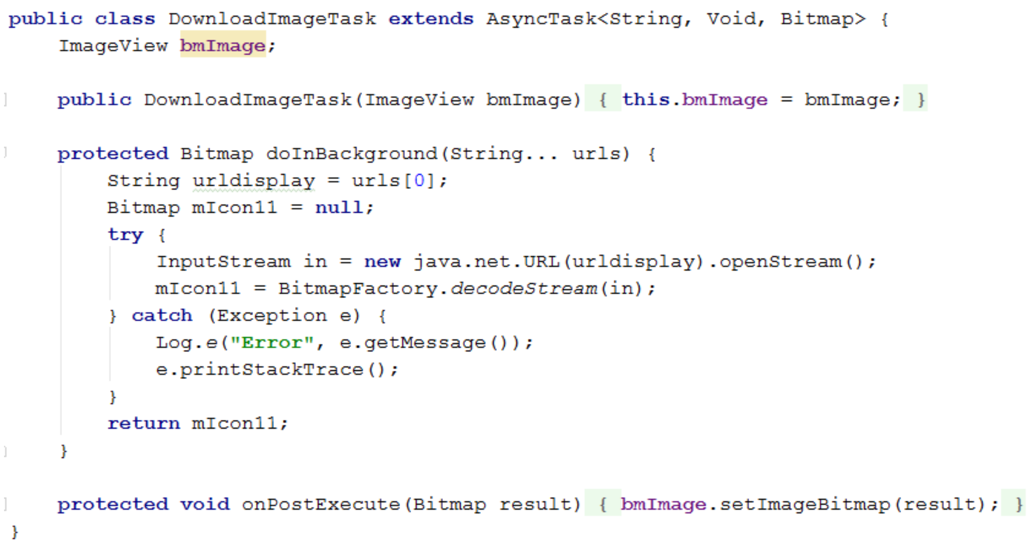
```java
final String fb_id = SharedPrefManager.getInstance(this).getFacebookId();

StringRequest stringRequest = new StringRequest(com.android.volley.Request.Method.POST, EndPoints.URL_GET_CHATS,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                progressDialog.dismiss();

                chatsList.clear();
                clAdapter.reset();
                try {
                    JSONObject obj = new JSONObject(response);
                    JSONArray arr= obj.getJSONArray("chats");
                    for (int i = 0; i < arr.length(); i++) {
                        String tempchatid=arr.getJSONObject(i).getString("chatid");
                        String tempchatname=arr.getJSONObject(i).getString("chatname");
                        String tempchattime=arr.getJSONObject(i).getString("time");
                        System.out.println("friends:  "+tempchatid+tempchatname+tempchattime);
                        chatsList.add(new Chat(tempchatid,tempchatname,tempchattime));
                    }
                    loadList();

                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                progressDialog.dismiss();
                Toast.makeText(HomeActivity.this, "Turn on Internet Connection to run this App!", Toast.LENGTH_LONG).show();
            }
        }) {

    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("fbid", fb_id);
        return params;
    }
};

MyVolley.getInstance(this).addToRequestQueue(stringRequest);
```

Fig. A.3 – Volley HTTP Request



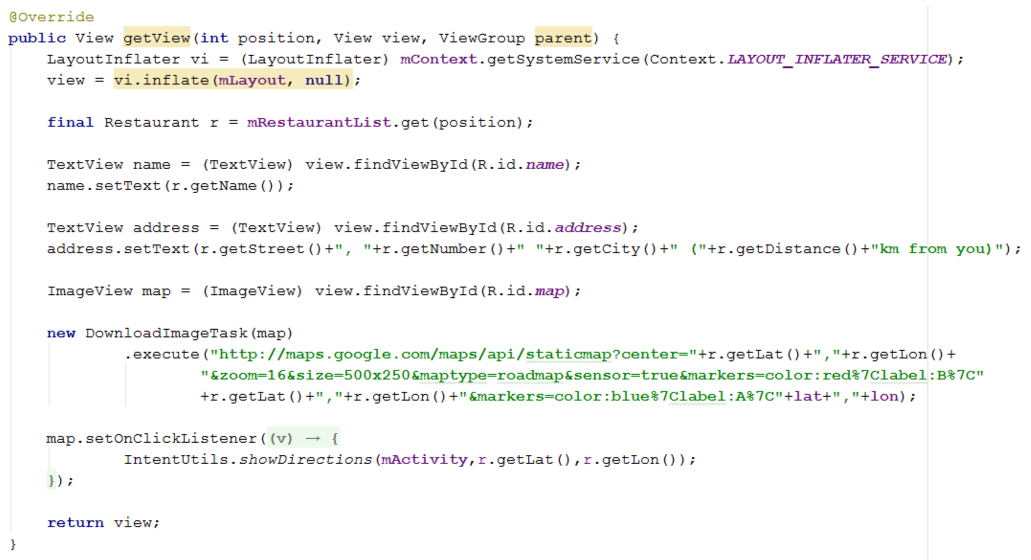
```java
public static synchronized SharedPrefManager getInstance(Context context) {
    if (mInstance == null) {
        mInstance = new SharedPrefManager(context);
        mInstance.setWitAiToken();
    }
    return mInstance;
}

//this method will save the device token to shared preferences
public boolean saveDeviceToken(String token){
    SharedPreferences sharedPreferences = mCtx.getSharedPreferences(SHARED_PREF_NAME, Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putString(TAG_TOKEN_FIREBASE, token);
    editor.apply();
    System.out.println("setdevice "+sharedPreferences.getString(TAG_TOKEN_FIREBASE, null));
    return true;
}
```

