Chatbots as a Novel Access Method for Government Open Data[⋆]

Simone Porreca, Francesco Leotta, Massimo Mecella, and Tiziana Catarci

Sapienza Università di Roma
Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti
`porreca.1673726@studenti.uniroma1.it`,
`{leotta,mecella,catarci}@diag.uniroma1.it`

Abstract. In this discussion paper, we propose to employ chatbots as a user-friendly interface for open data published by organizations, specifically focusing on public administrations. Open data are especially useful in e-Government initiatives but their exploitation is currently hampered to end users by the lack of user-friendly access methods. On the other hand, current UX in social networks have made people used to chatting. Building on cognitive technologies, we prototyped a chatbot on top of the OpenCantieri dataset published by the Italian Ministero delle Infrastrutture e Trasporti, and we argue that such a model can be extended as a generally available access method to open data.

1 Introduction

Open data generally refers to the idea that some data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control. In particular, according to the Open Definition, "a piece of data is open if anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness)"[1]. Some characteristics should be granted to provide open data, namely *(i)* accessibility – all the users can freely access to data, mostly free or at a very low cost, *(ii)* machine readability – data can be naturally "understood" and processed by machines, *(iii)* rights – data are released under certain licenses that bound softly the usage, the transformation and the distribution of those data.

⋆ An extended version of this paper appears as S. Porreca, F. Leotta, M. Mecella, S. Vassos and T. Catarci: Accessing Government Open Data through Chatbots. *Proc. Int. Workshop on the Practice of the Open Web (practi-O-web 2017), at ICWE 2017*, Rome, Italy, June 2017, Springer. This work has been partly supported by the Italian projects *NEPTIS*, *SM&ST* and *RoMA*, and the EU RISE prioject *FIRST*. The work of Francesco Leotta has been partly supported by the Lazio regional project *Sapientia* (FILAS-RU-2014-1186).

1 cf. `http://opendefinition.org/`

Much related to open data, *open government* is the governing doctrine that supports the right of citizens to access the documents and proceedings of the government for an effective public oversight. Enabling interested citizens to get more directly involved in the legislative process, making government information available to the public as machine readable open data, can facilitate government transparency, accountability and public participation. Opening up official information can support technological innovation and economic growth by enabling third parties to develop new kinds of digital applications and services.

Several national governments have created sites to distribute a portion of the data they collect, e.g., the European Commission has created two portals for the European Union: the EU Open Data Portal[2] giving access to open data from the EU institutions, agencies and other bodies, and the Public Data portal[3] providing datasets from local, regional and national public bodies across Europe. In October 2015, the Open Government Partnership (OGP) launched the International Open Data Charter, a set of principles and best practices for the release of governmental open data formally adopted by seventeen countries (including Italy[4]) during the OGP Global Summit in Mexico.

Despite all these initiatives, access from citizens to such open data is not always as large as expected. Technical issues often inhibit easy access from citizens, if no specific user-friendly applications are built on top of such open data for easy access and navigation. In this discussion paper, we present the disruptive idea of adopting chatbots as user-friendly access and querying method to open data. Nowadays persons are used to chat with friends over popular applications (e.g., WhatsApp or Facebook Messenger), and the typical interaction is indeed based on the paradigm ask – get a response. Citizens accessing open data would appreciate the same paradigm in querying the data, for which a chatbot can be a much more natural way of interaction than traditional web applications.

Developing chatbots over open data poses many challenges, such as interpreting the natural language adopted by users in querying the dataset, and translating into effective queries over the dataset. In this paper, we present a prototype of such a system built using the cognitive platform by IBM, namely Bluemix and related APIs, in order to evaluate the technical feasibility of the proposed idea.

The following of this paper is organized as it follows: Section 2 provides some background information and relevant work; Section 3 describes the architecture used to build a chatbot over the Open Cantieri dataset, published by the Italian Ministero delle Infrastrutture e Trasporti at `http://opencantieri.mit.gov.it`, by using a cognitive platform, and Section 4 describes the realization aspects. Finally Section 5 concludes the paper, by remarking future work, including a user evaluation to assess the usability of the approach, and to prove the argued simplicity of use.

[2] cf. `http://data.europa.eu/euodp/en/data/`
[3] cf. `http://publicdata.eu/`
[4] cf. `https://www.opengovpartnership.org/country/italy`

2 Background and relevant work

Chatbots are computer programs able to hold up a conversation with a user, either in textual or vocal form. Given the growing complexity of information systems, chatbots are specifically designed to support the user interaction and to make it as natural as possible. They do not only represent a faster and more natural way to access information, but they will also be a significant key factor in the process of humanizing machines in the near future [1,2].

