



Automatic Generation of Conversational Interfaces for Tabular Data Analysis

Marcos Gomez-Vazquez
marcos.gomez@list.lu

Luxembourg Institute of Science and
Technology
Esch-sur-Alzette, Luxembourg

Jordi Cabot
jordi.cabot@list.lu

Luxembourg Institute of Science and
Technology
Esch-sur-Alzette, Luxembourg
University of Luxembourg
Esch-sur-Alzette, Luxembourg

Robert Clarisó
rclariso@uoc.edu

Universitat Oberta de Catalunya
Barcelona, Spain

ABSTRACT

Tabular data is the most common format to publish and exchange structured data online. A clear example is the growing number of open data portals published by public administrations. However, exploitation of these data sources is currently limited to technical people able to programmatically manipulate and digest such data. As an alternative, we propose the use of chatbots to offer a conversational interface to facilitate the exploration of tabular data sources, including support for data analytics questions that are responded via charts rendered by the chatbot. Moreover, our chatbots are automatically generated from the data source itself thanks to the instantiation of a configurable collection of conversation patterns matched to the chatbot intents and entities.

CCS CONCEPTS

• **Human-centered computing**; • **Software and its engineering** → **Domain specific languages**; • **Information systems** → *Extraction, transformation and loading*; • **Computing methodologies** → *Natural language generation*; **Natural language processing**;

KEYWORDS

NLP, Chatbot, Data analysis, No-code, Code generation

ACM Reference Format:

Marcos Gomez-Vazquez, Jordi Cabot, and Robert Clarisó. 2024. Automatic Generation of Conversational Interfaces for Tabular Data Analysis. In *ACM Conversational User Interfaces 2024 (CUI '24)*, July 08–10, 2024, Luxembourg, Luxembourg. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3640794.3665577>

1 INTRODUCTION

Tabular data, consisting of samples (rows) and features (columns), is a prevalent data type in digital technology, increasingly used in *open data* published by public administrations¹. Despite its wide use, there's a significant lack of tools that allow non-technical users to easily explore this data, which limits the public advantage from

¹Just the EU portal <https://data.europa.eu/> registers over 1.5M datasets

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CUI '24, July 08–10, 2024, Luxembourg, Luxembourg

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0511-3/24/07

<https://doi.org/10.1145/3640794.3665577>

open data initiatives. Conversational User Interfaces (CUIs) such as chatbots and voicebots could improve the accessibility of tabular data [3]. Until now, chatbots for tabular data are either manually created (an option that is not scalable) or completely relying on general purpose LLMs (with limited capacity, especially for larger datasets, and with a risk of generating wrong answers, see Section 5).

This paper introduces a scalable, no-code tool that automatically creates chatbots for tabular data based on a schema inferred from the data itself. Such schema can be optionally enhanced by the user, or automatically with the help of a Large Language Models (LLMs) if needed. Our generated chatbots can handle a broad range of user intents and incorporate LLMs for generating responses through English-to-SQL translations when the intent corresponding to the user question is unrecognized. The entire setup is managed on our DataBot platform², which supports data import, chatbot management, and interactions via text or voice, providing outputs in tables or graph formats. This innovative approach not only simplifies the exploration and usage of tabular data for users without technical expertise but also offers organizations a convenient, effective method to enhance the value and utility of their data assets.

2 CHATBOT ARCHITECTURE

The architecture we propose for our generated chatbots is depicted in Figure 1. At the core of the bot, there is an intent matching process aimed at identifying the user questions and their parameters. If the bot is able to match the intent and recognize all the mandatory parameters for that specific intent, it will transition to the appropriate state in charge of generating the answer for that particular question. If not, a fallback mechanism is triggered and an English-to-SQL translation is done by a LLM to obtain the best possible tabular answer. Note also that the bot is prepared to be multilingual with minimal work.