Fig. A.4 – SharedPreferences Manager



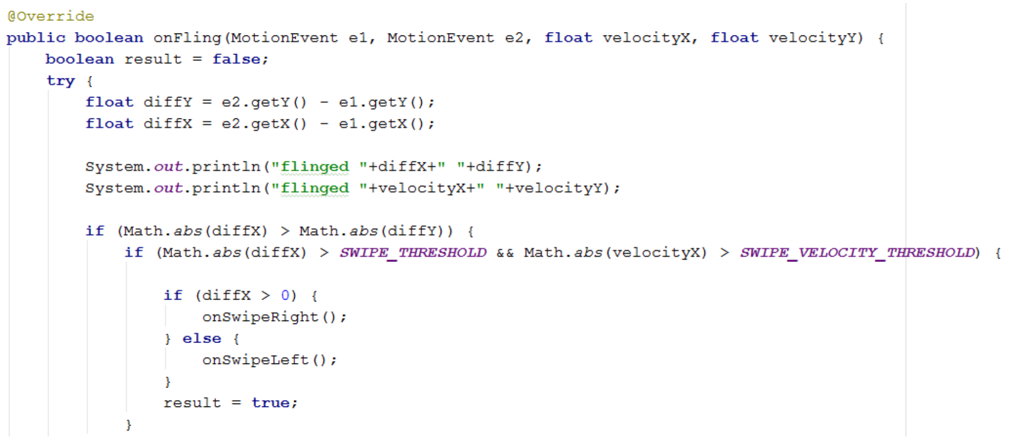
```java
public class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
    ImageView bmImage;

    public DownloadImageTask(ImageView bmImage) { this.bmImage = bmImage; }

    protected Bitmap doInBackground(String... urls) {
        String urldisplay = urls[0];
        Bitmap mIcon11 = null;
        try {
            InputStream in = new java.net.URL(urldisplay).openStream();
            mIcon11 = BitmapFactory.decodeStream(in);
        } catch (Exception e) {
            Log.e("Error", e.getMessage());
            e.printStackTrace();
        }
        return mIcon11;
    }

    protected void onPostExecute(Bitmap result) { bmImage.setImageBitmap(result); }
}
```

Fig. A.5 – Download images with Asynctask



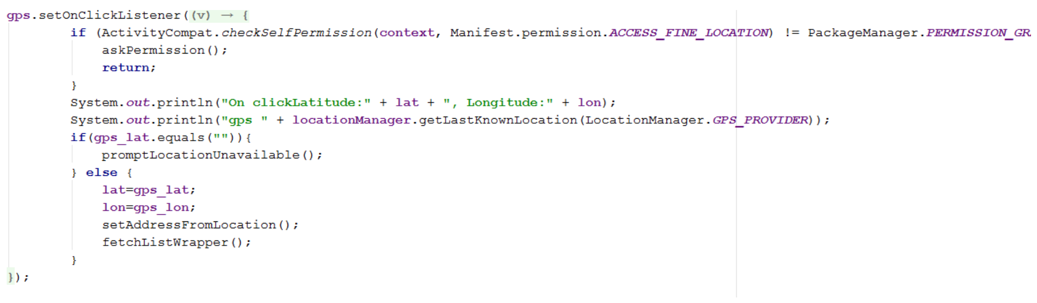
```java
@Override
public View getView(int position, View view, ViewGroup parent) {
    LayoutInflater vi = (LayoutInflater) mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    view = vi.inflate(mLayout, null);

    final Restaurant r = mRestaurantList.get(position);

    TextView name = (TextView) view.findViewById(R.id.name);
    name.setText(r.getName());

    TextView address = (TextView) view.findViewById(R.id.address);
    address.setText(r.getStreet()+", "+r.getNumber()+" "+r.getCity()+" ("+r.getDistance()+"km from you)");

    ImageView map = (ImageView) view.findViewById(R.id.map);

    new DownloadImageTask(map)
            .execute("http://maps.google.com/maps/api/staticmap?center="+r.getLat()+","+r.getLon()+
                    "&zoom=16&size=500x250&maptype=roadmap&sensor=true&markers=color:red%7Clabel:B%7C"
                    +r.getLat()+","+r.getLon()+"&markers=color:blue%7Clabel:A%7C"+lat+","+lon);

    map.setOnClickListener((v) → {
            IntentUtils.showDirections(mActivity,r.getLat(),r.getLon());
    });

    return view;
}
```

