GeCoAgent: a Conversational Agent for Empowering Genomic Data Extraction and Analysis

PIETRO CROVARI*, SARA PIDÒ*, PIETRO PINOLI*, ANNA BERNASCONI, ARIF CANAKOGLU, FRANCA GARZOTTO, and STEFANO CERI, Department of Electronics, Information and Bioengineering, Politecnico di Milano

With the availability of reliable and low-cost DNA sequencing, human genomics is relevant to a growing number of end-users, including biologists and clinicians. Typical interactions require applying comparative data analysis to huge repositories of genomic information for building new knowledge, taking advantage of the latest findings in applied genomics for healthcare. Powerful technology for data extraction and analysis is available, but broad use of the technology is hampered by the complexity of accessing such methods and tools.

This work presents GeCoAgent, a big-data service for clinicians and biologists. GeCoAgent uses a dialogic interface, animated by a chatbot, for supporting the end-users' interaction with computational tools accompanied by multi-modal support. While the dialogue progresses, the user is accompanied in extracting the relevant data from repositories and then performing data analysis, which often requires the use of statistical methods or machine learning. Results are returned using simple representations (spreadsheets and graphics), while at the end of a session the dialogue is summarized in textual format. The innovation presented in this paper is not only concerned with the delivery of a new tool but also in our novel approach to conversational technologies, potentially extensible to other healthcare domains or to general data science.

CCS Concepts: • **Applied computing** → **Bioinformatics**; **Computational biology**; • **Human-centered computing** → **HCI design and evaluation methods**; **HCI theory, concepts and models**; **Natural language interfaces**.

Additional Key Words and Phrases: Conversational agents, Natural language understanding, Genomic computing.

ACM Reference Format:
Pietro Crovari, Sara Pidò, Pietro Pinoli, Anna Bernasconi, Arif Canakoglu, Franca Garzotto, and Stefano Ceri. 2021. GeCoAgent: a Conversational Agent for Empowering Genomic Data Extraction and Analysis. *ACM Trans. Comput. Healthcare* 1, 1, Article 1 (January 2021), 31 pages. https://doi.org/10.1145/3464383

1 INTRODUCTION

Genomics is the study of whole genome of an organism, i.e., the genetic information encoding for the instructions to regulate the biological processes that govern the life, growth and development of a live being. Its main focus is the investigation of the DNA and how modifications (i.e., mutations) of certain portions of the DNA (e.g., genes) are involved

*These authors contributed equally to this research.

Authors' address: Pietro Crovari, pietro.crovari@polimi.it; Sara Pidò, sara.pido@polimi.it; Pietro Pinoli, pietro.pinoli@polimi.it; Anna Bernasconi, anna.bernasconi@polimi.it; Arif Canakoglu, arif.canakoglu@polimi.it; Franca Garzotto, franca.garzotto@polimi.it; Stefano Ceri, Department of Electronics, Information and Bioengineering, Politecnico di Milano, via Ponzio 34/5, Milan, Italy, 20133, stefano.ceri@polimi.it.

Manuscript submitted to ACM

in disease development. Last decades showed that genomics alone is not enough to explain most of the biological activities or diseases; in this scenario, important studies are concerned with epigenomics, i.e. reversible modifications on a cell's DNA that affect gene expression without altering the DNA sequence.

Human genomics and epigenomics experiments for health management are producing massive data amounts at rates that are presently superior to any other domain. Using this data as a complement to other forms of clinical knowledge allows clinicians to aim for precision medicine and biologists to support discovery in the genomics domain, with emphasis on tumors. Since its inception, genomic research has been looking for solutions able to interpret data, making them useful for their purposes. Bioinformatics tried to fill this need with different tools: an unprecedented number of computational methods (from software, high-performance computing, and cloud-based solutions, to web containers/orchestration platforms) have been created to help biologists and clinicians in their effort to discover knowledge from genomic data and translate these discoveries into new therapeutic strategies. Genomic data is not only massive but also complex (including multiple forms of heterogeneous signals, such as mutations, gene expressions, copy number alterations, transcription factor bindings or histone modifications, and so on). These aspects motivate genomic data management to move towards tertiary data analysis, in charge of combining and centralizing diverse signals; in the first three years of ERC Advanced Project GeCo, we designed and engineered powerful tools (GMQL [1, 2], META-BASE [3], GenoSurf [4]), based on solid abstractions, capable of mastering information extraction and integration.

To fully exploit GeCo computational tools, a bioinformatics background and hands-on attitude towards computational resources are needed, translating research objectives into customized search routines, by identifying the suitable tools to get out actionable genomics information, an issue emerged during a vertical application of the GeCo in predictive medicine for tumors; but biologist and clinicians are overwhelmed by bioinformatics tools. To overcome this hurdle, and make GeCo resources accessible to a broad audience, we have developed GeCoAgent, a fully integrated, user-centered web platform aimed at empowering end-user competences.

GeCoAgent is an ergonomic big data service that was mainly designed for biologists. Among them, in particular we target professionals who have limited know-how in computer science and normally use bioinformatics tools with the support of computer scientists while performing advanced data analysis. GeCoAgent is meant to empower this target group, as well as more tech-savvy clinical researchers and biologists, and make them more autonomous in data analysis tasks. Computer scientists involved in bioinformatics are an additional (secondary) target group for GeCoAgent, as it can empower them in performing data analysis more efficiently.

A unique feature of GeCoAgent is a powerful unique conversational interface. To make GeCo resources accessible to a broad audience, GeCoAgent uses: dialogues to interact with computational tools, an automata-driven conversational agent translated in a chatbot, and a dashboard where several visualizations – progressively built by the system – are shown to the user. A dialogic interface is thought to facilitate the interaction with a complex system, both in terms of time required to accomplish the operations and of minimizing the user's errors. The dialogue between the user and GeCoAgent web-platform integrates, in a continuous flow of interactions: i) *data extraction tasks*, producing an informed definition of the data that are necessary to respond to a specific need, and ii) *data analysis tasks*, where the researcher defines the procedures that support this process (including statistical tools, machine learning or deep learning libraries, as well as typical statistics supported by visualization tools).

At the end of a user session, GeCoAgent releases a complete program for solving the problem, in the form of a Python script that can run independently on the user's machine, and extracts datasets available through downloads from the servers. In this way, results produced by GeCoAgent are reproducible and technology independent. Our platform, with this intelligent assistant, introduces a new paradigm directly addressing the needs of genomics experts (clinicians and

biologists), who often get stuck within a routine of fixed interactions with computing systems; GeCoAgent empowers them, as it directly produces streamlined, cleaner and error-free solutions from requirements. In addition, a user-centred solution reduces the users' resistance towards computational resources with a saving of time and costs.

Paper Organization. In the next Section, we describe previous work on conversational technologies and specifically dwell into the systems which addressed data science processes. Then, we present background material concerning the GeCo project, the frame in which GeCoAgent is developed. In Section 4, we propose the requirements for the conversational agent, by defining the processes of data extraction and analysis and then the high-level tasks that compose the processes. We then discuss in Section 5 the detailed design of the system, which includes the identification of the elementary functions implementing each task and of the finite state automaton used for capturing task-oriented dialogues. Then, in Section 6 we discuss the system's deployment by describing the GeCoAgent Architecture, our use of RASA NLU for understanding task-oriented natural language sentences, and the multimodal Web interface. Finally, we discuss a concrete example of an application of GeCoAgent in Section 7, an empirical evaluation of GeCoAgent in Section 8 and we conclude in Section 9, by mentioning the future directions of our work.

2 RELATED WORK

The role of bioinformatics as an aid to research through computational resources has become more and more important in the last years due to the large amount of data to be processed and analyzed to answer complex biological and clinical questions [5]. Conventionally [6], bioinformatics tools and methods can be categorized into three classes:

- *primary* analysis, dealing with the production of the data itself;
- *secondary* analysis, in which produced datasets are manipulated in order to make them comparable (e.g., quality check, calling of genetic variants, harmonization of data from different sources);
- *tertiary* analysis, whose aim is to integrate heterogeneous data and perform holistic analysis to infer novel biological knowledge.

While primary and secondary are already well-defined disciplines, with many widely used algorithms and protocols [7], tertiary analysis has only recently acquired importance [8]; in the long term, however, we expect it to become the most crucial bioinformatics challenge. Given the increase in complexity, novel human-computer interaction paradigms (such as conversational agents) are also becoming more and more needed to bridge the gap with a growing community of tertiary data analysis users, which includes biologists and clinicians.

2.1 Conversational Agents for Data Science

In the early years of bioinformatics, the available tools were mainly command lines and were not accessible by nonexperts [9]. Nowadays, several design environments, such as OrangeBioLab [10], UCSC Xena [11] or Globus Genomics [12], offer user-friendly interfaces that are more easily accessible by bioinformaticians. UCSC Xena [11] or Globus Genomics [12] are two visual programming interfaces for genomic data: they are easy-to-use data management and analysis platforms. OrangeBioLab [10] is a data visualization tool that helps to manage data mining, preprocessing, predictive modeling, feature scoring on uploaded data. However, they still speak in terms of computations and algorithms; hence they do not bridge the gap with biologists and clinicians when they have no computer science background. Indeed, the users of those platforms are not required to program but need to understand computing; moreover, the number of advanced features offered by these tools is limited.

As stated by Bolchini et al. [13], there is an increasing awareness of the need for usability in bioinformatics applications; yet, not enough studies have been conducted to overcome this problem. Bolchini et al. [13] assert that bioinformatics applications need a more aware human-centered development process, in order to remove the entry barriers (also including heterogeneous knowledge to be acquired).

Conversational Technologies can play a fundamental role in this direction. Conversational Agents are systems that mimic human conversation through the exploitation of technologies such as Natural Language Processing, Artificial Intelligence, and Voice Recognition [14]. At first, Conversational Agent research aimed at creating an intelligence to be indistinguishable from humans and pass the Turing Test [15], as in the case of ELIZA [16], ALICE [17] and PARRY [18]. Thanks to the advancement of the underlying technologies, in recent years, Conversational Technology became extremely popular in the research panorama. Open-source solutions such as AIML [19] and RASA [20] as well as commercial solutions like Google Dialogflow[1], Watson Conversation[2], and Amazon Lex[3] have supported the production of applications based on Conversational Agents in many fields.

Lately, a lot of effort has been spent in understanding how this technology can support the execution of complex tasks [21]. For example, PLOW is a dialogue-based agent that teaches users to autonomously perform general tasks, such as retrieving information from web pages [22]. User studies are showing that these kinds of interfaces are becoming more efficient than traditional ones [21, 23, 24]; recent trend estimations believe that in a few years, many interfaces will shift to a conversational paradigm, as it is preferred by the users [25]. In the design of these applications there is a trade-off between freedom of expression and the domain-specific dimension; restricting the domain in which the Conversational Agent acts improves the convergence and at the same time gives the user more freedom of expression, because the bot is helped by the domain knowledge in correctly interpreting the user's sentences [26].

Dialogic interfaces can be supported by multimodal interaction, as shown in PUMICE [27], a Conversational Interface for End-User Development. Given the ambiguous nature of human sentences, PUMICE exploits a multimodal interface to solve the ambiguities created in the conversation. Users interact through the chat and then exploit a visual interface to clarify their intentions. The user study highlighted the effectiveness of a multimodal interface with respect to the conversation alone, arguing for the effectiveness, naturalness, and lower learning curve of multimodal Conversational Interfaces [27].

Conversational Agents for data science are applications where narrowing the application domain helps in guaranteeing the user's freedom of expression. Conversational tools for data visualization provide a fertile research stream in this field. Many effort have been made to enable users to create visualization through a conversation, such as the ones reported in [28–30]. Tory and Setlur [31] analyze in depth the implications of such an interface, highlighting the importance of a good interplay among the conversation and the visualization. The user studies highlighted the importance of producing the visualizations in traditional interfaces, therefore separated from the conversation, supporting PUMICE's thesis on the importance of mixed interfaces.

Data retrieval is another step of data science pipelines that can be covered by conversational interfaces. Several dialogue interfaces translate a conversation into SQL(-like) languages; some examples are Microsoft English, Precise, and Athena [32–34]. In the bioinformatics domain, BioGraphBot [35] is a Conversational Agent that exploits the ALICE framework [17], allowing users to retrieve data from BioGraphDB, a publicly available graph database based on the Gremlin query language [36]. To use this application, though, the user must have some knowledge of the query language

[1] https://dialogflow.com/
[2] https://www.ibm.com/watson
[3] https://aws.amazon.com/lex/

and be aware of the underlying schema. Maggie [37] is a service-oriented architecture-conversational interface designed to retrieve diverse sets of biological information from BioCatalog. Using Maggie, though, the user is not supported through the conversation flow.

