



(51) International Patent Classification:

G10L 15/02 (2006.01) G06F 16/242 (2019.01)

G10L 15/18 (2013.01)

(21) International Application Number:

PCT/SG2021/050121

(22) International Filing Date:

09 March 2021 (09.03.2021)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

10202002286R 12 March 2020 (12.03.2020) SG

(71) Applicant: NATIONAL UNIVERSITY OF SINGAPORE [SG/SG]; 21 Lower Kent Ridge Road, Singapore 119077 (SG).

(72) Inventors: PITT, Jason James; c/o National University of Singapore, Cancer Science Institute of Singapore, 21 Lower Kent Ridge Road, Singapore 119077 (SG). SUKUMAR WARRIER, Vinay; c/o National University of Singapore, Cancer Science Institute of Singapore, 21 Lower Kent Ridge Road, Singapore 119077 (SG). PERERA, Amaratunga Vithanage Akila Ravihansa; c/o National University of

Singapore, Cancer Science Institute of Singapore, 21 Lower Kent Ridge Road, Singapore 119077 (SG).

(74) Agent: DAVIES COLLISON CAVE ASIA PTE. LTD.; 10 Collyer Quay #07-01, Ocean Financial Centre, Singapore 049315 (SG).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) Title: SYSTEM AND METHOD FOR EXTRACTING DATA

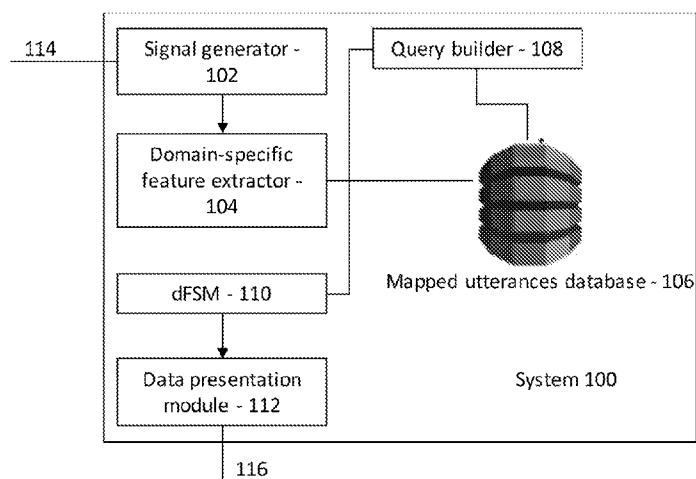


Figure 1

(57) Abstract: Disclosed is a system for extracting data from a database. The system comprises a signal generator for generating a signal corresponding to a verbal utterance of a user, a domain-specific feature extractor for extracting one or more domain-specific features from the signal, a mapped utterances database storing features of domain-specific attributes, and a query builder. The query building identifies one or more session attributes by mapping the one or more domain-specific features to said features in the mapped utterances database, each session attribute corresponding to a domain-specific attribute, and builds a query based on the one or more session attributes. The system further comprises a deterministic finite state machine having a unique state corresponding to the query, a data presentation module for extracting data from a database based on the unique state, and presenting the data to the user.

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

SYSTEM AND METHOD FOR EXTRACTING DATA

Technical Field

The present invention relates, in general terms, to a system a method for extracting data from a database. More particularly, the present invention relates to extracting data from a database based on domain-specific verbal utterances from a user.

Background

In various fields of endeavour, particularly scientific endeavour, massive amounts of data are being produced very rapidly. Some databases aggregate data from sources around the globe, resulting in a rapidly expanding data set.

In scientific research, database queries often require highly domain-specific knowledge and terminology that is difficult to input in the correct format to facilitate querying of large amounts of data. Taking genomics, for example, advances in high-throughput sequencing technologies have been a mixed blessing. Large-scale data production – while a driving force for biomedical discovery – frequently outpaces interrogation capabilities. Without implementing proper tools, the disparity between information and *bona fide* biological knowledge will continue to grow.

It would be desirable to overcome or [*alleviate/ameliorate*] at least one of the above-described problems, or at least to provide a useful alternative.

Summary

Disclosed herein is a system for extracting data from a database, comprising:

- a signal generator for generating a signal corresponding to a verbal utterance of a user;

- a domain-specific feature extractor for extracting one or more domain-specific features from the signal;

a mapped utterances database storing features of domain-specific attributes;

a query builder for:

identifying one or more session attributes by mapping the one or more domain-specific features to said features in the mapped utterances database, each session attribute corresponding to a domain-specific attribute; and

building a query based on the one or more session attributes;

a deterministic finite state machine having a unique state corresponding to the query; and

a data presentation module for extracting data from a database based on the unique state, and presenting the data to the user.