In order to correctly and efficiently design a chatbot, many techniques have to be not trivially combined, including pattern matching, parsing, artificial intelligence, machine learning, and ontologies. There are numerous approaches and methodologies proposed for this; in this work, we followed the approach that divides the chatbot architecture in three parts: *responder*, *classifier* and *graphmaster* [1]. The responder is the interface by which users access the system. It is responsible for taking the input and validate the output. The classifier is located between the responder and the graphmaster. It is dedicated to normalize the input to pass to the graphmaster and processing the output coming from the latter (e.g., interacting with a database). Finally, the graphmaster is the agent responsible to elaborate the correct output to the corresponding input. It represents the pattern matching element of the chain.

Tim Berners-Lee suggested a 5-star deployment scheme for open data[5], being a star when an organization makes data available on the Web (whatever format) under an open license, 2 stars when it makes data available as structured data (e.g., Excel file instead of image scan of a table), 3 starts when data are available in a non-proprietary open format (e.g., CSV as well as of Excel), 4 stars by using URIs to denote things, so that people can point at them and 5 stars when data are linked to other data to provide context [3]. Technologies supporting this vision of linked open data are those ones commonly referred as Semantic Web, including RDF/RDFS, OWL (ontologies) and SPARQL (for querying). In Italy, the AgID - Agenzia per l'Italia Digitale, publishes every year guidelines for Public Administrations on how to publish their data as open, including a model for metadata consisting of 4 levels [6]. In this work, we have built our prototype on the basis of a dataset which can be ranked at most at level 3 of the above classification.

During the last years, some attempts to apply chatbots to query and retrieve data have been made. In [4], a chatbot was constructed on top of some open data. Here the first step is to extract plain text from documents stored as PDF files by employing an optical character recognition (OCR) software. At this point, a set of possible questions about the extracted contents were constructed using a "Overgenerating Transformations and Rankings" algorithm, which was implemented using the question generation framework presented in [5]. Finally, the

[5] cf. `http://5stardata.info/en/`

[6] cf. `http://www.agid.gov.it/agenda-digitale/open-data`

matching patterns, essential to the chatbot's answering capability, are defined through Artificial Intelligence Markup Language (AIML).

Authors in [6] presents a system, called OntBot, which employs a mapping technique to transform an ontology into a relational database and then uses that knowledge to construct answers. Therefore, likewise our solution, OntBot does not need to handwrite all the knowledge base that stands behind the system. The main drawback of traditional chatbots, implemented for example through AIML, is the fact that the knowledge base has to be constructed ad-hoc by handwriting thousands of possible responses. OntBot, likewise our system, does not construct answers by looking for a matching one inside the database. Instead, it retrieves information from the database, which will be then used to build up the response.

3 Case study and proposed architecture

Open Cantieri, offered by the Italian Ministero delle Infrastrutture e dei Trasporti (MIT), is an open, complete and up-to-date repository about the realization status and history of the public infrastructures. All the available data are generated and published by public sources. Open Cantieri offers a unified platform, with specific views, in which all these different datasets are collected together. The platform is a collection of open data in a very raw form: datasets can be downloaded as single CSV files, sometimes grouped in archives. Very often, unfortunately, different files do not employ the same keys to represent concepts (e.g., cities are represented using their code in some files whereas their names is used in others) and manual mapping between different representations was needed. More generally, the files do not follow any standard on field names and reported values.

Figure 1 shows the architecture of the proposed solution. The user interface to the system is implemented as a Facebook Messenger application. The back-end core of the system is deployed on the IBM Bluemix cloud computing platform. In particular, a Java Server Page (JSP) handles the requests and constructs the corresponding responses by orchestrating two Bluemix service instances: *(i)* an instance of Watson Conversation, specifically created and trained, which is responsible for processing, and *(ii)* an instance of Compose for MySQL, which handles the connection and the query to the back-end database. In particular, the user interacts through the chat interface, e.g., issuing a question as "How much money have been invested in public infrastructures in the south of Italy in 2015?" (step 1). The sentence is forwarded to the JSP, which handles it in order to construct the appropriate response (step 2). The Watson Conversation instance receives a request constructed starting from the user's input, and generates the corresponding response (step 3). According to the provided response, the JSP page defines an SQL query to be issued to the database through Compose (step 4). Once all the elements needed to construct the output are collected, the JSP will proceed to generate the response to be shown through the Facebook Messenger chat.