Ava [38] is a Conversational Agent developed to support data scientists to compose data analytic pipelines. To do that, it exploits a template for structuring data science tasks, typically composed from a set of nested operations in which the data scientist's main task consists of inserting some parameters, to create a Controlled Natural Language Interface [39] that unambiguously converts the users' input sentences into an executable Jupiter Python notebook. To support the dialogue's translation, Ava conceptual model relies on a finite state machine that represents the Data Analysis workflow. In this way, the Conversational Agent is able to properly interpret the user input, according to the current state. Experimental validation of the platform showed that the conversational interaction with Ava is significantly more efficient than the traditional notebook interface. On the other hand, Ava system does not allow data exploration, since the underlying state machine provides only linear interactions.

Following Ava's example, Iris Conversational Agent [40] transforms the conversation with the user into executable Python functions. Leveraging on the idea that human conversations are based on a combination of atomic speech acts [41], the authors create a system where individual functions can be nested and combined through the mean of a conversation. While the interaction evolves, Iris conversational engine builds a syntax tree that described the chosen operations, which finally is converted into executable code. Also, Iris's empirical evaluation showed the efficiency of such an interface since the user was able to accomplish the assigned tasks about two times faster. Iris functional structure guarantees a much higher level of freedom of expression for the final user, at the cost of a greater user expertise in Python programming, since the atomic speech acts of the system serve as wrappers for the existing Python functions.

3 BACKGROUND

We designed and developed GeCoAgent on top of the tools provided by the GeCo platform, that is under development since 2015. Such resources are fully dedicated to *tertiary* analysis of genomic data. The two main assets are: a data repository that integrates heterogeneous genomic and epigenomic data from various public sources (driven by a conceptual model to organize such data and make them easily retrievable) and the GenoMetric Query Language, to perform complex queries on such data. A bird's-eye view of GeCo components is presented in Figure 1.

In the remainder of this Section, we present GeCo tools with a particular emphasis on those aspects that have been exploited for the development of GeCoAgent.

3.0.1 GeCo Data Repository. Within the GeCo repository, data are organized by using the Genomic Data Model (GDM) [42], which couples the outcome of a biological experiment with clinical and biological information of the studied sample. More in detail, in GDM, the outcome of an experiment is represented as regions of a reference genome, identified by their coordinates and associated with experimental values, while metadata information is represented as free attribute-value pairs. A portion of the metadata, which has been identified to be the most valuable and frequently used, has further been modeled using a relational schema, namely the Genomic Conceptual Model (GCM) [43], and stored in a relational DBMS. The schema includes four metadata dimensions (as shown in the updated version of the model [44]), i.e., the technique, the biological process, the management of the experiment, and the data organization parameters. Open data are retrieved from different sources by using the data integration pipeline META-BASE, described in [3], which performs data extraction, cleaning, normalization, and ontological enrichment and eventually produces

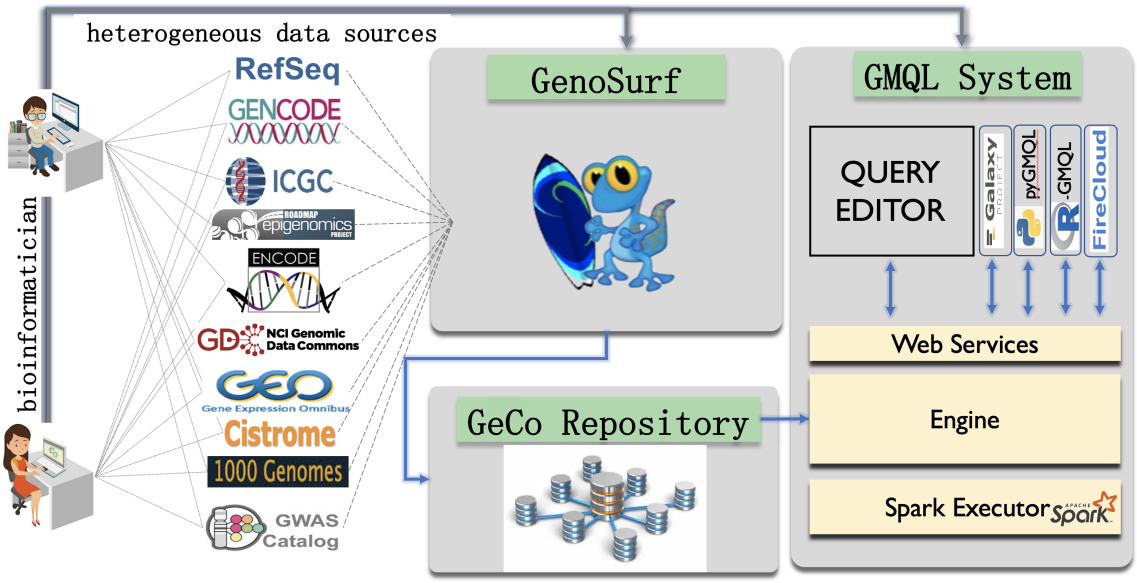


Fig. 1. Bird's eye view of the GeCo platform

both the datasets in GDM format and a well-organized metadata representation in relational format. The GeCo repository can then be inspected by means of GenoSurf, a metadata-driven search engine that exposes the database content by providing users with: i) several drop-downs to select specific values for a set of curated metadata (defined in the GCM [43]); ii) a textual search feature for source-specific metadata key-value pairs; iii) a table of results matching the set filters, updated at each change.

3.0.2 GenoMetric Query Language. Genometric Query Language (GMQL, [1]) is a high-level language for querying genomic datasets, organized using GDM. From an abstract view, the language is an algebra over datasets consisting of region data and metadata, thereby producing result datasets which carry the metadata of the input datasets from which they depend; the building blocks of the language include classical as well as domain-specific algebraic operations. This language organization maps very well to high-level data abstractions, which are used in GeCoAgent to express the semantics of computations, disregarding algorithmic or formatting details, which are typical of bioinformatics languages.

4 REQUIREMENTS ANALYSIS

Bioinformatics tertiary analysis is a complex process; to get to results, researchers alternate hypothesis formulation and testing, and often slightly modify them in an incremental fashion, until they reach scientifically acceptable results. With this purpose, the interaction captured by our conversational agent should not be limited to understanding instructions given by the user to the system; the agent should be able to understand the results of the analysis, the potential following steps, and to formulate proper conclusions.

GeCoAgent must enact a solid underlying process that, at every step, is aware of what users can or cannot do and of which tools and visualization support can facilitate their actions at best, by interpreting the user's intents

and accordingly choosing its response. Detailed requirement analysis is fundamental to capture such a process. To this end, we analyzed and elicited first the requirements of the process (see Section 4.1), then the requirements of the conversation (see Section 4.2). The former requirements stem from our experience of tertiary analysis, that we have gathered by observing how GeCo tools were used by hundreds of users; they represent the process knowledge that GeCoAgent should possess in order to be able to perform an effective tertiary analysis. Instead, conversation requirements represent the typical interactions that a biologist or clinician would naturally perform in order to drive the process without knowing the computational abstractions of a classical workflow-driven process. Thus, they describe the typical knowledge gap that must be covered by GeCoAgent to bring on board users with limited computational background.

4.1 Process Requirements

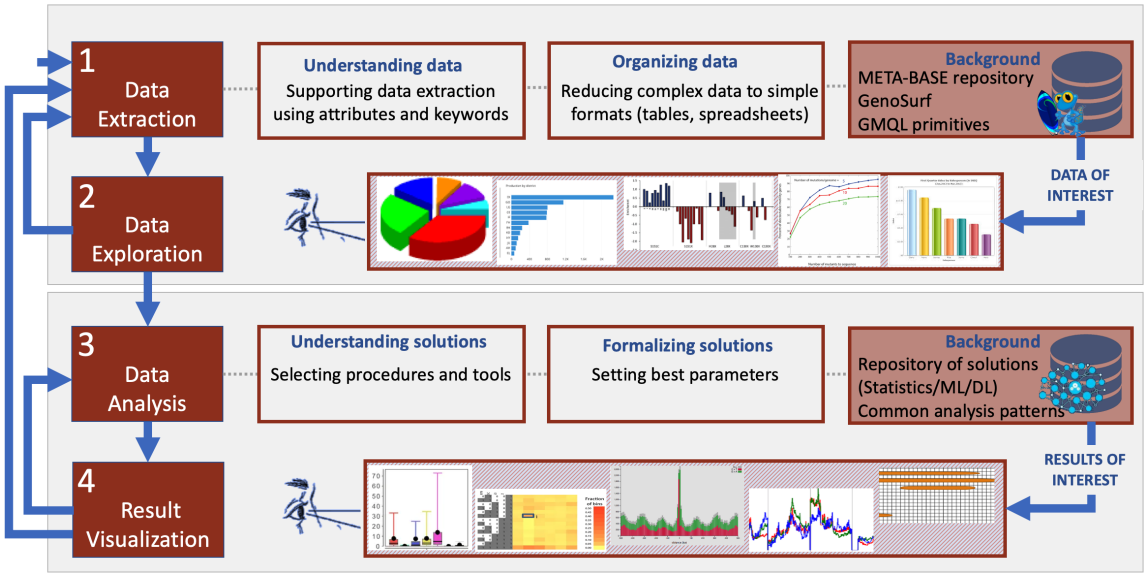


Fig. 2. Overall tertiary data analysis process adopted by GeCoAgent, which consists of data extraction followed by data analysis. Data extraction includes the two macro phases of defining objectives, for constructing the "universe of interest" out of a wider information basis, and of inspection of that universe by means of data visualization tools. Data analysis includes the two macro phases of defining the data analysis procedures out of a repository of techniques, and then inspecting the results produced by those techniques, typically through data visualizations.

4.1.1 Phases definition. The overall process supported by GeCoAgent can be appreciated in Figure 2, where four phases are identified in a typical genomic data-extraction-and-analysis pipeline. The phases resulting from our high-level conceptualization are concerned with data extraction, exploration, analysis, and visualization.

For extracting the data of interest, the researcher should *define objectives*: which data, how organized, how retrieved. As a result, users define an *universe of interest* that can be further explored and evaluated. Such a second phase consists of *inspecting the obtained universe*, taking advantage of many possible statistical qualitative/quantitative visualization techniques. Once the data is extracted, it can be analyzed. The third phase involves *defining an analysis procedure*, i.e., understanding suitable procedures/tools, appropriate parameters to be set, taking advantage of an existing repository

of methodologies (e.g., Statistics, Machine Learning, and Deep Learning libraries) and commonly employed solutions. When the *result of interest* has been generated, the fourth phase of *result inspection* can be performed with the support of other visualization tools.

As it can be observed in Figure 2, various paths can combine the four phases in many ways; the user may first observe the whole universe to get inspiration from all the available data and their interactions, then focus on a specific portion of it; or instead select datasets one by one and evaluate their characteristics. During analysis, algorithms and parameters are progressively adapted until the user is confident with results; at any stage, the user can also decide that the initial datasets do not fit the needs of the analysis anymore and thus go back to the data extraction phase and re-iterate the process.

4.1.2 Tasks definition. We next identified tasks, i.e. recurrent patterns of interaction that are used for data extraction or for data analysis. Table 1 summarizes exemplary tasks of the such activities; tasks are selected so as to cover a significant number of requirements, while at the same time being orthogonal and thus easily composable.

Table 1. Listing of data extraction and analysis tasks, i.e., recurring patterns in GeCoAgent processes.

Extraction tasks	Analysis tasks
E1: Extract relevant regions from a dataset	**A1**: Identify outliers
E2: Merge all regions from multiple samples of a dataset	**A2**: Feature selection or imputation
E3: Pair regions from two datasets at minimum distance	**A3**: Patient/Region Classification
E4: Count experimental regions overlapping with a reference dataset	**A4**: Regression/Survival Analysis
E5: Combine samples from two datasets	**A5**: Network Analysis
E6: Exclude samples from a dataset which intersect another dataset	**A6**: Co-occurrence or mutual exclusion analysis
E7: Extract confirmed regions from a dataset	**A7**: Pattern matching
E8: Count experimental regions overlapping confirmed experiments	**A8**: Mutation Analysis
E9: Build new properties for each region of a dataset	**A9**: Gene Enrichment Analysis

For what concerns data extraction tasks, we adopted a top-down approach, leveraging on the experience that we have developed by using the GeCo framework, which in turn is the result of our interaction with bioinformaticians who routinely perform tertiary data management. Examples include the merging of all regions of multiple experiments into a single sample (file), the pairing of experimental regions when they are at minimum distance, the counting of experimental regions (e.g. mutations) overlapping with a reference dataset (e.g., genes), the extraction of confirmed regions (e.g., regions which overlap in at least a given number of experiments).

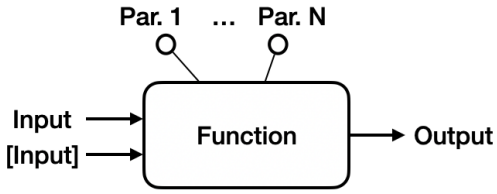
For what concerns data analysis, we adopted instead a bottom-up approach, by inspecting the analysis tasks that were most frequently adopted in tertiary data analysis, normally implemented by using off-the-shelf libraries written in Python and R. We identified various general statistics and machine learning tasks that apply to genomic datasets as to any other data science datasets, such as the identification of outliers or feature selection or imputation. We also identified some tasks that are domain-specific and characterize tertiary genomic analysis, such as mutation analysis and genomic pattern matching.