2.1 Execution Path for Recognized Intents

Once the chatbot knows the user's intent, it starts the process of generating the answer. This overall process is the same in all intents, though its implementation varies on each intent.

The first step is to analyze the intent parameters (entities). If the extracted parameters are OK (there are no missing parameters or wrong value types) the bot moves to the next step of the workflow. When the bot finds confusing or missing parameters, it triggers

²<https://github.com/BESSER-PEARL/databot>

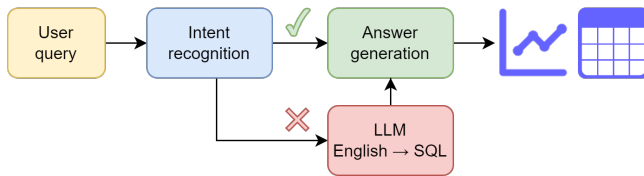


Figure 1: Diagram of the architecture of the generated chatbots.

an interaction with the user to clarify the doubts. If the problem persists, it moves to the fallback state.

When all parameters are OK, the bot proceeds with the answer generation. At this point, the bot knows exactly what kind of question the user is asking (based on the recognized intent) and the variables involved in the question (the recognized parameters). The bot can generate 2 kinds of answers: tables or charts (depending on the type of information the user asks for, either implicitly or explicitly). In both cases, the bot behaves in a deterministic way, building the right SQL queries based on the detected information and reporting it in the desired output format (e.g., generating a pie chart and displaying it in the interactive GUI of the bot).

2.2 LLM Fallback Path

Despite the bot’s best efforts, sometimes it may fail to understand the user’s question, either because it is not one of the questions the tool foresaw in the generation or because it is too far from the training sentences. When this happens, the bot cannot give an exact answer on its own. At this point, it could just tell the user that it was unable to understand the question, but we try to be more useful and add to the bot a powerful, optional and configurable fallback mechanism.

The fallback relies on a LLM to automatically translate the user’s query to an equivalent SQL statement. This approach does not always provide a perfect translation, and therefore, it may generate a wrong answer but it is worth trying as our experiments suggest that users prefer an approximate result (even if potentially wrong) than a plain “sorry” message. Note that the bot always warns the user when answering via this fallback strategy and provides the SQL suggested by the LLM to the user for explainability purposes. The quality of the answers will depend on the selected LLM. As of today, we obtain the best results with GPT-4 [11]. When triggering the fallback mechanism, the bot will: 1 - Send a prompt to the chosen LLM with instructions on the task it must perform (translation of a query in English into an equivalent SQL query) and the data schema in JSON format as additional context, 2 - Receive the LLM-generated SQL query and run it on the bot’s dataset, and 3 - Display the tabular answer together with the SQL query used to obtain it.

3 AUTOMATED CHATBOT GENERATION PROCESS

Figure 2 shows the workflow our tool follows to generate the bots from an initial tabular data source, depicted as a CSV file in the Figure. The generated bots will follow the architecture explained in the previous section. The process is fully automatic, although the

data owner can optionally participate in the data schema enrichment step to generate more powerful bots. This enrichment can also be automated by using LLMs. The next subsections describe in more detail each step.

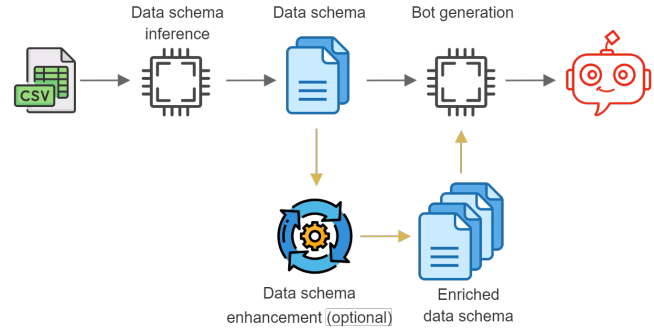


Figure 2: The chatbot generation process.