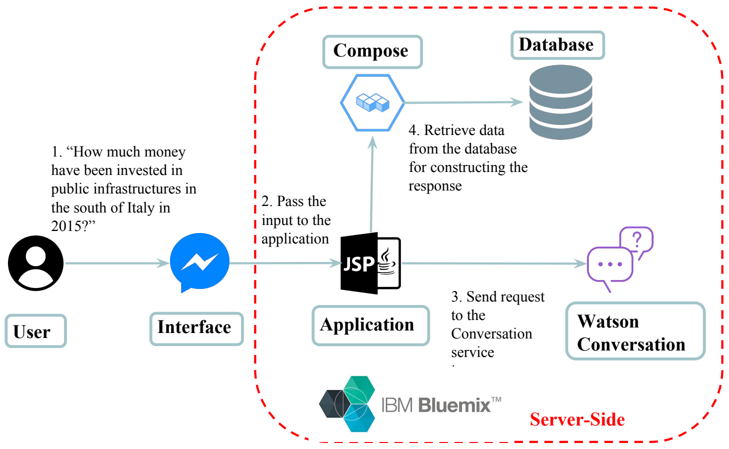
Compose
Database
4. Retrieve data from the database for constructing the response
1. "How much money have been invested in public infrastructures in the south of Italy in 2015?"
2. Pass the input to the application
JSP
User
Interface
Application
3. Send request to the Conversation service
Watson Conversation
IBM Bluemix™
Server-Side

Fig. 1. Architecture

In order to generate the response, Watson Conversation performs the following operations (the reader should also refer to following subsections): intents and entities extraction; verification of which node, within the Dialog Tree, has conditions satisfied by these information; and, finally, return of the nodes response. In our specific case, the intent is triggered by "How much money", which is associated to the intention of knowing the investment amount about the highway management, regarding the intent "#Investment". The entities are: "South of Italy", which is a specific value of the entity "@Geographical_Region" and "2015", which is a value of "Year". In the Conversation response, the application is able to find all the essential information for constructing the output. First of all, a flag, called "DB_Search", is retrieved from the Conversation response in order to understand if a database search is required. This is achieved by constructing the SQL query, starting from the information obtained from Watson Conversation, and by send it to Compose for MySQL, which will retrieve the desired data from the database. The SQL query is specifically constructed with the "Text" included into the Conversation response, which is one of many JSON variables returned with the response itself. Once all the elements needed to construct the output are collected, the application will proceed to generate the users output and send it back through the interface.

In the following, we describe the single components.

3.1 Watson Developer Cloud and Conversation

IBM Watson Developer Cloud (WDC) offers a set of services for developing Cognitive Applications, which consists of programs able to take advantage of the most modern technologies in artificial intelligence, machine learning, and natural language processing. Each WDC service provides a REST API for interacting with it, and most of these services also includes Software Development Kits (SDKs) for various programming languages. In our work we used the Java one.

Inside IBM WDC, the Watson Conversation service allows to create an application that understands natural language input and uses machine learning to respond to users in a way that simulates a conversation between humans. When an instance of this service is created, it is able to contains several *workspaces*. A workspace is a container for all the artifacts that define the conversation flow and it is responsible for the natural language processing operations. A workspace includes the following elements:

Intents. An intent represents the intention and the purpose behind user input. It could be associated with the "goal" the user wants to achieve with every request and thus it is important to define one intent for each type of user request the application has to support. Each intent is prefixed with the character "#" and, during its creation, the developer is encouraged to provide "positive examples", in order to allow the system to construct the corresponding model. A positive example is a sentence that clarifies the way in which the intent could be presented to the system. By collecting at least five positive examples, the instance of the Conversation service will be able to perform a deep learning process, which will train the service itself to recognize that specific intent. The most important fact is to distinctly define each intent from the others. The "borders" between intents should be clear in order to allows the system to correctly recognize them inside user requests. If there is the need for an intent to have more "interpretations", depending on a particular user request, it is possible to make use of another Watson Conversation's element: the Entity.

Entities. An entity is an element, corresponding to a term or an object, that could be used in order to better specify the intention behind a user request. They are frequently used in combination with intents to increase their range of possible interpretations and meanings. Each entity is prefixed with the character "@" and is associated with a set of values. Each value of a specific entity represents an object or a term that belongs to the same category defined by the entity itself. In this way, an entity called "day of the week", could be include values as "Monday", "Thursday", etc. Associated with each value, the developer has the possibility to insert synonyms, in order to be sure that the system will recognize a specific value of the entity even if the user provides it with a different word.