4.2 Conversation Requirements

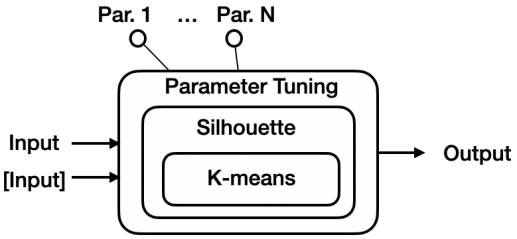
In order to elicit the desired characteristics of the conversational interface of GeCoAgent, we performed semi-structured interviews with 8 researchers who have experience in bioinformatics tertiary analysis. The interviews started with general questions about their research goals during tertiary analysis and progressively dug more deeply into various

aspects: the flow of reasoning researchers would follow to meet their goals; the types of high-level tasks they would need to perform; the kinds of questions they would ask to themselves at the various steps; how they would formulate such questions to a tool like GeCo, and what they would desire the system to offer as a reply; the degree of (non-)linearity of the process, with the identification of the moments in which they might need to "go back" to a previous step in order to redo, undo, or change what they have previously done. By clustering the interviews transcripts and the sticky notes eventually filled by the interviewed researchers, we identified the main themes; we then articulated each theme in terms of conversational requirements for the Conversational Agent and the whole conversational interface, which informed the design of GeCoAgent as reported in the next sections.

- *Process Awareness.* The Conversational Agent should be aware of the general process of tertiary analysis; at every point of time it should know what is the current step of the process, and exploit this contextual knowledge to better understand the user, to generate more accurate utterances, and to suggest proactively what to do next (e.g., speaking about the possible supported actions). The conversational flow is not rigidly pre-defined a priori, but the dialogue is dynamically composed by keeping the process context into consideration. In addition, the dialogue is parametric in the sense that data and functions the Conversational Agent can "speak about" at each point of time are dictated by the current context of the process. Additionally, the Conversational Interface as a whole should make evident, through textual or visual clues, the current context of the process, to orient the user and to highlight the underlying context-driven nature of the conversation.
- *Discontinuity in the research process.* Bioinformatics tertiary analysis is a strongly non-linear research process and therefore its execution results into an intricated topology of interactions and operations. Initially, goals might be open or ill defined, and their operationalization might involve a number of back-and-forth steps with several trial-and-error situations. The production of high-quality analytic results is very often the sum of many little improvements and adjustments at different steps of the process general pipeline. As a consequence, the conversational agent must enable users to go back to a previous step, undo, or make modifications to, previous decisions, and eventually resume the process; it should also provide suggestions (see previous point) that open up multiple alternatives to the user, but could be withdrawn at any time.
- *Strong modularity.* Most research processes are obtained as the composition of the same set of operational steps that must be reapplied many times, possibly with different choices of input variables and/or selected datasets. As a consequence, the user should have the freedom of composing modules according to needs, and reusing them, while the Conversational Agent should be able to recognize such modularization process and use modules as conversational primitives.
- *Multiple dialogic strategies.* The Conversational Agent should be able to sustain multiple styles and strategies of dialogue depending both on the context (see above) and the user's intention "at large". For example, when the researcher has a clear requirement in mind (e.g., which could be detected because she has followed a forward flow so far and expressed precise questions), the conversational style of the Agent should be the one of an an executor that interprets user's requests of data and operations and translate them into system functions, with little dialogic interaction. Instead, when the researcher is uncertain about what to do, e.g., because she is still framing the problem space or refining its exploration goals, or she does not understand the platform's behavior), the Conversational Agent should structure the conversation in order to play the role of a maieutic tutor, engaging the user in a more verbose and multistep dialogue that helps her to shape requirements, or guides her progressively through the process.



(a) Structure of a function. It has one or two data inputs, one or more parameters and a data output.



(b) Structure of a higher-order function for data analysis. Parameter tuning is computed on top of the Silhouette function that in turn is applied to K-means, a domain-independent data clustering function.

Fig. 3. Structure of simple and high-order functions.

- *Multimodality.* Given the visual nature of many operations and outputs involved in tertiary analysis, any interface supporting this process includes *data visualization features* as well as functions on such visualizations. In addition, there are a number of simple, easy to understand operations that might be done more naturally through few clicks on a conventional GUI instead of asking the Conversational Agent to perform then. The Conversational Agent operates therefore in a multimodal interaction environment; it should be able to understand the interplay between GUI-based user interactions and conversation-based interactions, mastering the dialogue content and flow accordingly.

5 DESIGN

Without loss of generality, we model a tertiary genomic research as a process consisting of steps of *data extraction*, *data exploration*, *data analysis* and *result visualization*. Thus, the process is best described by a workflow language, able to represent each step as well as their interconnections, resulting in a directed graph where each node is a *function*, i.e., an atomic operation, and the edges indicate their dependencies and temporal precedence. We first define the functions, then their interconnections to form a workflow.

5.1 Functions

In Section 4.1.2, we defined the most common tasks, i.e. the recurrent high-level operations that are computed during a tertiary analysis. Even if tasks are very well suited to describe the research process, their level of abstraction is too high to describe the process from the operational point of view. In fact, tasks are not atomic, but composed by multiple operations chained in a pre-determined order. On top of that, many operations composing the tasks are not exclusive for some task, but repeated in many of them. For example, the extraction task E1 "Extract relevant regions from a dataset" and the task E2 "Merge all regions from multiple samples of a dataset" contain both the selection of the samples. As a consequence, we have to stoop to a lower level. Thus, we now define *functions*, which are the atomic operations that compose the tasks and, more generally, the tertiary analysis research pipeline.

From the semantic point of view, we can categorize functions according the role they play into the four steps of the process:

- *Data extraction functions* are used to retrieve specific portions of data from the repository and to elaborate the extracted information by means of classical relational algebra operations (e.g., selection, projection, grouping) as well as bioinformatics specific ones (e.g., mapping, binning). Data extraction functions are orthogonal, expressive enough to grant the possibility of generating arbitrary processes, and concise enough to allow the composition of a compact workflow, suitable to be designed by means of a conversational agent. To this aim, we mainly adopted the semantics of the operators defined by GMQL [2], whose operators work on GDM datasets and elaborate both genomic data and associated metadata. We also include in this class the Pivot functions; they transform data in GDM format into flat tables (or Excel files), using parameters to define which pieces of information have to be represented as rows and columns. The result of a Pivot function concludes the Data Extraction phase and produces the *Universe of Discourse*, i.e. tables that can be visualized and used in data analysis operations.
- *Data exploration functions* are used for summarizing and visualizing extracted data in tabular format. Summarization functions allow to compute some statistics on the data, e.g., counting, grouping, and averaging. Visualization functions are used to plot data characteristics in the form of charts (e.g., histograms, piecharts, boxplots).
- *Data analysis functions* are either domain-independent, e.g. machine learning methods and algorithms, as well as validation functions for measuring the data analysis performance, and domain-specific, e.g. algorithms which embed knowledge about genomics. As illustrated in Figure 3b, they may be built through higher-order composition, including a specific data analysis function encapsulated within a validation method, in turn, encapsulated within a parameter tuning function.
- *Result visualization functions* support the visualization of the results; they also include statistical tests which are most widely used in scientific data analysis. Thanks to these functions, users better understand the outcomes of the analysis and can communicate the results to other scientists.

Table 2 contains a listing of the functions supported by GeCoAgent, grouped by the phase in which they are used.

Table 2. List of functions divided by phase of the analysis process.

Extraction	Exploration	Analysis	Visualization
GMQL-like functions	**Summarization functions**	**Domain Indep. functions**	heatmap
select samples	min	classification	linechart
select regions	max	clustering	histogram
project metadata	median	regression	PCA diagram
project regions	avg	reduce data dimensionality	piechart
cover	find distribution	complex networks	violin plot
union	density	hypothesis testing	clustermap
difference	summary	parameter tuning	Kaplan-Meier curve
map	**Visualization functions**	**Domain Spec. functions**	enrichment analysis visualization
join	heatmap	mutation analysis	
merge	linechart	motif discovery	
Pivot functions	histogram	enrichment analysis	
pivot	PCA diagram	**Validation**	
label samples	piechart	cross-validation	
join pivot	mutation enrichment visualization	AUC, accuracy, precision, recall, F1	
flat		AIC	
		BIC	

From the syntactic point of view, functions are *unary* or *binary*, according the number of data input they accept. Functions' behavior depends on zero, one or many *input parameters*, that can be optional or required. Each function produces a *data output*, that can be taken as input by the next function. In addition, they have possible *side-effects*, such as displaying a graphic, logging the analysis, or creating a portion of a Jupyter notebook that encodes the analysis.

Functions are graphically represented as nodes of process graph; a schematic representation of a generic function is reported in Figure 3a.

A particular class of functions is 'higher-order functions', intensively used within the data analysis phase. Higher-order functions require one (or more) function as parameters; graphically, they are represented using a concentric structure in which higher-order functions encapsulate the parameter functions. Figure 3b illustrates one example of such structure, in which 'Parameter Tuning' is a higher-order function, taking the 'Silhouette' function as a parameter; in turn, also 'Silhouette' is a higher-order function that takes 'K-means' as a parameter.

The formulation in functions presents two main advantages. First, we are providing a set of primitives that are able to efficiently describe all the tasks. For example, task E2, i.e., 'Merge all regions from multiple samples of a dataset', is performed through two GMQL functions, 'Select samples' and 'Merge'. On the other hand, the modular definition of functions allows their arbitrary composition, even if not prescribed by a defined task. In this way, tasks are not the only mean of interaction for the user, but rather a sort of recommendation. Users can follow these suggestions, or instead use functions freely.

5.2 Task-driven Workflow

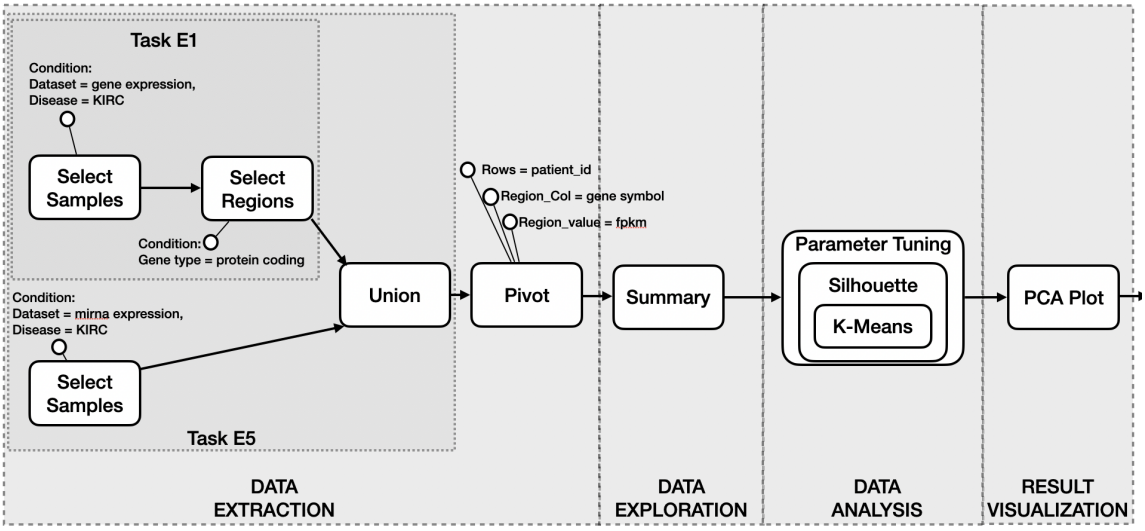


Fig. 4. An example of application using functions. The user selects expression data from a cohort of patients affected by KIRC, i.e. kidney renal clear cell carcinoma. Then she combine the samples and pivots the dataset and gets a table whose rows represent patients, columns genes or microRNAs and values contains fpkm expression data. The user plots the data using PCA and then applies k-means along with a parameter tuning procedure to identify the best number of clusters. Results are plotted, again using a PCA.