In embodiments, each domain-specific attribute is one or more of an entity type – e.g. gene, illness – and value – e.g. BRCA-1, breast cancer. Each domain-specific features may therefore be a verbal, speech or otherwise pronunciation based feature of the attribute when spoken.

The verbal utterance may comprise an initial verbal utterance and a further verbal utterance, the query builder being then configured to:

build the query based one or more said session attributes corresponding to the initial verbal utterance (**initial session attributes**); and

update the query based on one or more said session attributes corresponding to the further verbal utterance (**further session attributes**). The query builder may be configured to update the query by adding the further session attributes to the query. The query builder may be configured to update the query by replacing one or more initial session attributes with one or more corresponding further session attributes.

The query builder may further be configured to determine:

a change of topic based on the further session attributes; and

to build the query based only on the further session attributes.

The query building is further configured to determine:

a change of context based on the further session attributes; and

to build the query based on a proper subset of the initial session attributes.

The receiver may comprise a domain non-specific feature extractor for determining domain non-specific features of the verbal utterance, and the signal is generated based on the domain non-specific features.

The data presentation module may be in communication with a plurality of databases, and:

- determine, based on the unique state, at least one database from the plurality of databases, the at least one database comprising the data; and
- extract the data from the at least one database.

The data presentation module may comprise a database application program interface (dAPI) for mapping at least one of the unique state and query to a query interpretable by each database of the at least one database.

Also disclosed herein is a method for extracting data from a database, comprising:

- generating a signal using a signal generator, the signal corresponding to a verbal utterance of a user;

- extracting, using a domain-specific feature extractor, one or more domain-specific features from the signal;

- identifying, using a query builder, one or more session attributes by mapping the one or more domain-specific features to features stored in a mapped utterances database (stored features), each session attribute corresponding to a domain-specific attribute associated with the stored features

- building a query based on the one or more session attributes;

- determining a unique state of a deterministic finite state machine (dFSM), corresponding to the query; and

- extracting data from a database based on the unique state of the dFSM, for presentation to the user.

The verbal utterance may comprise an initial verbal utterance and a further verbal utterance, and building the query comprises:

- building the query based on one or more said session attributes corresponding to the initial verbal utterance (**initial session attributes**); and

updating the query based on one or more said session attributes corresponding to the further verbal utterance (**further session attributes**).

Updating the query may comprise adding the further session attributes to the query. Updating the query may comprise replacing one or more initial slot values extracted from the initial verbal utterance. The session attributes that are stored for each turn of a multi-turn conversation are the slot values. Thus, "session attributes", in the context of stored session attributes or attributes that persist over multiple turns in a multi-turn conversation, refers to stored slot values from a previous (e.g. initial) verbal utterance.

Building the query may comprise:

- determining, using the query builder, a change of topic based on the further session attributes; and
- building the query based only on the further session attributes.

Building the query may comprise:

- determining, using the query builder, a change of context based on the further session attributes; and
- building the query based on a proper subset of the initial session attributes.

The method may further comprise:

- extracting, at the signal generator, one or more domain non-specific features of the verbal utterance; and
- generating the signal based on the domain non-specific features.

Extracting data from the database may comprise:

- determining, based on the unique state, at least one database, from a plurality of databases, comprising the data; and
- extracting the data from the at least one database.

Extracting the data from the database may comprise mapping at least one of the unique state and query to a query interpretable by each database of the at least one database.

Also disclosed herein is a method for recognising out-of-vocabulary terms, comprising:

- storing a list comprising one or more out-of-vocabulary (OOV) terms in a mapped utterances database;

- identifying, from the list, at least one entity type and entity value for each OOV term;

- receiving a signal from each of one or more remote terminals, each signal comprising one or more speech recordings, each recording corresponding to a known one of said OOV terms;

- updating the mapped utterances database based on the one or more speech recordings; and

- producing an automatic speech recognition (ASR) correction bundle based on the updated mapped utterances database.

The method may further comprise transforming an output of an ASR model based on the ASR correction bundle. The ASR correction bundle is built based on outputs from a specific ASR model. Thus, each ASR correction bundle is specific to a particular ASR engine associated with the specific ASR model.

The OOV terms may comprise terms used in the field of cancer genomics.