Fig. A.6 – Entities in ListAdapters

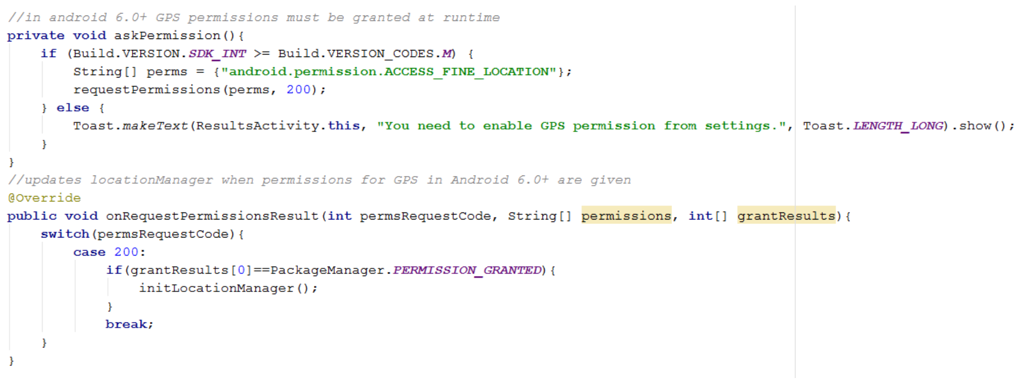


```java
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
    boolean result = false;
    try {
        float diffY = e2.getY() - e1.getY();
        float diffX = e2.getX() - e1.getX();

        System.out.println("flinged "+diffX+" "+diffY);
        System.out.println("flinged "+velocityX+" "+velocityY);

        if (Math.abs(diffX) > Math.abs(diffY)) {
            if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) > SWIPE_VELOCITY_THRESHOLD) {

                if (diffX > 0) {
                    onSwipeRight();
                } else {
                    onSwipeLeft();
                }
                result = true;
            }
```

Fig. A.7 – Swipe Listeners



```java
gps.setOnClickListener((v) → {
        if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GR
            askPermission();
            return;
        }
        System.out.println("On clickLatitude:" + lat + ", Longitude:" + lon);
        System.out.println("gps " + locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER));
        if(gps_lat.equals("")){
            promptLocationUnavailable();
        } else {
            lat=gps_lat;
            lon=gps_lon;
            setAddressFromLocation();
            fetchListWrapper();
        }
});
```



```java
//in android 6.0+ GPS permissions must be granted at runtime
private void askPermission(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        String[] perms = {"android.permission.ACCESS_FINE_LOCATION"};
        requestPermissions(perms, 200);
    } else {
        Toast.makeText(ResultsActivity.this, "You need to enable GPS permission from settings.", Toast.LENGTH_LONG).show();
    }
}
//updates locationManager when permissions for GPS in Android 6.0+ are given
@Override
public void onRequestPermissionsResult(int permsRequestCode, String[] permissions, int[] grantResults){
    switch(permsRequestCode){
        case 200:
            if(grantResults[0]==PackageManager.PERMISSION_GRANTED){
                initLocationManager();
            }
            break;
    }
}
```

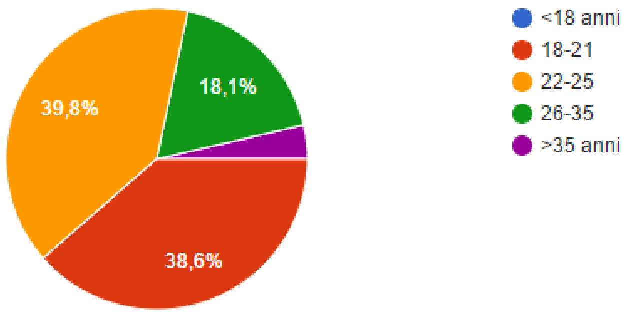
Fig. A.8 – Location Listener and Asking GPS Permissions

A.3 Questionnaire

Here it is a complete data visualization of the questionnaire results (83 answers). Questions and answers are in Italian, as the interviewees were Italian students.