A genomic application is a process that includes functions as nodes, executed in an order dictated by precedence relationships. A common practice in computer science is to represent processes as directed graphs where the nodes are the functions along with their parameters, and the edges represent the flow of the execution. Figure 4 shows a simple application consisting of clustering the expression data of a particular disease, i.e., kidney renal clear cell carcinoma (KIRC); a *pivot* operation transforms the GDM dataset into a conventional table. In the first phase, the data extraction, we can recognize two tasks as patterns of functions: "Task E1", i.e., extract relevant regions from a dataset, composed by the

two functions "Select samples" and "Select regions", and "Task E5", i.e., combine samples from two datasets, composed of two "Select samples", one "Select regions" and a "Union". In the next phase, data exploration, a summary of retrieved data is displayed. Then, within data analysis, the user clusters the data using the *k-means* algorithm; the number of clusters is automatically selected to optimize the silhouette score. Finally, the data visualization phase concludes the process; the user manually evaluates the results by plotting the Principal Component Analysis of the obtained clusters. In platforms such as Galaxy[4], Orange[5] or Taverna[6], the workflows in Fig. 4 could be built by a bioinformatician who knows which functions are available, how to provide their input parameters, and how to interconnect them by means of suitable workflow edges. In our context, however, we target a biologist or clinician who is aware of needs and intents but unaware of the corresponding tasks and functions; a conversational agent must capture such needs and intents through a task-oriented dialogue, and then produce as result a workflow as the one of Fig. 4. Therefore we need to start from a different workflow representation, having user's interactions as main focus.

5.3 Conversation-Driven Workflow

Traditional conversational interfaces are also modelled on workflow diagrams, which are however modeling the status of a conversation in terms of what the conversational agent understands of the user's needs and intents; at each moment, during dialogue, the agent is in a given state of the automaton and as a response to external inputs (i.e., messages from the user) may change its state.

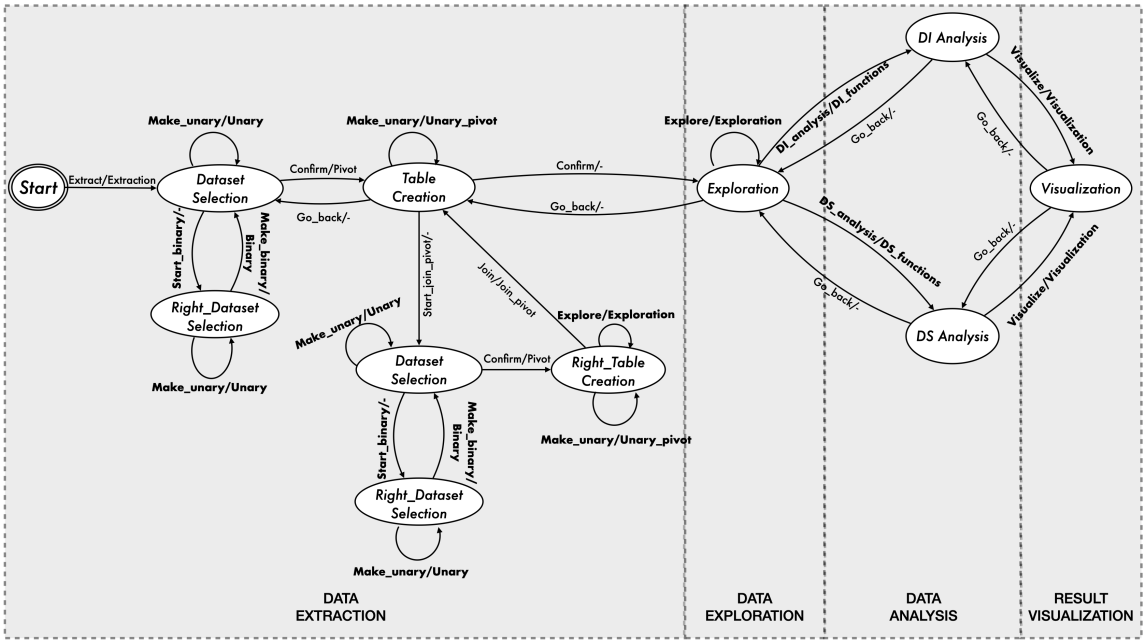


Fig. 5. Macro automaton that represent the generative process of any workflow considered in our framework. Every transition represents a function, every state corresponds to data created by function executions.

[4]https://usegalaxy.org
[5]https://orange.biolab.si
[6]https://taverna.incubator.apache.org

We define a high-level finite state automaton that can enact any process starting from the elementary functions, shown in Figure 5. In the state machine, every transition represents a function, whereas the states correspond to data created by function executions. All the transitions that begin in the same state must accept data with the same data type, all the transitions that end in the same state must return data with the same data type.

For example, the workflow defined in Fig. 4 is generated by following the path on Fig. 5 that operates on the `gene expression` dataset by selecting its samples and regions, then adds as right dataset `mirna expression` by selecting its samples, then performs the binary `union` operation. The confirmation allows to create the table using a `pivot`, that is next explored using the `summary`. The analysis part starts with a domain independent clustering analysis, performed by the `K-Means` method and `Siluhette` validation, which is repeated multiple times so as to tune parameters. At the end the visualization allows to display a PCA diagram. Thanks to the generality of the automaton, in addition to following pre-defined tasks, users can deviate at an any time and customize their workflow according to their needs.

We regard this automaton as the *macro automaton*, since it is able to map the conversation at the level of functions. However, this automaton is not sufficient to describe the interaction at the dialogic level. In fact, functions must be selected and invoked, but in addition some parameters must be set in order to obtain the desired result; these parameters often must be chosen through an iterative process, that requires more than a single interaction with the user. For this reason, for every function we define a *micro automaton*, in charge of ruling the conversation during the execution of the specific function.

Figure 6 is an example of micro automaton, namely the one in charge of executing data selection function. The micro automaton is invoked as effect of the first edge, labeled 'Extract/Extraction', in the macro automaton. At the beginning the user indicates if she would like to retrieve annotations or experimental data. In the first case, she will be asked for three parameters, namely *source*, *content type* and *assembly*. In the second one, more parameters can be defined, such as the *disease* or the *tissue* of the samples. Special treatment is given to the possibility of searching by *health condition*, an important aspect of many datasets used in human genomics. At the end, if the user confirms the choices, the selected samples are extracted.

5.4 From Automata to Conversations

Globally, the conversational agent works as follows: it starts the execution on the macro automaton, asking to the user to indicate the target of the first operation. When the user has chosen, GeCoAgent instantiates the corresponding micro automaton and its execution begins; the dialogue aims at understanding the user needs and intentions, either explicitly defined or inferred by means of answers to context-specific queries. The generative process of the conversation is guided by the automata: from every given state, the agent proactively emits a message for the user, it interprets the response of the user, infers her intent, and follows the edge marked with the same intent. At *micro*-level, the mechanism of questioning and interpreting the user's answers is used to fill all the required parameters and dependencies required by a given function, invoke it, and update the context. Possibly, some of the required parameters for the function are taken directly from the context, thus skipping portions of the conversation and making the agent less verbose.

By using state diagrams, the agent knows the exact state of the system, and it is able to anticipate the possible choices. Thus, GeCoAgent can proactively suggest the most adequate functions to the inexperienced users by looking at the tasks that are compatible with the trace followed during the interaction. For example, after having selected two datasets, GeCoAgent knows that the user must perform a binary operation between them, and can ask questions to understand the specific operation that the user needs. In summary, a conversation is captured by GeCoAgent thanks to the following components:

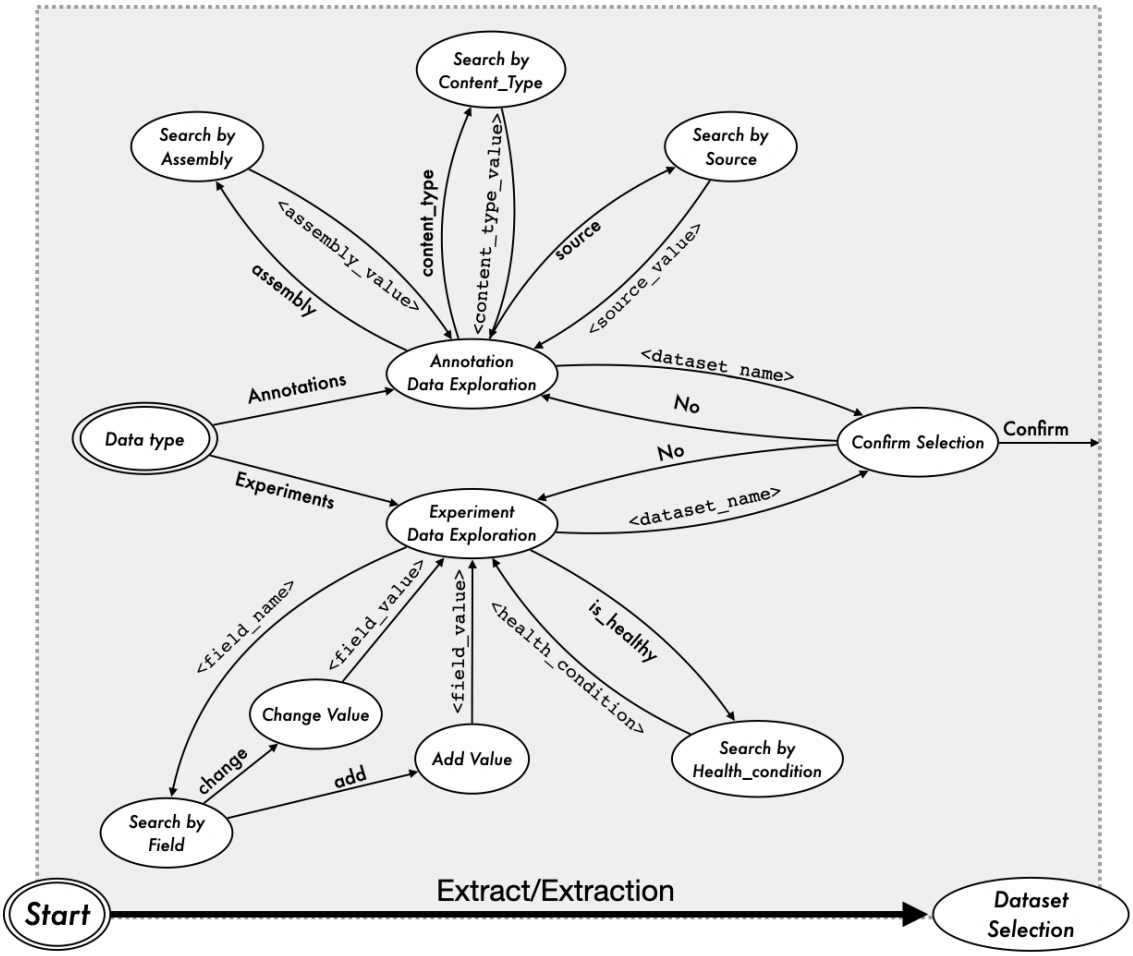


Fig. 6. An example of micro automaton for dataset exploration, invoked as effect of the first edge, labeled 'Extract/Extraction', in the macro automaton. Thanks to the dialog, user's intentions are captured and function's parameters are captured; in parallel, information is collected in the context object. When the micro automaton executed is completed, execution of the macro automaton resumes.

- a *macro* automaton, represented in Fig. 5 that captures the processing of functions; every edge is marked with a pair of labels "intent/micro", where the former label denotes an intent captured by reading the last user's message and the latter label, that can be missing, identifies a micro-automaton. For simplicity, similar labels and micros are grouped; they are represented using **bold characters**. E.g., **Make_unary** identifies the generic intent of performing a unary operation on the dataset, such as selection, filtering, or grouping, while **Unary** groups the generic micro automata associated to the unary operations. Also, all the trivial edges are omitted, such as help requests;
- a set of *micro* automata, each of those modeling in detail the conversation for a particular function; the role of a micro automaton is to extrapolate from the conversation the required inputs, outputs, parameters and connections of the function.

- finally, a companion *context object*, which contains information about the choices made by the user during the conversation, e.g., the set of values already provided for some parameters, that can be reused without generating portions of conversation for them.

Predefined *tasks* in Table 1 correspond to the most used patterns of interaction; they allow to detect the intention of the user early and to focus the dialogue. A pattern can be recognized by means of simple questions (e.g. during data extraction, after the user has selected mutations, the bot can ask if she is interested in exploring how they distribute to known genes; or during analysis, the bot can ask if the data analysis is concerned with classification or clustering); once the pattern is understood, dialogue generation is greatly helped, and best default options can be suggested by GeCoAgent (e.g. about the clustering, validation, and parameter tuning methods).

The conversational engine can exploit the knowledge gathered through the task definition, the history of the interaction, and the past executions to recommend the user the optimal choices. In this way, the conversational agent is not only an executor of operation, but a process facilitator, that can actively support the user through the whole process.

6 DEPLOYMENT

The architecture of GeCoAgent adopts a classic three-layer organization; it leverages on the use of state-of-the-art NLP and Conversational technologies to understand the user's intent from a written dialogue and on a highly expressive multimodal Web interface.

6.1 Architecture

Nowadays, several open source tools for the automated creation of conversational agents from a high-level definition of the anatomy of the dialogue are available; among the most popular, Chatterbot, Botpress, and Rasa. These tools are designed for simple applications (e.g., reserve a restaurant table or managing the FAQ of a bank), where the dialogue is much more predictable. Furthermore, these tools usually restrict their functionalities to the understanding of the user inquire and the prediction of the best response, but do not support the complex background operations that are needed for building and executing a complex data science workflow. Therefore, we implemented an independent software system whose architecture is depicted in Figure 7.