Dialog. The dialog represents the flow of the conversation, divided in several branches, which defines how the application responds when it recognizes the defined intents and entities. The dialog is composed by several nodes, structured in a tree-like graph. At a very basic level, each node is defined by two main elements: the condition and the response. When the condition, composed by elements like intents and entities, is satisfied, the node is considered "activated" and hence its response will be returned as output. The response could be a sentence, another node, or it can be defined by the developer. In order to maintain the state of the conversation through each interaction with the user, the instance service keeps a JSON variable called "context". In this element there are several variables, which can be customized by the developer, and, among them, there is the "Dialog Stack", which contains the stack of all the nodes visited during the conversation and the first one, the "Contextual Node", is the ID of the node that should be returned when the user will start another interaction, within the same session, with the instance of the service Watson Conversation.

3.2 Compose for MySQL

IBM Compose for MySQL is a platform able to simplify the maintenance and the management of a MySQL database; it automatically executes some common operations as backups, scaling and health check. Even though the database can be accessed as a normal MySQL database, the main benefit offered by Compose is that no management aspects (such as security issues or scaling) has to be manually handled.

4 Realization aspects

As seen in the previous sections, at a certain point of the interaction with the user, the system (through the Compose component) requires to retrieve data from a database in order to build answers. As stated above, the Open Cantieri dataset does not follow any standard, thus a database schema able to rationalize the information contained in the different CSVs has been defined. The result of this operation is a schema that does not match anymore in terms of tables and columns with the original file, making it necessary to proceed to an ETL (Extract, Transform and Load) operation.

Once the database was set up, we had to configure the Watson Conversation service to be able *(i)* to understand user requests, *(ii)* to find out if a database search is needed to fulfill the request, and *(iii)* to present a response template to the user. The response template is filled by the JSP using the data retrieved from the database through the Compose component.

In order to define and create a Watson Conversation instance, we need to define intents and entities useful for our purposes. An intent, in our case, represents an argument the user is interested to, e.g., the highway management or the airport system. In order to correctly define them, we have collected all the possible ways in which a user could refer to them, and then we passed these ones as positive examples in the intent's creation process. An entity, on the other hand, corresponds to the values that may concern a specific intent, e.g., the concessionaire societies for the highway management or the airports of the airport system. All these elements were then used in order to construct the dialog of the Watson Conversation instance. Here, we had to figure out all the possible questions the user might ask and the ways in which he might do it. The user may, for example, specify an argument, and then ask for more specific data about it through other questions. He may specify, as argument, the highway s management and then asks for the name of all the concessionaire societies. The user may, at anytime, specify a new argument or asks for more questions, in a human-like conversation. The system can also recognize when the user insert an invalid input and it will help him to correct it.

5 Concluding remarks

We have presented our preliminary idea of combining a chatbot with open data. It involves the employment of several novel instruments and services that are increasingly employed by the researchers and practitioners involved in the development of smart services. Our intent was to take advantage of these new technologies in order to make something new, able to improve the accessibility of open government data.

Open data should be accessible to the public; with our prototype, we would like to showcase a new mean to consult them, in such a way that allows the user to easily retrieve and analyze them.

Future work will include an extensive validation of the approach on a sample of users. Additionally, structured data represents only a face of government complexity. Next steps will include automatic analysis of procedures in order to provide users with a mean to explore bureaucracy in a simpler manner. The conjunction of structured data with unstructured ones may provide public administrations a useful tool to turn open data into something directly usable by citizens.

References

1. S.A. Abdul-Kader, J. Woods. Survey on Chatbot Design Techniques in Speech Conversation Systems International Journal of Advanced Computer Science and Applications, 6 (7), 2015, `https://thesai.org/Downloads/Volume6No7/Paper_12-Survey_on_Chatbot_Design_Techniques_in_Speech_Conversation_Systems.pdf`.
2. Y.P. Yang. An Innovative Distributed Speech Recognition Platform for Portable, Personalized and Humanized Wireless Devices Computational Linguistics and Chinese Language Processing, 9 (2), pp. 77-94, 2004.
3. C. Bizer, T. Heath, T. Berners-Lee. Linked DataThe Story So Far. International Journal on Semantic Web and Information Systems, 5 (3), pp. 122, 2009.
4. L. Pichponreay, C.H. Choi, W.S. Cho, J.H. Kim, K.H. Lee. Smart Answering Chatbot based on OCR and Overgenerating Transformations and Ranking. Proc. ICUFN 2016, IEEE, DOI: 10.1109/ICUFN.2016.7536948
5. M. Heilman, N.A. Question Generation via Overgenerating Transformations and Ranking. Language Technologies Institute, Carnegie Mellon University, Technical Report CMU-LTI-09-013, 2009, `http://www.cs.cmu.edu/~ark/mheilman/questions/papers/heilman-smith-qg-tech-report.pdf`.
6. H. Al-Zubaide, A.A. Issa. OntBot : Ontology based ChatBot. Proc. ISIICT 2011, IEEE, DOI: 10.1109/ISIICT.2011.6149594