3.1 Data Schema Inference

To automatically create a bot, we only need one ingredient: a tabular dataset. The dataset must follow a 2D structure, being composed by columns (attributes) and rows (records). Therefore, valid formats for this approach include CSV or XLSX, although other formats such as JSON or XML can be supported as long as they follow a tabular-like structure (e.g., nested attributes are not supported). The size of the dataset may impact the bot performance. The dataset queries will be done with Python’s Pandas library, so the platform scalability strongly relies on Pandas’ scalability capabilities (e.g., to execute several bots on top of gigabytes of data, a certain level of computational power and memory will be necessary to run them)

From the structure of the dataset we will gather the list of columns/fields (with their names). From the analysis of the dataset content, we will infer the data type of each field (numeric, textual, date-time,...) and its diversity (number of different values present in that specific column). Based on a predefined (but configurable) diversity threshold, we automatically classify as *categorical* those fields under the threshold. Categorical fields are implemented as an additional bot entity so that users can directly refer to their values in any question. All this information conforms the metadata the bot will be trained on.

At this point, the chatbot can be already generated. Thanks to the schema inferred from the data source, the chatbot will be able to recognise certain kinds of questions relying on the knowledge extracted from the data (e.g., “Which is the maximum value in X?”, where X would correspond to one of the detected numeric columns’). Think of fields and rows as input and output parameters of the user’s questions the bot must be able to answer, e.g., users can ask for the value in field X of rows satisfying a certain condition in field Y.

3.2 Data Schema Enhancement

Even though chatbots can be automatically created, they could have limited success for some data sources. This could happen, for

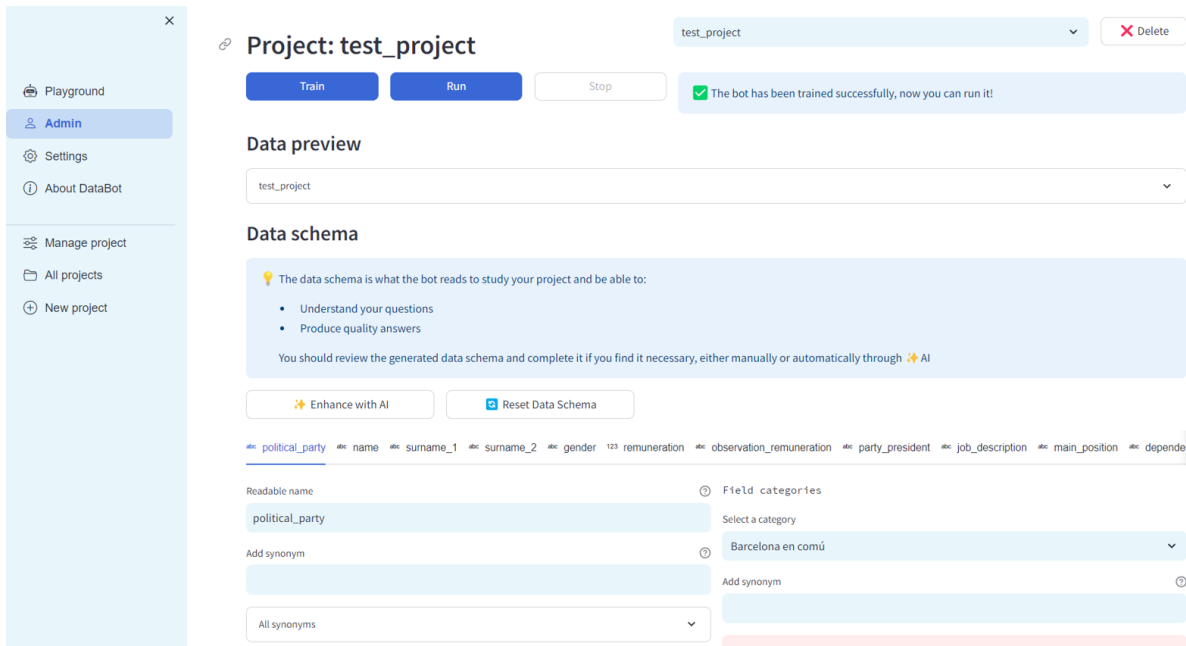


Figure 3: Screenshot of the admin User Interface, where bots can be executed and the data schemas can be enhanced.