- The *frontend* allows the user to interact with GeCoAgent in a friendly way. It consists of a single-page web application, with several functional modules; it has been implemented using Vue.js framework to guarantee modularity and extendability, which is further detailed in Section 6.3. The module manager orchestrates the connection with the backend, and dispatches the information sent by the server. A set of independent modules elaborates the information to then show it on the Graphical User Interface.
- The *backend* comprises several components, the most relevant of which are: the Natural Language Understanding (NLU) unit, the dialogue manager (which includes the support for the automatons, the context state, the summarizing and the Jupyter notebook production modules), a relational data engine used for providing fast access to extracted datasets, and the executable implementation of the aforementioned atomic functions of which any workflow is composed.
- The *data layer* is composed of two interconnected components: GenoSurf and GMQL, the former for a curated retrieval of integrated data and the latter for the genome-aware manipulation of the extracted dataset. Both modules expose REST APIs, which are invoked by the backend for executing the data extraction *functions*.

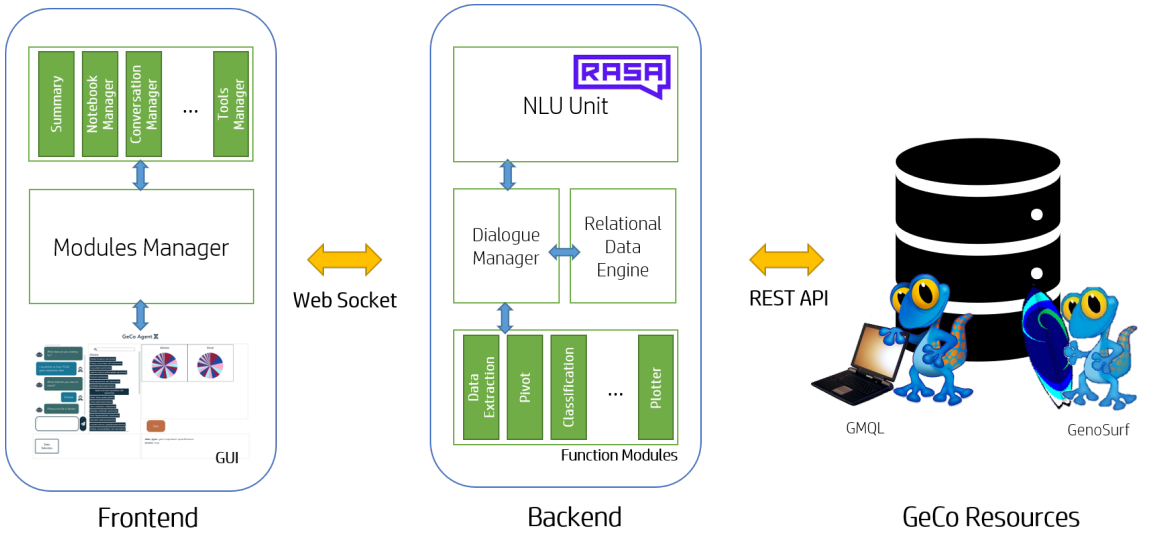


Fig. 7. The three-tier architecture of GeCoAgent. The frontend is composed of a modules manager that connects with the server and dispatch the information to the functioning modules, and updates the GUI accordingly. The backend includes the NLU Unit (implemented using RASA), the dialogue manager, and the function modules implementation; a relational engine supports the fast execution of data exploration and analysis primitives upon exported data. The data layer includes GeCo Resources.

Furthermore, GenoSurf APIs are intensively invoked by the dialogue manager module of the backend in order to build the response sentences and provide the user with the list of available field values.

The backend is implemented in Python using the Flask framework to serve the frontend and manage users' sessions. The data extraction functions are implemented by invoking the REST APIs provided by GenoSurf and the GMQL engine and standard Python methods for the manipulation of the tabular data obtained after a *Pivot*. The inspection, analysis, and visualization function are fully implemented in Python, leveraging the large availability of libraries for data analysis and visualization; underlying, a relational data engine equipped with suitable materialized views and indexes supports fast execution over extracted data. The communication between the frontend and the backend occurs by means of Web Socket (using the socket.io package). This enables fast, real-time, and bidirectional communication; the communication is often instantiated by the backend, which pushes pieces of information to the frontend.

6.2 Natural Language Understanding

Many chatbot frameworks allow to build natural language processing software, including Google DialogFlow, IBM Watson, and Rasa. We opted for Rasa NLU [20], an open-source library for understanding natural language, as it supports a comparatively larger number of intents. In particular, Rasa uses machine learning algorithms for understanding the meaning of texts, thereby being preferred in building custom NLP for complex chatbots [45].

6.2.1 Intent Classification and Entity Extraction. Intents denote the user's motivation; intent classification produces a categorization of user's sentences. Entities represent the target information that has to be understood and extracted by the chatbot. In order to classify intents and extract entities, Rasa predicts a set of slot-labels and slot-values associated with different segments of the input [46]. To do so, it processes incoming messages by means of a combination of

classifiers and extractors. These components are executed one after another in a so-called processing pipeline, which includes components for preprocessing, entity extraction, intent classification, and other purposes. Each component analyzes its input and creates its output, used by subsequent components in the pipeline [47].

Specifically, we use *SpacyNLP* as word vector source for loading pre-trained models, *SpacyTokenizer* for splitting a text into tokens, *SpacyFeaturizer* and *LexicalSyntacticFeaturizer* to transform the input messages into vector representations. Then, our pipeline includes:

- *DIETClassifier* for classifying intents and extract entities from users' messages. A dual Intent Entity Transformer (DIET) classifier is based on a transformer and a conditional random field (CRF) tagging layer.
- *CRFEntityExtractor* for refining entity extraction, based on a conditional random field (CRF) tagging layer.
- *EntitySynonymMapper* for mapping synonymous entity values to the extracted entity name. When an entity is extracted, this method checks if it should be changed with a synonym.

These three methods allow to better detect the entities mentioned in the messages.

In order to parse the messages, Rasa makes use of a model trained with sentences that are grouped according to the intent; the phrases used to train the model can contain one or more highlighted entities. The more sentences are provided for each intent, the better the model is trained, as the context is better defined and the concepts are extracted with higher accuracy. In our system, the expressions used to train the model are based on biological and clinical applications of human genomics; since there are many different types of genomic data, phrases map the bioclinical terms and their synonyms.

By using templates generated as a result of the user-driven requirement analysis explained in Section 4.2, we generate many different sentences that contain biomedical entities and their synonyms, e.g. a disease can be found in the training file with the complete name or with the abbreviation (kidney renal clear cell carcinoma or kidney carcinoma). We also use a file for each entity that contains all the synonyms found during the interviews in Section 4.2. These synonyms are then mapped through the correct entity thanks to the EntitySynonymMapper component.

6.2.2 Dialogue Management. A conversational agent should not only understand the intent of the user and the involved entities, but it should also build a context that allows directing the dialogue. In a chatbot system, the Dialogue Management part monitors the current system, understands the conversation state, and produces a dialogue. Dialogue management systems can be categorized as finite-state-based, frame-based, or agent-based [14]. In the first class, the dialogue flow is defined by a sequence of states, in the second class, the user is driven through a dialogue that also depends upon additional information available to the system [14], in the third case, the dialogue evolves dynamically as both the user and the bot are considered capable of reasoning [14]. Moreover, the dialogue management module can be user-directed, system-directed, or mixed according to the agent that drives the discourse [14].

Our dialogue management module can be considered not only finite-state but also frame-based, as the bot builds the dialogue according to both the current state of the automaton and to context information. GeCoAgent drives the dialogue, as it starts the conversation and asks questions, presenting the user with choices that allow producing the next state. The user can ask for help or provide the details requested by the bot. In addition to that, we handle errors and allow the user to change her choices. Changes in choices may change the dialogue directions, generating a new workflow that can be different from the previous one. At the end of the session, the user can see a summary of the more important messages of the conversation. The bot also provides a Python script that contains the operations that were executed for extracting and analyzing genomic data.

6.3 Multimodal Interface Design

The interview sessions described in Section 4.2 clearly indicated that a conversational interface alone was not enough to support such interaction. In fact, researchers must interact with many different kinds of information at all times during their work, including raw data, graphs, tables, documentation, and the history of previous operations. A conversation is able to convey only one information at a time. For these reasons, we decided to implement a multimodal interface, where the conversation is immersed within a traditional GUI. In this way, GeCoAgent leverages on the potentialities of both a conversational agent, that provides continuous support and allows the researcher to work using the natural language, and a GUI, that provides a channel where heterogeneous information can be shown in parallel.

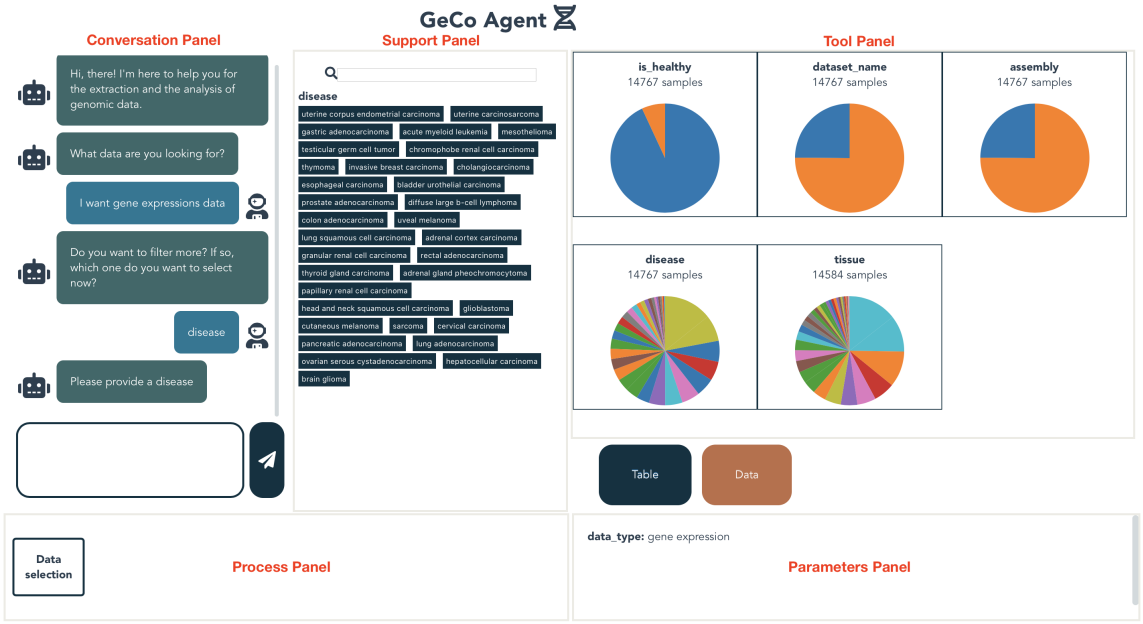


Fig. 8. A screenshot of the interface during the dialogue of an user with GeCoAgent. The user wants gene expression data and she wants to filter the disease. The interface shows the possible diseases and the available samples in the pie-charts. GeCoAgent interface is divided in 5 functional areas. The first row includes the *Conversation*, *Support*, and *Tool* panels; the second row includes the *Process* and the *Parameter* panels.

GeCoAgent interface, shown in Fig. 8, is divided in five functional areas, each one designated to a specific role in the interaction, positioned in two rows:

- The upper row includes the *Conversation Panel*, the *Support Panel* and the *Tool Panel*; they describe the tools actively used during the interaction.
- The second row includes the *Process Panel* and the *Parameters Panel*; they provide information for the user orientation.

We next describe each of them.

Conversation Panel. It includes the interaction between the user and the Conversational Agent; the whole conversation history is preserved within the panel, such that they browse the panel and recall what has been done before.

Support Panel. It has been designed to contain any information that can support the researcher during the conversation. The design reflects a trade-off between two requirements: on the one hand, the conversation must be kept as clean as possible, such that it contains only the information relevant to the current work; on the other hand, the researchers need support information that guides them on what can be done and what cannot. All this information has been moved from the Conversation Panel to this one, exclusively dedicated to this aim. In the Support Panel, the researcher can find two kinds of information:

- The available choices at the current state of the conversation, such as the possible operations at a certain point of the pipeline, or the available values to choose among to set a filter on the dataset;
- Tips and hints to suggest how to exploit GeCoAgent at its best.

When a set of available choices is shown, it is accompanied by a help icon. When the mouse hovers the icon, a tool-tip panel is shown explaining in detail what the user can do with the listed choices. For example, if researchers are filtering the data and have to choose according to which parameter the have to filter by, the tool-tip will explain that parameters can be added (in conjunction or disjunction with the already inserted ones), or removed. Given the cardinality of the choice set, that in some cases can be considerable (i.e. tens of possible elements), the support panel is provided of a search bar. In this way, users are not overwhelmed by the information that is present at the same time on the interface, but can explore it incrementally as they need. In summary, researchers can work exclusively in the chat, they can look at the choices shown by the Support Panel, or receive full support through the help tool-tip.