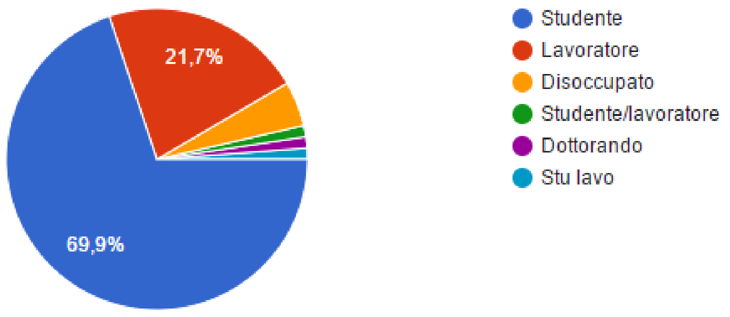
A che fascia di età appartieni?

83 risposte



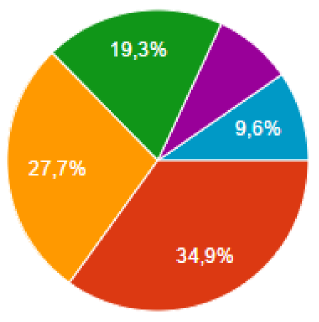
Qual è la tua occupazione?

83 risposte



Riguardo gli Assistenti Virtuali:

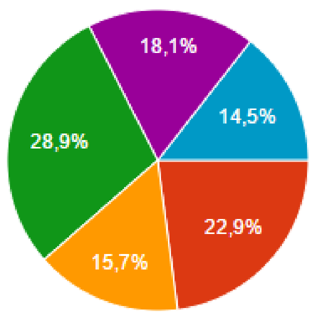
83 risposte



● Li usi occasionalmente o di frequente
● Li usi raramente
● Li hai provati ma hai smesso di usarli
● Ne hai sentito parlare ma non li hai mai usati
● Prima d'ora non sapevi della loro esistenza
● Li usi di occasionalmente o di frequente

Riguardo i Chatbot:

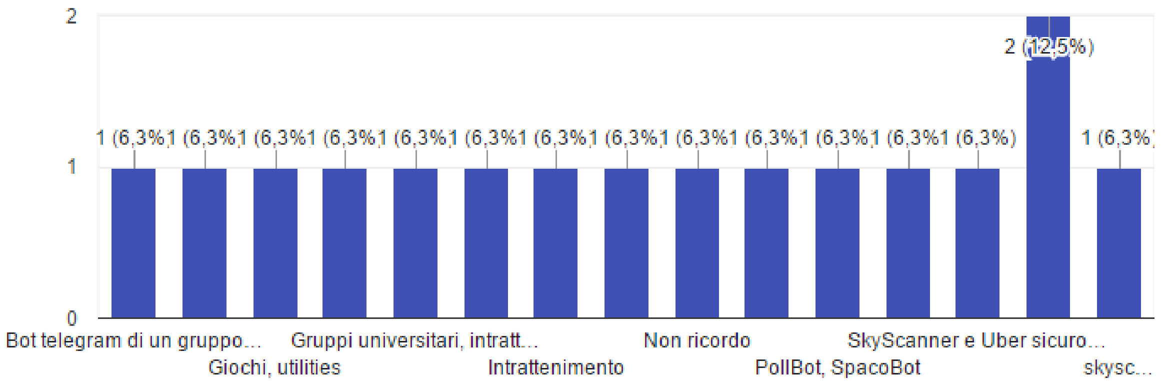
83 risposte



● Li usi occasionalmente o di frequente
● Li usi raramente
● Li hai provati ma hai smesso di usarli
● Ne hai sentito parlare ma non li hai mai usati
● Prima d'ora non sapevi della loro esistenza
● Li usi di occasionalmente o di frequente

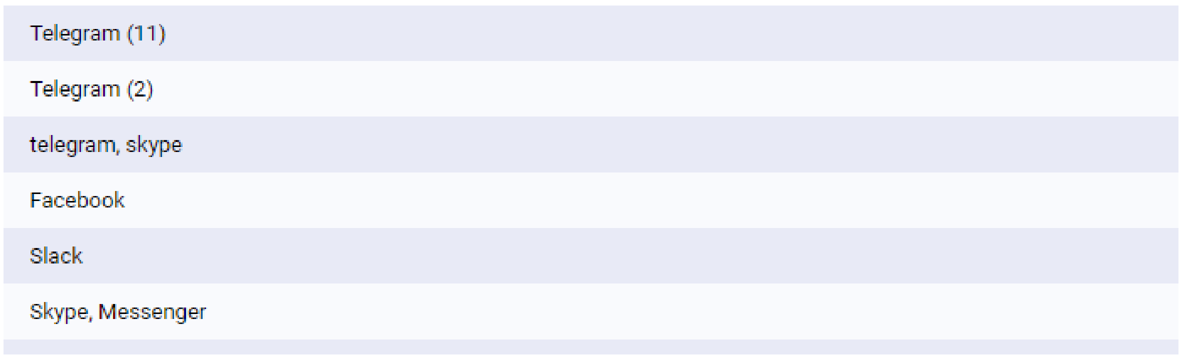
Quali Chatbot hai provato? (facoltativo) (es: Info/Gruppi universitari, Intrattenimento, SkyScanner, CNN, Uber,...)