instance, if the user uses words very distant to those present in the data schema. Another common problem is the semantics of fields. Some fields may compose an address (street, number, city, etc.) or a date (year, month, day, hour). This is something users could ask about, but they are probably not aware of the internal structure of the dataset and could ask about fields that actually do not explicitly exist as such.

Our bots' philosophy is based on being sure in the answers they provide. Therefore, increasing the bot comprehension or permission to understand what is unknown could derive in a much higher failure rate. This can be considered a limitation but it is also as a safety mechanism. However, we provide the user with the necessary tools to optionally enhance their bots' capabilities by enriching the automatically inferred data schema. As an example, the schema could be enriched by adding synonyms or creating new virtual columns (result of merging fields). These improvements can be done either manually by the bot creator or using a LLM to automate the process, as shown in Figure 3. The data schema model is designed to be easily extensible to add different knowledge components. Therefore, further work on this matter to augment the bot comprehension of the data can be done in the future.

3.3 Bot Generation

The bot generation phase takes the (potentially enriched) data schema and instantiates a set of predefined conversation patterns, gathered, improved and extended via several experiments with users, to generate the actual set of questions the bot will be trained on (*i.e.*, the intents' training sentences). The training phase of the bot will vary depending on the chosen intent classifier component. The main idea behind this is that this component learns the kind of questions it will receive by seeing example (training) annotated

sentences. In NLP, this problem is known as Text Classification. On top of this core component, the generator will add the fallback mechanism and other auxiliary conversations and components to create a fully functional bot.

3.3.1 Intents. The generated bots will contain a set of predefined intents, whose training sentences will be generated from a template bundle and completed with the data schema information³. These intents have been designed to suit as many datasets as possible and to match any dataset query that involves the columns and their values as embedded parameters. These queries mainly (but not only) include any SQL 'select' statement (in an equivalent natural language form), column comparisons, column or value frequencies, etc., regarding the tabular answer intents, and histogram, boxplot, bar, pie or line chart generation (among others) regarding the chart answer intents.

The advantage of this conversational model is that it is easily extensible with further intents allowing the integration of new bot capabilities by just defining the new intent and its proper response.

3.3.2 Entities. The bots contain a set of entities used to recognize relevant elements within intents through their parameters. These entities consist of a set of values, each of them containing an optional set of synonyms. In our context, the parameters the bots must recognize are mainly elements relative to the data they serve, like field names or values. In other words, their content depends on the data content. There are other data-independent entities such as operators (*e.g.*, 'maximum' or 'minimum') or row names (*e.g.*, 'row' or 'entry', though the user can add domain-specific row names, like 'person' or 'employee')

³<https://github.com/BESSER-PEARL/databot/blob/main/src/app/bot/library/intents.json>