Tools Panel. It contains visual feedback for the user are shown. Since the visualization may be multiple, in the lower part of the panel, the users can select the one to see, which is shown in the upper canvas. Only one visualization can be shown at a time, but the user is free to switch between them. Visualizations are dynamically added and removed according to the flow of the conversation; When a new one is instantiated, the corresponding tag in the lower part is changed. GeCoAgent can toggle between the visualizations to suggest the user what to focus on. Anyway, the user can switch the tabs at any time. The following tools revealed to be necessary:

- *Data Visualization*, where graphs and charts are plotted. Visualizations are automatically generated by GeCoAgent, but the user can require additional ones.
- *Table Visualization*, where data tables are shown.
- *Lists Visualization*, where the users can explore lists of any nature, such as information related to the data, or sets of manually-curated datasets that are compatible with the operations the user is performing.

Process Panel. It provides a visual cue of how the process is evolving, by stating which step is being done, so that the user is oriented with respect to the whole process; it also contains a visual summary on the previous steps. The panel is composed of a series of blocks, each one representing an operation performed from the beginning of the execution. When the execution of a module is finished, the corresponding block changes colors, and a new one is added to the line. If the user hovers the mouse on one of the completed blocks, a tool-tip panel opens, describing all the choices done within the corresponding module.

Parameters Panel. It displays detailed information on the current operations; once the execution of the module is completed, the panel is cleaned, and all the same information is moved in the corresponding tool-tip in the Process Panel.

7 CONCRETE EXAMPLE

We analyze in detail a concrete example, showing how the design choices find concrete expression in a real problem. Let us suppose the user wishes to be trained in performing a data analysis on expression data. she would like her choice of the dataset to be driven by the number of samples available for specific tumors; to perform a significant analysis in statistical terms, a specific tumor with a homogeneous number of tumoral/healthy samples is preferred. After retrieving the data, she would like to transform it in a suitable format for further analysis (e.g., a comma-separated-values file is preferred). As she does not precisely know how to choose the appropriate data and how to prepare the analysis best, she relies on GeCoAgent; it guides her through the extraction process, by also providing information that characterizes the required data. Figure 9 shows an example dialogue between GeCoAgent and the user, capturing the interaction needed to achieve her objective of extracting the table of expression data.

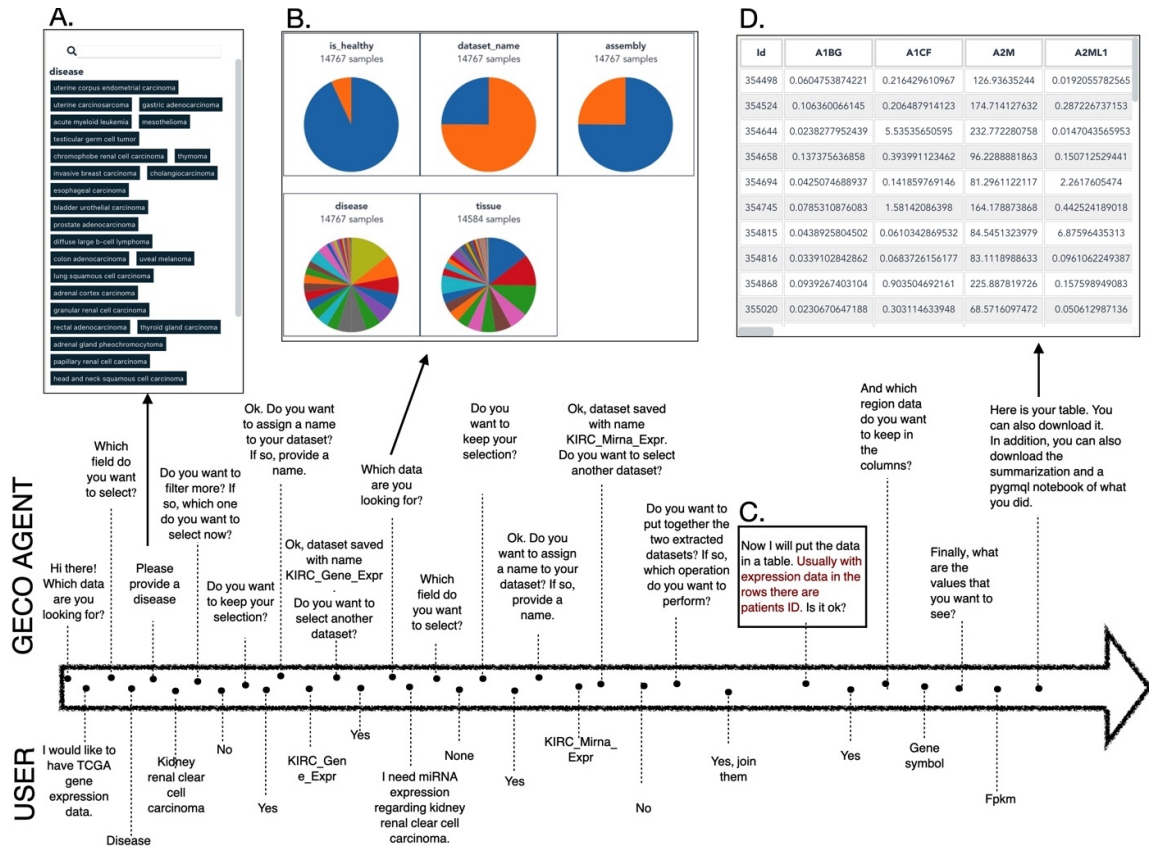


Fig. 9. A timeline of an actual GeCoAgent conversation for a genomic data extraction task. Throughout the conversation the user is aided in completing the task effectively; e.g., (A) the interface's *Support Panel* shows multiple options, the *Tool Panel* shows (B) the distribution of the metadata and (D) the selected dataset. Also, the Conversational Agent (C) guides the user by suggesting most common choices or best practices.

More in detail, the pipeline starts with GeCoAgent asking the user which kind of data should be selected. In the beginning, in case the user does not have a defined preference, the platform suggests to define at least the type

(i.e., annotations or experimental data); then it proceeds by suggesting the filters based on which the data could be additionally refined. In the example, the user requires data regarding gene expressions directly. Thus, GeCoAgent shows the chosen data type in the bottom right panel, while in the upper right corner it includes a set of pie-charts that represent the number of the available samples partitioned by the available fields (these, in turn, are shown in the center panel).

Figure 8 of the previous Section shows the point of the conversation in which the user is asked to add additional filters to her choice; first she asks to filter on 'disease', then, after having seen the number of samples in the pie-charts, she decides to extract 'gene expression' regarding 'kidney renal clear cell carcinoma'. She appears to be convinced with the current selection of data and proceeds with the conversation.

Following the instructions, she answers to the bot and since she does not change any previous choices, after providing a name for the prepared dataset, she can download it by clicking on the button in the bottom left panel (this appears after the confirmation of the dataset). The user may also, as it happens in this example case, decide to select another dataset. Indeed, since she wants a complete picture of the expression data for kidney cancer, she decides not only to have the expression data regarding the long protein coding gene, but also the microRNA expression data.

Also in this case, GeCoAgent allows to give a personalized name to the extracted dataset and to download it in one's local machine. The user would like to end the extraction/exploration session by transforming the selected data into a single table containing all the expression values. GeCoAgent suggests different functions to join the two datasets together (such as UNION and DIFFERENCE).

In this case, the user would like to use the two datasets together, thus decides to perform the UNION operation. To transform the data into a table GeCoAgent requires some parameters: a table can have different metadata on the rows, different metadata or region data in the columns and different values shown in the table. As indicated with red font in Figure 9, GeCoAgent provides a suggestion to the user, relying on standard practices (i.e., common choice of parameters for certain kind of data).

In conclusion, the user obtains a table with patient IDs in the rows, gene symbols in the columns and fragments per kilobase million (fpkm) as values. GeCoAgent provides a file that summarizes the process followed to obtain the data – highlighting the choices of the user and abstracting away the details of the conversation that are not relevant for the selection of filters and parameters.


```
{ "Summary": "You have chosen tcga gene expression quantification data with
    filter 'disease'='kidney renal clear cell carcinoma'. You renamed the first
     dataset 'KIRC_Gene_Expr'. You have chosen mirna expression quantification
    for kidney renal clear cell carcinoma. You renamed the second dataset '
    KIRC_Mirna_Expr'. You use UNION operation and you renamed the complete
    dataset 'KIRC_Expr'. The table provided has patients ID in the rows, no
    metadata in the columns and gene symbols in the columns and fpkm as values
    .",
  "Choices": [
    {
      "source": "tcga",
      "data_type": "gene expression quantification",
      "disease": "kidney renal clear cell carcinoma",
```

```
      "name": "KIRC_Gene_Expr"
    },
    {
      "data_type": "mirna expression quantification",
      "disease": "kidney renal clear cell carcinoma",
      "name": "KIRC_Mirna_Expr"
    },
    {
      "binary_operation": "union"
    },
    {
      "rows": "patient ID",
      "metadata columns": [],
      "region columns": ["gene symbol"],
      "region values": "fpkm"
    }
  ],
}
```

Additionally, it provides a Jupyter PyGMQL [48] notebook to reproduce the analysis when needed. In Figure 10 there is the notebook downloaded from this example.

8 USER STUDY

8.1 Goal and research variables

We performed a preliminary empirical study on GeCoAgent in order to assess its effectiveness in supporting biologists and bioinformaticians in tertiary analysis. As our goal was to investigate the factors that may either promote or prevent its adoption within each of the two main target groups,we focused our evaluation on *usability*, the main motivation for GeCoAgent. We collected two objective measures, concerned with the user's performance in executing tertiary analysis tasks: *task completion* and *task execution time*. We also considered subjective measures collected through semi-structured interviews, which were also used for interpreting the quantitative results and for exploring the *perceived potential for adoption*. We were also interested to get insights on the typical *user's behaviors* emerging from our observation of the users while using GeCoAgent; thanks to these observations, the study allowed us to elicit new requirements for our tool.

8.2 Participants

We recruited two groups of persons (hereinafter referred to as "users") for a total of 14 people. Group 1 was composed of 7 biologists who had some experience in working with genomic data but had limited know-how in computer science. Group 2 comprised 7 bioinformaticians, i.e., persons with advanced computer science skills, general competence on computational biology, but limited know-how on gene expression analysis. All users were aged between 25 and 30 (M=26 in both groups) and none of them had previous experience in GeCoAgent. Three members of GeCoAgent research team (hereinafter referred to as "researchers") participated as moderators and observers during the study.

All users participated on a voluntary basis and were recruited by email at external research or clinical institutes collaborating with us in previous projects . Before the study they signed a consent form which explained the purpose of the study and its procedure, and included all regulatory rules we followed to guarantee the anonymity and privacy of the data collected.

8.3 Procedure

Due to the current pandemic outbreak, the study was performed online using a video conference software to enable the communication between users and researchers and the researcher's observation of user behavior during the use of GeCoAgent. Each user attended one session, which comprised three steps.

Step 1. Presentation of the tertiary analysis activity to be performed. The assigned activity involved common tasks in tertiary analysis such as data retrieval, conversion of retrieved data into a tabular format, and data clustering. The users were required to cluster data about patients affected by pancreatic adenocarcinoma according to their gene expressions. The activity was decomposed in 3 specific tasks:

GeCoAgent Notebook

Logging into GMQL

```
import gmql as gl

gl.set_remote_address("http://gmql.eu/gmql-rest/")
gl.login()
gl.set_mode("remote")
```

Select first dataset

```
genes = gl.load_from_remote("GRCh38_TCGA_gene_expression", owner="public")
KIRC_Gene_Expr = genes.select(meta_predicate=genes["disease_code"] == 'KIRC')
```

Select second dataset

```
mirnas = gl.load_from_remote("GRCh38_TCGA_mirna_expression", owner="public")
KIRC_Mirna_Expr = mirnas.select(meta_predicate=mirnas["disease_code"] == 'KIRC')
```

Union

```
KIRC_Expr = KIRC_Gene_Expr.union(other=KIRC_Mirna_Expr)
```

Pivot

```
KIRC_Expr = KIRC_Expr.materialize("./KIRC_Expr", all_load=False)
```

```
KIRC_Expr = KIRC_Expr.to_matrix(index_regs=['gene_symbol'],
                                columns_meta=["biospecimen__bio__bcr_analyte_barcode"],
                                values_regs=["reads_per_million_mirna_mapped"],
                                fill_value=0)
KIRC_Expr = KIRC_Expr.T
```

Fig. 10. PyGMQL notebook generated from the conversation with the user.

T1: find the gene expression data for pancreatic adenocarcinoma; data extraction using GeCoAgent was from GenoSurf [4] (introduced in Section 3.0.1), unconstrained users extracted the datasets either from GenoSurf or directly from The Cancer Genome Atlas (TCGA), which is the standard repository for cancer-related gene expression dataset. The two sources contain the same datasets, as GenoSurf imports TCGA data.