16 risposte



Su che piattaforma (facoltativo) (es: Messenger, Telegram, Skype,...)

23 risposte



Telegram (11)

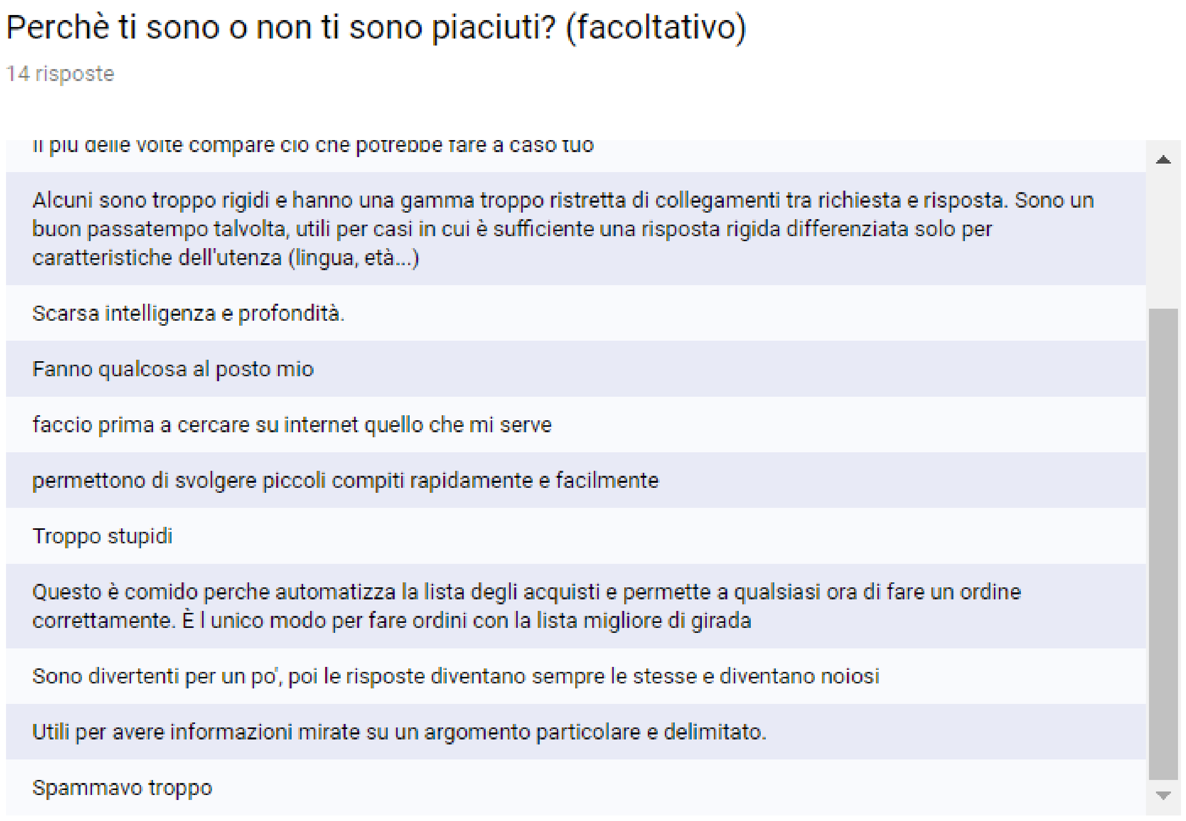
Telegram (2)

telegram, skype

Facebook

Slack

Skype, Messenger



Perchè ti sono o non ti sono piaciuti? (facoltativo)

14 risposte

il più delle volte compare ciò che potrebbe fare a caso tuo

Alcuni sono troppo rigidi e hanno una gamma troppo ristretta di collegamenti tra richiesta e risposta. Sono un buon passatempo talvolta, utili per casi in cui è sufficiente una risposta rigida differenziata solo per caratteristiche dell'utenza (lingua, età...)

Scarsa intelligenza e profondità.