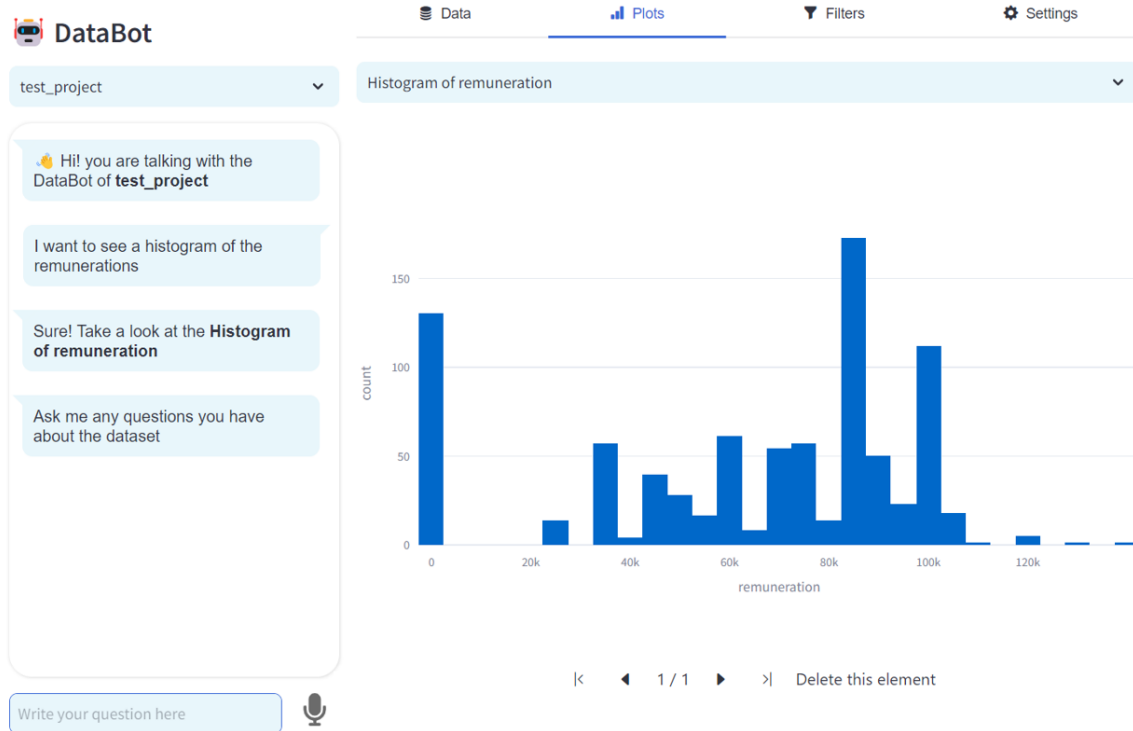


Figure 4: Screenshot of the interactive dashboard showing a graphic answer (a histogram) generated by the bot.

3.4 Using the Generated Bots

Once admin users are ready to generate a chatbot, they can go ahead just by pressing the *Train & Run* buttons to (locally) deploy the bot. It will be available in the playground tab of the platform. The playground is the UI for the chatbot end-user (e.g., the citizen). It offers an interactive dashboard with a chat box on the left side of the canvas, together with a text input box and a voice input button. On the right side, there is the dashboard itself, composed of a set of tabs aimed at organizing the chatbot-generated content and configuration options. It is also possible to create filters, restricting the search space of the bot when generating an answer (e.g., filter by gender, before some date or with some numeric field lower than a threshold). Figures 4 and 5 show two different interactions with a chatbot that generated graphical and tabular answers, respectively.

4 TOOL IMPLEMENTATION

All the tool components are written in Python. The UI relies on Streamlit, a Python library for GUI development. The generated chatbots rely on the BESSER Bot Framework⁴, a Python open-source bot development tool for its runtime execution. All the data management is performed with Pandas, the *de facto* Python library for this type of task. Finally, for all the LLM-based tasks (data schema enhancement, intent classification and English-to-SQL fallback) we rely on OpenAI’s GPT-4 through its API. All the source code of the DataBot platform is freely available on GitHub.

⁴<https://github.com/BESSER-PEARL/BESSER-Bot-Framework>

The design of the conversation patterns has been the result of three preliminary experiments with datasets from the city of Barcelona⁵, the Catalan Government⁶ and the LIS Cross-National Data Center in Luxembourg⁷, where we deployed a chatbot for each dataset and gathered the user interactions (both from the data owners and the citizens) to refine the intents the bots should recognize.

5 RELATED WORK

In this section, we compare our approach with other works focusing on the exploration and exploitation of tabular data by non-technical people.