T2: create a table to operate on samples, thus with the patients (i.e. samples) on the rows, the gene symbols on the columns and the expression values (in fpkm) in the cells without adding any labels;

T3: identify the clusters of the patients using k-means clustering, using automatic parameter tuning to find the optimal number of clusters (in the range [2,5]).

The maximum time for the execution of a single task, called *task time limit*, was 15 minutes.

Step 2. Execution of the assigned tasks. We were interested to explore our research variables for each of the main user profiles - biologists and bioinformaticians. For the latter, we also wanted to compare GeCoAgent against generic tools and/or programming languages (such as PyTthon, R, or similar) that are commonly adopted in bioinformatics research. We therefore differentiated the execution of tasks between the two groups of participants. *Group 1 (biologists)* performed the assigned tasks in GeCoAgent only. *Group 2 (bioinformaticians)* performed the assigned tasks in two experimental conditions: *"using GeCoAgent"* and *"using other tools"*, i.e., systems and/or programming languages they were familiar with in their regular tertiary analysis work. The order of the two experimental conditions was randomized among participants to counterbalance potential learning effects.

Participants were asked to "think aloud" [49] during task execution, i.e., to say whatever was coming into their minds as they were approaching each task, including what they were looking at, thinking, doing, and feeling. Users shared their screens with the remote researchers and activated both the camera and the microphone, to enable remote researchers observe and listen what users were doing and saying. Figure 8 of Section 6 is a picture of the user's screen during an interview. The screenshot shows the point of the conversation in which GeCoAgent asks to add additional filters and the user wants to select the disease: it highlights the interaction of the user that at the beginning uses natural language sentences while in the second message exploits the available keywords.

Researchers measured the execution time of the tasks and took notes on task completion and how users interacted with the system, especially noting places where they encounter difficulty. Test sessions were video-recorded so that during the analysis stage of the study we could go back and refer to what participants did and how they reacted.

Step 3. Final interview. At the end of each session, the researcher administered a short semi-structured interviews focusing on the perceived usability and utility of the tool(s) used to execute the tasks, and the potential for adoption. Interviews were videorecorded. The questions were:

- How much do you find GeCoAgent ease to use, in a scale from 1 (very difficult) to 5 (very easy)? What are the motivations for your score?
- How much do you find GeCoAgent useful, in a scale from 1 (useless) to 5 (very useful)? What are the motivations for your score?
- Would you recommend GeCoAgent to biologists (yes/no) and for which reasons? Would you recommend GeCoAgent to bionformaticians (yes/no) and for which reasons?
- What are the main advantages of GeCoAgent?
- What are the main disadvantages of GeCoAgent?

8.4 Main Results

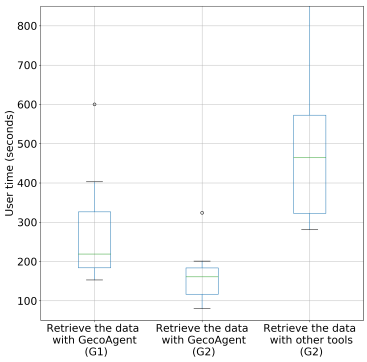


Fig. 11. Boxplots of the time (in seconds) required for the data retrieval task T1. The first boxplot is relative to biologists (G1) using GeCoAgent, the second one to bioinformaticians (G2) using GeCoAgent, the third one to bioinformaticians (G2) using "other tools".

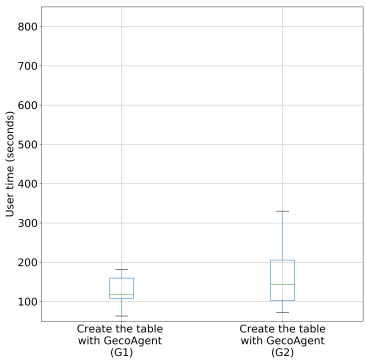


Fig. 12. Boxplots of the time (in seconds) required for the table creation task T2 using GeCoAgent. The first boxplot is relative to biologists (G1), the second one to bioinformaticians (G2).

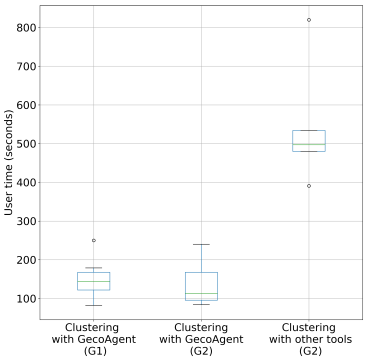


Fig. 13. Boxplots of the time (in seconds) required for the clustering task T3. In the first boxplot is relative to biologists (G1) using GeCoAgent, the second to bioinformaticians (G2) using GeCoAgent, the third one to bioinformaticians (G2) using "other tools".

Analysis of Quantitative Data. We measured *task completion* on a binary scale (1=task completed within the time limit; 0= task not completed by the time limit). All 14 users completed all tasks using GeCoAgent. Still when considering other tools and/or programming languages, the results on task completion for the users in Group 2 (bioinformaticians) were less positive. No user in G2 completed task T2 in this experimental condition. Three users explicitly declared they were unable to perform it and gave up very soon; four users tried hard but could not complete the task within the time limit. Similar behaviours occurred for T1 and T3: One user for task T1 and one for T3 gave up; One user for T3 completed the task only partially.

For the *time on task* variable, we considered only the time for the tasks completed within the time limit. For those performed in GeCoAgent (which were completed by all users within the time limit) there was is no significant difference for the two user groups, as shown by the Wilcoxon signed-rank two-paired test between the two distributions, $p > 0.1$ (T1: $p = 0.219$; T2: $p = 0.579$,; T3: $p = 0.579$). For Group 2, we compared the time on task in the two experimental conditions - "using GeCoAgent" and "using other tools" - on only tasks T1 and T3. The outcome showed that in both tasks, the time on task on GeCoAgent was significantly *lower* than "using other tools". On average, with GeCoAgent, T1 was 3.0 times faster (Wilcoxon test, $p = 0.0313$) and T3 was 3.9 times faster (Wilcoxon test, $p = 0.0156$).

The quantitative data from the final interview show that GeCoAgent's *perceived usability* was good. The average usability score on all users was 4.00/5 (SD=0.78), with a lower average rate for biologists of 3.86/5 (SD=0.90) and a higher average rate for computer scientists of 4.14 (SD=0.69). Despite the above data indicate that some biologists encountered difficulties while using GeCoAgent, this did not affect perceived utility, which was even higher than usability: the average utility score on all users was 4.57/5 (SD=0.65); and the value was the same for both groups, with small differences only on the standard deviation (SD=0.79 v.s. SD=0.53).

Concerning *potential for adoption*, all participants recommended the use of GeCoAgent for biologists. For bioinformaticians, the recommendation of the tool was not unanimous: twelve participants (5 biologists and 7 bioinformaticians)

recommended it and two participants (biologists) did not recommend it. One biologist and two bioinformaticians recommended GeCoAgent only for the preliminary phases of Tertiary Analysis (e.g., Data Retrieval and Exploration).

Analysis of Qualitative Data. We used thematic analysis [50] to elicit recurrent topics emerging from the transcripts of interviews and thinking aloud recordings. In the rest of this section, capital letter B denotes biologists and C denotes bioinformaticians. 7 participants (4 biologists, 3 bioinformaticians) mentioned that, even if the system was easy to use, there is an initial learning phase that however is short, lasting only for the first few interactions; after them, they acquired confidence with the system, "*the platform becomes easy*" [B3]

Among the advantages mentioned by the users, *intuitiveness* (4 participants), *ease of use* (7 participants) and *fun* [B7] were the most recurring terms. *Efficiency* was also mentioned among the benefits of GeCoAgent: B2 and C6 said that operations in GeCoAgent are "*faster*"; B2 also noted that "*GeCoAgent saves time by avoiding coding errors*", and C3 appreciated "*automatically generated plots*".

Nine participants *liked* the GeCoAgent chat-based interface. The chatbot questions were defined "*simple*" [C5] and "*easy to understand*" [B1]. C2 enjoyed how the chatbot asked for "*confirmation between one step and the following one*" while C3 found the chatbot proactivity helpful: "*I like how the bot anticipates the steps and suggests them to you*" . 4 participants found the visual interface well organized and functional, as "*all the elements are well organized*" and "*on the same interface*", generally providing "*a clear functional organization of all the panels*". Users appreciated the smooth interplay between the two interaction modalities (the graphical interface and the conversational one); in particular, they liked how the chatbot suggested where to find information on the interface [C3], the display of information synchronous with the chat [B4], and the keywords in the support panel that suggest what to do next [B4, C1, C6]. Finally, several users in Group 2 also highlighted the potential of GeCoAgent not only as an operational tool to support demanding tasks, but also as an *educational* tool for novice biologists or bioinformaticians, to learn about key reasoning steps and operations to perform during tertiary analysis.

When asked for *disadvantages*, 4 computer scientists highlighted the limited transparency of the choices performed by the tool, particularly the lack of explanations of the parameters automatically used by the chatbot. Some users claimed that in some moments it was not easy to understand the alternatives for the next step, or feedback was too concise in explaining what to do - they would have preferred longer help message. They also claimed that, in order to use GeCoAgent systematically, they would need "*more control on the available operations*" [C3]. Biologists found GeCoAgent functionalities powerful yet limited sometimes: they would use the tool more intensively if the system included a wider number of analytical operations needed in their everyday work [B1, B5].

Finally, the analysis of video recordings, researcher's notes taken during the sessions, and conversation logs indicated that the comprehension capability of the chatbot was satisfactory. The conversational agent was unable to "go-back" to a specific step required by the user (a functionality not yet implemented), but it misinterpreted the user only in two cases. The analysis of conversation flows also highlighted two recurrent patterns of users' conversational behavior: talking with the chatbot "concisely", i.e., by using only the keywords presented in the graphic interface panel, and "extensively", i.e., formulating complete sentences.

Discussion. The evaluation of GeCoAgent on a simple but realistic use case suggests that this tool has a good degree of usability, perceived utility, and perceived potential for adoption. Regardless the different user profiles, the initial learning curve was short, and participants considered the system from easy to very easy to use. All of them were able to complete all assigned tasks in GeCoAgent even if they were not familiar with the platform nor had received any preliminary training. We hypothesise that our design approach, informed by a careful elicitation and modeling of the

cognitive and operational processes of tertiary analysis, could have contributed to make the use of the system more intuitive and predictable. Overall, the study result suggest that GeCoAgent could empower biologists with limited computer science skills, enabling them to perform tertiary analysis tasks more effectively and more autonomously, reducing the need of support from by bioinformaticians. Yet, also bioinformaticians could benefit from the use of GeCoAgent. The comparison of the time taken to perform tasks T1 and T2 in GeCoAgent and by programming the required operations from scratch indicates that our system enables also bioinformaticians to perform some complex operations more efficiently, also relieving them from the burden of learning new programming skills (which some participants missed, preventing them to complete task T2).

GeCoAgent was initially conceived mainly as a tool for expert professionals in genomic analysis. An interesting insight emerged from the study that broadens the spectrum of our target users and the context of use: according to the participants, the tool holds a potential also for the novice; furthermore, it could be used for educational purposes, to train students in biology and bioinformatics and to promote learning of tertiary analysis.

Concerning the design features of GeCoAgent, the multimodal interaction approach and the smooth integration of GUI (graphical user interface) and chatbot behaviors emerged as a point of strength of the system. The two paradigms were perceived as complementary and mutually enforcing each other. Still, some limitations were noticed in the operational flow supported by the system, via GUI end/or chatbot. In some cases, it was perceived as too rigid, i.e., lacking alternative paths to reflect the multiple ways to reach a solution or to reflect the contingent cognitive goal of the user. Concerning the interaction with the chatbot, some participants used compact, keywords-based expressions, and they also appreciated the concise style of the chatbot utterances. Other users found the style of the chatbot sentences too minimalistic. In addition, they would have needed more feedbacks on their interactions, and more articulated explanations of the actions to do or the operations performed by the system. Managing different conversation styles and contents to dynamically address the communication requirements of the different users in different moments of the interaction is a challenging goal, which we have in our future research agenda.

Overall, the results of the study are encouraging. They highlight the potential of integrating traditional (GUI) interfaces and conversational interaction to support complex activities in tertiary analysis, paving the ground towards further research in interactive technology for bioinformatics that might lead to new solutions for researchers in the field. Still, the findings of our empirical study should be taken with caution and so far can be hardly generalized, considering the preliminary and exploratory nature of the research, and its limitations. Particularly, the size of the user sample is small, although comparable or higher than previous empirical studies on conversational interaction in similar domains. For example, the empirical evaluations of Iris [40] and Ava [38] involved 8 and 10 participants respectively (using a within-subjects study design). In addition, we tested our system on a small set of tasks, which are common in tertiary analysis, but are not the only ones involved in this complex activity. The evaluation of GeCoAgent on a different and wider set of tasks might highlight weaknesses that the present study did not unveil.