Fanno qualcosa al posto mio

faccio prima a cercare su internet quello che mi serve

permettono di svolgere piccoli compiti rapidamente e facilmente

Troppo stupidi

Questo è comido perche automatizza la lista degli acquisti e permette a qualsiasi ora di fare un ordine correttamente. È l unico modo per fare ordini con la lista migliore di girada

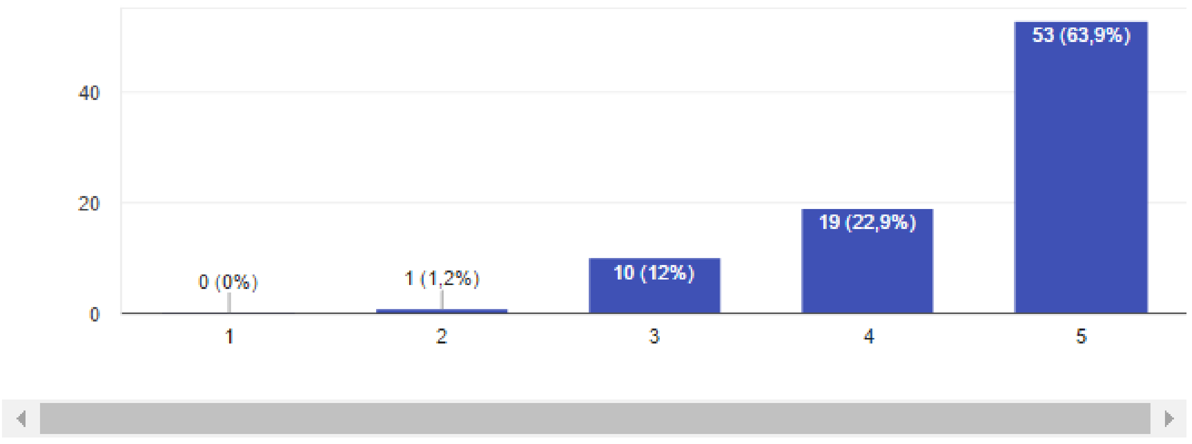
Sono divertenti per un po', poi le risposte diventano sempre le stesse e diventano noiosi

Utili per avere informazioni mirate su un argomento particolare e delimitato.

Spammavo troppo

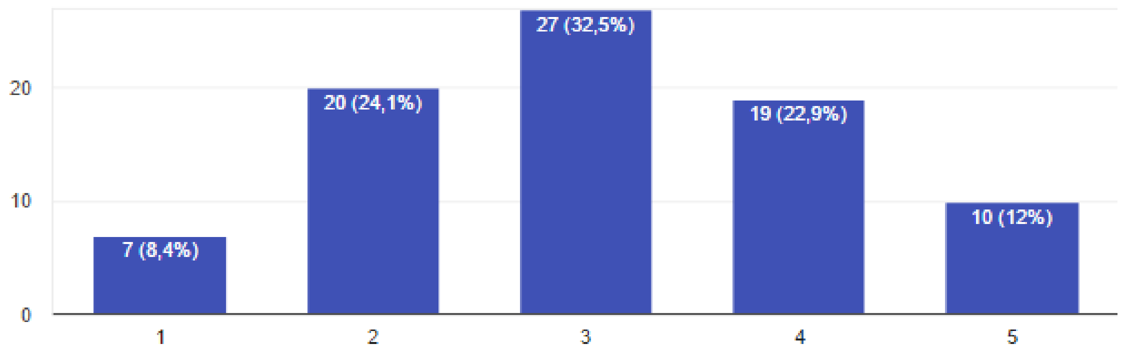
App di messaggistica (Whatsapp, Messenger, Telegram,...)

83 risposte



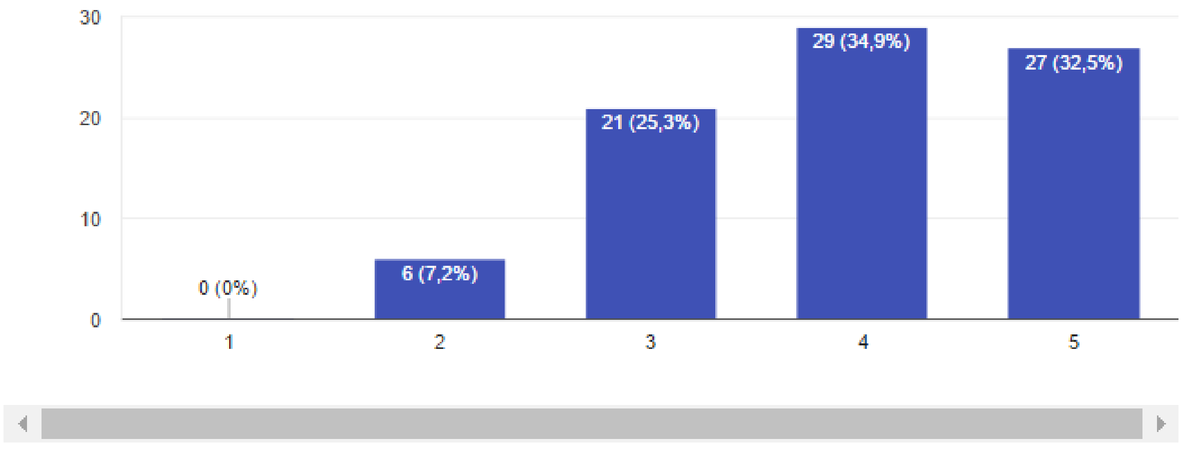
Altre app (Giochi, svago, informazione,...)