A first group of works (e.g., Socrata⁸) focuses on the generation of charts and interactive dashboards [17] to help users filter and view the data they want. However, while really useful to see trends and global data perspectives, these works cannot be used to answer concrete adhoc questions on specific aspects of the data.

Other approaches opt for a direct English-to-SQL translation when querying tabular data, such as [1, 10, 14]. The major concern with these “uncontrolled” translations is that they can generate wrong queries and therefore come up with wrong factual answers. This latter issue is also the main concern with generative chatbots

⁵<https://opendata-ajuntament.barcelona.cat/data/en/dataset/carrecs-electes-comissionats-i-gerents>

⁶<https://dadesobertes.seu-e.cat/dataset/ge-p-pressupostos-per-programes-detallat>

⁷<https://www.lisdatacenter.org/wp-content/uploads/files/access-key-workbook.xlsx>

⁸<https://dev.socrata.com/>

The screenshot shows the DataBot interface with a chat window on the left and a data view on the right. The chat window shows a user asking "who makes more money?" and the bot responding with a detailed explanation of how it analyzed the data to find the highest earning politician. The data view on the right shows a table with columns for name, surname_1, surname_2, and remuneration. The first row shows a politician named Marta Labata Salvador with a remuneration of 140,000. Above the table, an information box indicates that the data was obtained via an AI-generated SQL query: `SELECT name, surname_1, surname_2, remuneration FROM df ORDER BY remuneration`.

Figure 5: Screenshot of the interactive dashboard showing a tabular answer generated by the bot through the fallback mechanism powered by GPT-4. On the top-right side, there is an information box indicating that the displayed answer has been obtained after running an AI-generated SQL statement, and the actual SQL is also shown.

based on LLMs that could also be used to chat with the citizens. They tend to “hallucinate” and invent facts, which is something too risky for a public-facing chatbot, especially for a government administration [15].

More similar to our efforts, a couple of Proof of Concepts of intent-based chatbots used for open data have been published [4, 13]. Both bots were manually created. This is in contrast with our approach where bots are automatically generated. An exception is [5] where the bot generation is semi-automated but it requires a mandatory and extensive annotation process while we focus more on a scalable approach able to generate chatbots with no human intervention if so desired. Generation of chatbots from other types of data sources like APIs [7, 16], web pages [6], knowledge graphs [2] or even software designs [12] has also been explored and some of their ideas could be exploited as well for tabular data. However, we did not find any existing solution for automatic generation of

intent-based chatbots from tabular datasets that we could compare with this work.

Another approach being widely used to query data with LLMs is Retrieval-Augmented Generation (RAG) [9]. While being useful for unstructured data sources like plain text, RAG has some limitations to query tabular data. Some queries may need to retrieve an entire column (e.g., to sum all the values), and RAG’s methodology consists of selecting relevant parts of the data to send them to a LLM as context so it can easily find the answer. This is not the best solution for tabular data since it is not scalable. Furthermore, LLMs still can suffer hallucinations.

To sum up, we believe our approach proposes a novel combination of strategies to mix the best of both worlds (intent-based and LLM-based chatbots) and opens the door to a more massive use of chatbots for tabular data thanks to our automatic generation strategy.

6 CONCLUSIONS AND FURTHER WORK

We have presented a new tool to automatically generate chatbots from tabular data sources to help non-technical users explore this type of data. This is especially useful in the current trend towards more transparency and openness in the public administration, with more and more open data sources released each day. Our chatbots encode a significant number of potential questions users may want to ask the data. Such questions are automatically generated based on an initial analysis of the structure and content of the data source.

As further work, we plan to enrich the training of the chatbots with the use of ontologies. The idea would be to map the data schema to ontological concepts to be able to consider more semantic information in the training. We also plan to extend the set of conversation patterns including questions on the validity, origin and possible biases of the data [8].