9 CONCLUSIONS AND FUTURE DIRECTIONS

Thanks to the combination of a number of distinctive features, GeCoAgent stands out as a system which provides significant advantages to the state-of-the-art:

- The *concept of GeCoAgent* embeds expertise in genomic computing, both for what concerns the high level of abstractions of supported models and languages and the availability of solid systems, which provide effective data management, integrative pipelines, and repositories encompassing most relevant open data sources.

- The *requirements of GeCoAgent* stem from a combination of top-down experience (both domain-specific and with a broader data science vision) and bottom-up interview of biologists and clinicians. They allow us to structure the process of genomic computing into alternations of steps of data extraction and data analysis, each supported by exploration and visualization. Within this broad structure, we recognize recurrent patterns of interaction, that can be useful for the early recognition of user's intent and for improved structuring of the process.
- The *design of GeCoAgent* is driven by an original construction of automata for driving the conversational agent, by both capturing the need of enacting the classical data extraction and data analysis functions as well as capturing the user's intent. The automaton is also driven by the progressive construction of the process' state, which provides contexts to conversations.
- The *implementation of GeCoAgent* combines two important technologies for empowering conversational agents, namely NLP recognition using machine learning acting on a corpus of task-oriented dialogues, and a multimodal user interface which combines the chatbot with synchronized data visualizations and with a concise description of the evolving context of analysis.

The result of a design session is delivered not only in terms of extracted datasets and statistical results, each equipped with its own readers so as to be maximally user-friendly (e.g., Excel tables, diagrams and plots in standard visualization formats), but also in terms of a summarization text describing the whole process and of a Python script, embedded within a Jupiter notebook and linked to both internal and external data resources, which can be used to faithfully replicate the process. In this way, while a biologist or clinician is satisfied by the user-friendly delivery of results, he is also reassured that such results can be replicated (perhaps with the support of a bioinformatician) or rerun at later times and/or upon new datasets. In particular, replicability is stronger thanks to the use of Python scripts which can be locally executed and are quite independent from GeCo technology.

By bridging the cognitive gap which currently separated biologists and clinicians to the use of bioinformatics tools, GeCoAgent can also play a role for efficiently integrating genomics into the public healthcare research, facilitating the implementation of personalized medicine in the context of diagnostics, prevention and treatment. Some of the advantages of GeCoAgent stem also from the power of integrative pipelines for genomic data integration [3], that guarantee seamless integration and enrichment of genomic metadata with novel aspects of biological and clinical research.

While GeCoAgent is already working and testable, it misses several components which will be progressively added. The requirements, design and implementation of GeCoAgent are inspired by modularity (at all three levels), therefore the resulting system can seamlessly evolve. We plan to extend the current core by taking advantage of continuous cycles of requirement elicitation and validation, performed within GeCo collaborations and with the independent contributions of external groups. We will also consider the extension of GeCoAgent in order to deliver a domain-independent conversational agent for data science.

ACKNOWLEDGEMENTS

This work was supported by the ERC Advanced Grant 693174 "Data-Driven Genomic Computing (GeCo)".

REFERENCES

[1] Marco Masseroli, Pietro Pinoli, Francesco Venco, Abdulrahman Kaitoua, Vahid Jalili, Fernando Palluzzi, Heiko Muller, and Stefano Ceri. 2015. GenoMetric Query Language: a novel approach to large-scale genomic data management. *Bioinformatics* 31, 12 (2015), 1881–1888.

[2] Marco Masseroli, Arif Canakoglu, Pietro Pinoli, Abdulrahman Kaitoua, Andrea Gulino, Olha Horlova, Luca Nanni, Anna Bernasconi, Stefano Perna, Eirini Stamoulakatou, et al. 2019. Processing of big heterogeneous genomic datasets for tertiary analysis of Next Generation Sequencing data. *Bioinformatics* 35, 5 (2019), 729–736.

[3] A. Bernasconi, A. Canakoglu, M. Masseroli, and S. Ceri. 2020. META-BASE: a Novel Architecture for Large-Scale Genomic Metadata Integration. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2020), 1–1.

[4] Arif Canakoglu, Anna Bernasconi, Andrea Colombo, Marco Masseroli, and Stefano Ceri. 2019. GenoSurf: metadata driven semantic search system for integrated genomic datasets. *Database: The Journal of Biological Databases and Curation* 2019 (2019).

[5] Andreas D Baxevanis, Gary D Bader, and David S Wishart. 2020. *Bioinformatics*. John Wiley & Sons.

[6] R. Gabe. 2010. A hitchhiker's guide to Next Generation Sequencing - Part 2. https://blog.goldenhelix.com/a-hitchhikers-guide-to-next-generation-sequencing-part-2/. Last accessed May 1st, 2021.

[7] Anna Bernasconi, Arif Canakoglu, Marco Masseroli, and Stefano Ceri. 2021. The road towards data integration in human genomics: players, steps and interactions. *Briefings in Bioinformatics* 22, 1 (2021), 30–44. https://doi.org/10.1093/bib/bbaa080

[8] Stefano Ceri, Anna Bernasconi, Arif Canakoglu, Andrea Gulino, Abdulrahman Kaitoua, Marco Masseroli, Luca Nanni, and Pietro Pinoli. 2017. Overview of GeCo: A project for exploring and integrating signals from the genome. In *International Conference on Data Analytics and Management in Data Intensive Domains*. Springer, 46–57.

[9] Antony T Vincent and Steve J Charette. 2015. Who qualifies to be a bioinformatician? *Frontiers in genetics* 6 (2015), 164.

[10] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. 2013. Orange: Data Mining Toolbox in Python. *Journal of Machine Learning Research* 14 (2013), 2349–2353. http://jmlr.org/papers/v14/demsar13a.html

[11] Mary J Goldman, Brian Craft, Mim Hastie, Kristupas Repečka, Fran McDade, Akhil Kamath, Ayan Banerjee, Yunhai Luo, Dave Rogers, Angela N Brooks, et al. 2020. Visualizing and interpreting cancer genomics data via the Xena platform. *Nature Biotechnology* (2020), 1–4.

[12] Ravi K Madduri, Dinanath Sulakhe, Lukasz Lacinski, Bo Liu, Alex Rodriguez, Kyle Chard, Utpal J Dave, and Ian T Foster. 2014. Experiences building Globus Genomics: a next-generation sequencing analysis service using Galaxy, Globus, and Amazon Web Services. *Concurrency and Computation: Practice and Experience* 26, 13 (2014), 2266–2279.

[13] Davide Bolchini, Anthony Finkelstein, Vito Perrone, and Sylvia Nagl. 2009. Better bioinformatics through usability analysis. *Bioinformatics* 25, 3 (2009), 406–412.

[14] Liliana Laranjo, Adam G Dunn, Huong Ly Tong, Ahmet Baki Kocaballi, Jessica Chen, Rabia Bashir, Didi Surian, Blanca Gallego, Farah Magrabi, Annie YS Lau, et al. 2018. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association* 25, 9 (2018), 1248–1258.

[15] AM Turing. 1950. Mind. *Mind* 59, 236 (1950), 433–460.

[16] Joseph Weizenbaum. 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (1966), 36–45.

[17] Richard S Wallace. 2009. The anatomy of ALICE. In *Parsing the Turing Test*. Springer, 181–210.

[18] Kenneth Mark Colby. 1975. *Artificial paranoia: a computer simulation of paranoid process*. Pergamon Press.

[19] Richard Wallace. 2003. The elements of AIML style. *Alice AI Foundation* 139 (2003).

[20] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *ArXiv* abs/1712.05181 (2017).

[21] Marti Hearst and Melanie Tory. 2019. Would You Like A Chart With That? Incorporating Visualizations into Conversational Interfaces. In *2019 IEEE Visualization Conference (VIS)*. IEEE, 1–5.

[22] James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. 2007. Plow: A collaborative task learning agent. In *AAAI*, Vol. 7. 1514–1519.

[23] Petter Bae Brandtzaeg and Asbjørn Følstad. 2017. Why people use chatbots. In *International Conference on Internet Science*. Springer, 377–392.

[24] Esther Kaufmann and Abraham Bernstein. 2010. Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Journal of Web Semantics* 8, 4 (2010), 377–393.

[25] Asbjørn Følstad and Petter Bae Brandtzæg. 2017. Chatbots and the new world of HCI. *interactions* 24, 4 (2017), 38–42.

[26] Sharon Oviatt. 1999. Ten myths of multimodal interaction. *Commun. ACM* 42, 11 (1999), 74–81.

[27] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M Mitchell, and Brad A Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 577–589.

[28] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. 2015. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 489–500.

[29] Kedar Dhamdhere, Kevin S McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. Analyza: Exploring data with conversation. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. 493–504.

[30] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. 2017. Applying pragmatics principles for interaction with visual analytics. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 309–318.

[31] Melanie Tory and Vidya Setlur. 2019. Do what i mean, not what i say! design considerations for supporting intent and context in analytical conversation. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 93–103.

[32] Adam Blum. 1999. Microsoft english query 7.5: Automatic extraction of semantics from relational databases and OLAP cubes. In *VLDB*, Vol. 99. 247–248.

[33] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. 149–157.

[34] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R Mittal, and Fatma Özcan. 2016. ATHENA: an ontology-driven system for natural language querying over relational data stores. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1209–1220.

[35] Antonio Messina, Agnese Augello, Giovanni Pilato, and Riccardo Rizzo. 2017. BioGraphBot: A Conversational Assistant for Bioinformatics Graph Databases. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, 135–146.

[36] Antonino Fiannaca, Massimo La Rosa, Laura La Paglia, Antonio Messina, and Alfonso Urso. 2016. BioGraphDB: a new GraphDB collecting heterogeneous data for bioinformatics analysis. *Proceedings of BIOTECHNO* (2016).

[37] Walter Ritzel Paixão-Côrtes, Vanessa Stangherlin Machado Paixão-Côrtes, Cristiane Ellwanger, and Osmar Norberto de Souza. 2019. Development and usability evaluation of a prototype conversational interface for biological information retrieval via bioinformatics. In *International Conference on Human-Computer Interaction*. Springer, 575–593.

[38] Rogers Jeffrey Leo John, Navneet Potti, and Jignesh M Patel. 2017. Ava: From Data to Insights Through Conversations.. In *CIDR*.

[39] Norbert E Fuchs and Rolf Schwitter. 1995. Specifying logic programs in controlled natural language. *arXiv preprint cmp-lg/9507009* (1995).

[40] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. 2018. Iris: A conversational agent for complex tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[41] Daniel Vanderveken. 1990. *Meaning and speech acts: Volume 1, principles of language use*. Vol. 1. Cambridge University Press.

[42] Marco Masseroli, Abdulrahman Kaitoua, Pietro Pinoli, and Stefano Ceri. 2016. Modeling and interoperability of heterogeneous genomic big data for integrative processing and querying. *Methods* 111 (2016), 3–11.

[43] Anna Bernasconi, Stefano Ceri, Alessandro Campi, and Marco Masseroli. 2017. Conceptual Modeling for Genomics: Building an Integrated Repository of Open Data. In *Conceptual Modeling*, Heinrich C. Mayr, Giancarlo Guizzardi, Hui Ma, and Oscar Pastor (Eds.). Springer International Publishing, Cham, 325–339.

[44] Anna Bernasconi, Arif Canakoglu, and Stefano Ceri. 2019. From a Conceptual Model to a Knowledge Graph for Genomic Datasets. In *Conceptual Modeling*, Alberto H. F. Laender, Barbara Pernici, Ee-Peng Lim, and José Palazzo M. de Oliveira (Eds.). Springer International Publishing, Cham, 352–360.

[45] Sumit Raj. 2018. Building chatbots with Python. *Using Natural Language Processing and Machine Learning. Apress* (2018).

[46] Thierry Desot, Stefania Raimondo, Anastasia Mishakova, François Portet, and Michel Vacher. 2018. Towards a French Smart-Home Voice Command Corpus: Design and NLU Experiments. In *International Conference on Text, Speech, and Dialogue*. Springer, 509–517.

[47] Srimoyee Bhattacharyya, Soumi Ray, and Monalisa Dey. 2020. Context-Aware Conversational Agent for a Closed Domain Task. In *Proceedings of the Global AI Congress 2019*. Springer, 303–318.

[48] Luca Nanni, Pietro Pinoli, Arif Canakoglu, and Stefano Ceri. 2019. PyGMQL: scalable data extraction and analysis for heterogeneous genomic datasets. *BMC bioinformatics* 20, 1 (2019), 560.

[49] Ted Boren and Judith Ramey. 2000. Thinking aloud: Reconciling theory and practice. *IEEE transactions on professional communication* 43, 3 (2000), 261–278.

[50] Greg Guest, Kathleen M MacQueen, and Emily E Namey. 2011. *Applied thematic analysis*. sage publications.