83 risposte



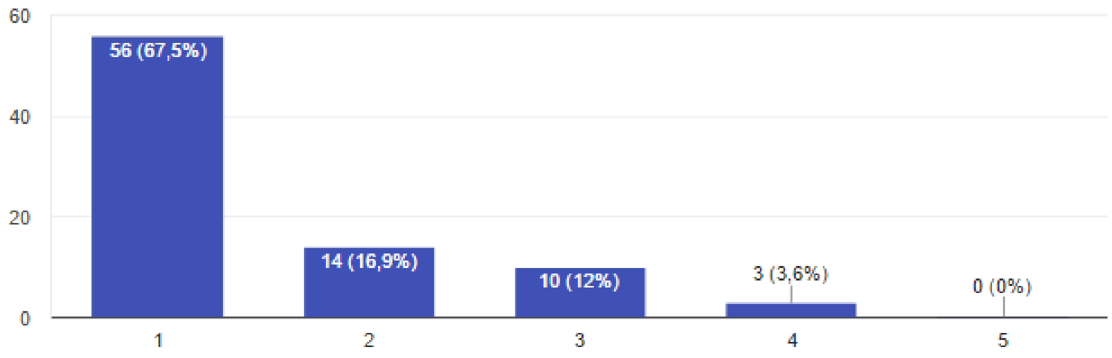
Browser (Chrome,Firefox,Safari,...)

83 risposte



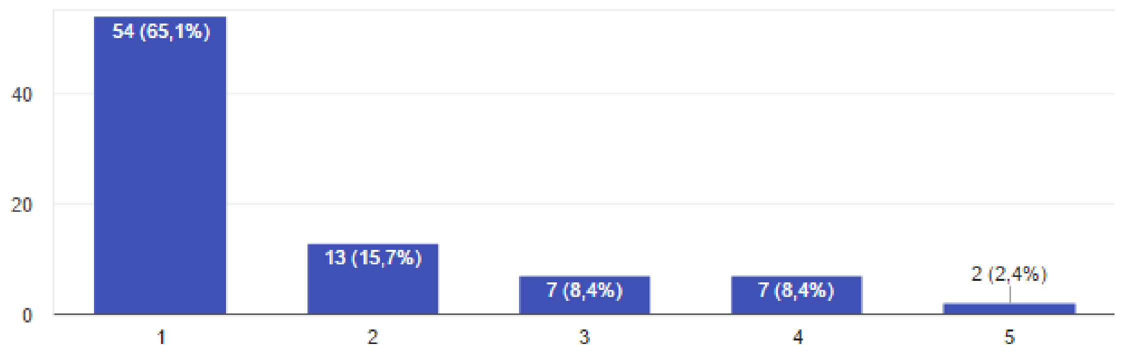
Assistenti virtuali (Siri, Google Now,...)

83 risposte



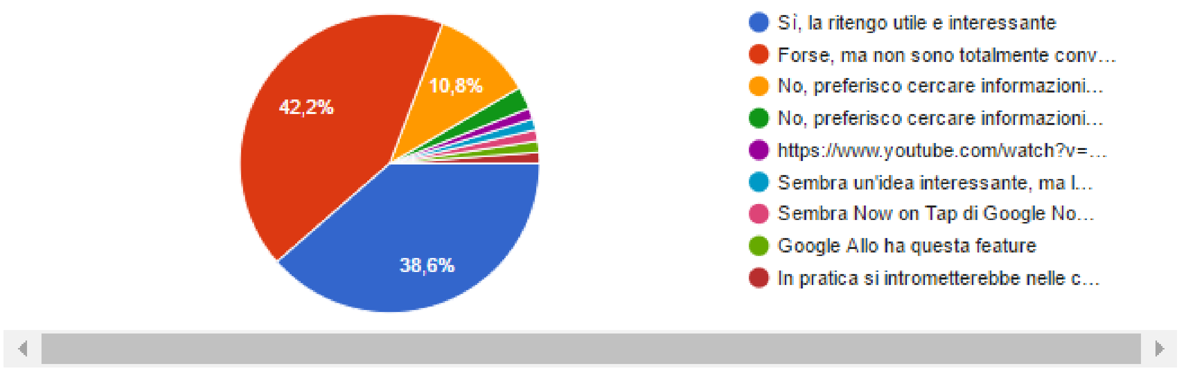
Chatbots (Bot di Telegram, Messenger, Skype,...)