ACKNOWLEDGMENTS

This project is supported by the Luxembourg National Research Fund (FNR) PEARL program, grant agreement 16544475. This work has been partially funded by the Spanish government (PID2020-114615RB-I00/AEI/10.13039/501100011033, project LOCOSS).

REFERENCES

- [1] Abhijith Neil Abraham, Fariz Rahman, and Damanpreet Kaur. 2022. Table-Query: Querying tabular data with natural language. *CoRR* abs/2202.00454 (2022). arXiv:2202.00454
- [2] Caio Viktor S. Avila, Wellington Franco, José Gilvan Rodrigues Maia, and Vânia Maria P. Vidal. 2020. CONQUEST: A Framework for Building Template-Based IQA Chatbots for Enterprise Knowledge Graphs. In *25th Int Conf on Applications of Natural Language to Information Systems, NLDB 2020 (LNCS, Vol. 12089)*. Springer, 60–72.
- [3] Marcos Baez, Claudia Maria Cutrupi, Maristella Matera, Isabella Possaghi, Emanuele Pucci, Gianluca Spadone, Cinzia Cappiello, and Antonella Pasquale. 2022. Exploring Challenges for Conversational Web Browsing with Blind and Visually Impaired Users. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, Article 234, 7 pages.
- [4] Iván Cantador, Jesús Viejo-Tardío, María E. Cortés-Cediel, and Manuel Pedro Rodríguez Bolívar. 2021. A Chatbot for Searching and Exploring Open Data: Implementation and Evaluation in E-Government. In *DG.O'21: The 22nd Annual Int Conf on Digital Government Research*. ACM, 168–179.
- [5] Nicola Castaldo, Florian Daniel, Maristella Matera, and Vittorio Zaccaria. 2019. Conversational data exploration. In *Int. Conf. on Web Engineering*. Springer, 490–497.
- [6] Pietro Chittò, Marcos Baez, Florian Daniel, and Boualem Benatallah. 2020. Automatic generation of chatbots for conversational web browsing. In *International Conference on Conceptual Modeling*. Springer, 239–249.
- [7] Hamza Ed-Douibi, Javier Luis Cánovas Izquierdo, Gwendal Daniel, and Jordi Cabot. 2021. A Model-Based Chatbot Generation Approach to Converse with Open Data Sources. In *Web Engineering - 21st International Conference, ICWE 2021 (LNCS, Vol. 12706)*. Springer, 440–455.
- [8] Joan Giner-Miguel, Abel Gómez, and Jordi Cabot. 2022. DescribeML: a tool for describing machine learning datasets. In *Companion of the 25th Int. Conf. on Model Driven Engineering MODELS 2022*. ACM, 22–26.
- [9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
- [10] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing. *CoRR* abs/2012.12627 (2020). arXiv:2012.12627
- [11] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]
- [12] Sara Pérez-Soler, Gwendal Daniel, Jordi Cabot, Esther Guerra, and Juan de Lara. 2020. Towards Automating the Synthesis of Chatbots for Conversational Model Query. In *Enterprise, Business-Process and Information Systems Modeling*. Springer, Cham, 257–265.
- [13] Simone Porreca, Francesco Leotta, Massimo Mecella, Stavros Vassos, and Tiziana Catarci. 2018. Accessing government open data through chatbots. In *Int. Conf. on Web Engineering*. Springer, 156–165.
- [14] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
- [15] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion* 81 (2022), 84–90.
- [16] Mandana Vaziri, Louis Mandel, Avraham Shinnar, Jérôme Siméon, and Martin Hirzel. 2017. Generating Chat Bots from Web API Specifications (*Onward! 2017*). Association for Computing Machinery, 44–57.
- [17] Aoyu Wu, Yun Wang, Mengyu Zhou, Xinyi He, Haidong Zhang, Huamin Qu, and Dongmei Zhang. 2021. MultiVision: Designing analytical dashboards with deep learning based recommendation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 162–172.

Received 18 April 2024; revised 23 May 2024