83 risposte



Ti piacerebbe una feature simile nella tua app di messaggistica?

83 risposte



Sei rimasto incuriosito e ti va di saperne di più sul progetto? Vuoi partecipare alla fase di testing? lascia la tua mail qui sotto!

13 risposte

A.4 Testing

A.4.1 Use-Cases (Individual testing)

Here you can find the transcript of the Use-Case testing (in Italian)

– Scarica l'apk sul telefono dal link fornito sopra
– Installa l'app dal telefono trovando l'apk nella cartella download, ti verrà chiesto di acconsentire all'installazione di app da sorgenti sconosciute (ovvero non presenti su Play Store)
– Apri l'app
01 Effettua Login con Facebook
02 Effettua Logout dall'applicazione dal menù
03 Dopo aver rieffettuato il login, crea una nuova chat
– alla prima esecuzione dell'app la tua lista amici è vuota, sincronizzala con facebook e dovresti ora vedere il mio profilo
04 Invia il tuo primo messaggio
05 Chiedimi di uscire a cena in un altro messaggio (in italiano o inglese)
– Se l'Intelligenza Artificiale ha riconosciuto il tuo intento di mangiare fuori, un pulsante compare affianco al messaggio interessato, cliccalo per aprire la pagina con la lista dei ristoranti (di default impostata a Trento)
06 visualizza la lista dei ristoranti più vicini a te usando il GPS (il database contiene per ora solo i ristoranti di Trento, non preoccuparti di questo)
07 visualizza la lista dei ristoranti più vicini a "Via Roma, Trento"
– Entra nella pagina dettagli ristorante premendo su uno dei ristoranti in lista, ora da qui prova a:
08 aprire il collegamento all'app "telefono" del tuo smartphone per visualizzare il numero del ristorante
09 visitare il sito web del ristorante
10 aprire una nuova mail indirizzata al ristorante
11 trovare il ristorante su Google Maps
– Effettuando uno swipe (scorrendo il dito sullo schermo) verso l'altro, puoi inoltrare i dettagli ristorante come messaggio, così da condividerlo con l'utente con cui stai messaggiando
12 Invia il ristorante come messaggio, poi prova ad aprire direttamente le info del ristorante dalla chat
13 Dalla chat, consiglia (questa volta scrivendo in lingua inglese) di andare al cinema tra una settimana
– Un altro pulsante contestuale a fianco del messaggio dovrebbe comparire, il quale ti porterà alla lista dei film
14 Visualizza la lista dei film presenti al cinema e scegline uno
– Se l'Intelligenza artificiale ha riconosciuto la posizione temporale nel tuo messaggio (next week), dovresti vedere la programmazione del film inerente al film scelto per la data tra 7 giorni
15 Effettuando swipe destra/sinistra, puoi vedere la programmazione al cinema nei giorni da quello scelto a una settimana dopo.
16 Sempre tramite Swipe in alto, invia il film per la data che hai scelto all'altro utente.
– I task sono finiti, completa il questionario rispondendo alle ultime domande generiche sull'esperienza.

A.4.2 Scenario (Pair testing)

Here you can find the transcripts of scenario simulation between two people.

BOTH Login
BOTH Logout
BOTH Login (again)
USER 1 Closes App
USER 2 Clicks Help from Menu
USER 2 Clicks new chat
USER 2 Syncs friendlist

USER 2 Opens chat with USER 1
USER 2 Sends "Hey there!"
USER 1 Opens app froom notification (Firebase may take time in sendind push notifications)
USER 1 Navigates to chat, answers message: "Hi, USER 2"
USER 2 Sends: "Would you like to hang out for dinner tonight?"
USER 1 Sends: "Yeah, I'd like to!"
USER 2 Clicks contextual link
USER 2 Finds Nearby restaurants
USER 2 Looks for restaurants in "Via Roma, Trento"
USER 2 Opens "Il cappello restaurant"
USER 2 Checks its location on Google Maps
USER 2 Checks phone number, email, website
USER 2 Shares restaurant
USER 1 Sends "I've been there, they serve really good food"
USER 1 Sends "Also, I'd like to catch a movie on the weekend, wanna come with me?"
USER 1 Clicks link
USER 1 Chooses "Guardians of the Galaxy"
USER 1 Checks movie schedule, swiping to saturday's
USER 1 Shares saturday's schedule
USER 2 Sends "Awesome, everybody is talking about it. Let's go for the 21:30 screening at Nuova Roma"
USER 1 Sends "Alright, see you tonight then.. :)"
BOTH Close app pressing back