Advantageously, the system analyses verbal utterances. This negates the need to provide and use a graphical user interface for query inputting, and streamlines the time it takes getting from question to answer.

Advantageously, the system and methods facilitate domain-specific automatic speech recognition (ASR) correction. In the genomics domain, for example, gene names do not explicitly fall under acronyms or abbreviations in conversational speech. The present system and methods can help resolve the exact gene the user is uttering within reasonable variation in pronunciation. The system and methods may also be able to disambiguate different ways of uttering cancer types and genomics data types by extension.

Advantageously, using the method for recognising OOV terms, enables collection of OOV term pronunciations, using various input devices such as Alexa®, as

labelled data useable to build or update a machine learning model (e.g. using an ASR correction bundle to transform an output of the ASR model) to augment the domain-specific analysis and data extraction service provided by the system and method as set out above. Moreover, domain expertise can be leveraged to navigate progressive query building, or query flow, designed with specific states that deliver speech response (presented by the data presentation module) accompanied by graphical outputs depending on the device the user is using the access the system or use the method. This feature helps project data visualization onto, for example, Alexa Presentation Language (APL) compatible devices such as Echo Show™ and Fire TV Stick™.

Advantageously, embodiments of the system and methods described herein provide a generic framework to capture key information in a user utterance as attributes and track those attribute changes during a multi-turn conversation. This framework enables a voice application to function more naturally by creating a user context based on the attribute values. This is achieved by implementing a finite state machine. Pragmatically, the system and method allow users to conversationally formulate complex database queries.

Brief description of the drawings

Embodiments of the present invention will now be described, by way of non-limiting example, with reference to the drawings in which:

Figure 1 schematically illustrates a system for extracting data from a database in accordance with present teachings, in terms of functional blocks;

Figure 2 illustrates an overall system or network of devices for implementing end-to-end analysis of verbal utterances in accordance with the present disclosure;

Figure 3 illustrates a method performed on a system in accordance with Figure 1;

Figure 4 is a block diagram of an example device for physically implementing the system of Figure 1, in terms of physical components;

Figure 5 illustrates state-based navigation flow performed in the system of Figure 1;

Figure 6 illustrates a method for recognising OOV terms in accordance with the present teachings;

Figure 7 illustrates an OOV mapper model;

Figure 8 is a schematic process flow for collecting and sourcing pronunciations for OOV terms;

Figure 9 illustrates automatic speech recognition training and deployment phases;

Figure 10 is a schematic deployment architecture for deploying the model generated in accordance with the method of Figures 6 and 7;

Figures 11 and 12 illustrate plots outputted by the system, or result from the method, disclosed herein in response to particular queries;

Figure 13 is a visual output from invocation of the 'compare' intent or Compare By function for contrasting TP53 mutations and copy number alterations across all cancer types from The Cancer Genome Atlas; and

Figure 14 is a visual output from invocation of the 'split-by' intent or Split-By function for evaluating mRNA expression differences in TP53 within CDH1 mutant (MUT) and wild type (WT) patients.

Detailed description

Disclosed herein is a system for extracting data from a database. The system includes a query builder. The query builder enables, among other things, updating of a previously built query based on verbal utterances from the user. This type of update can include identifying changes in topic or context. By updating the query, rather than replacing the query, domain-specific feature interrogation is implemented on top of the state-tracking framework described above. This creates a natural conversational experience when performing complex genomic analysis using voice. Instead of forcing users to form long and complex utterances, which are common during scientific inquiry, the present system and method may allow them to traverse a stepwise navigation flow expressing values for attributes of interest (e.g. gene, cancer type, data type, and dataset). This framework may therefore mimic the systematic human thought process and reduces users' cognitive load.

Embodiments of the system and method provided herein leverage domain-specific knowledge. In extracting domain-specific attributes, which can include user intent in a particular domain, intent-mapping schemes are employed that allow users to perform conventional cancer genomics queries with simple voice commands. In addition to vocal responses, graphical results can be delivered to users via their input device – e.g. using Alexa Presentation Language. The mapped utterances database is easily expandable, and may draw from existing databases – e.g. for cancer genomics the mapped utterances database may amalgamate data from the popular The Cancer Genome Atlas resource. The mapped utterances database may comprise a hierarchical transposition of the data to facilitate a deterministic finite state machine, allowing users to intuitively and systematically traverse analytical levels such as genes, cancer types, and data types. Embodiments of the present invention may therefore support interactive, exploratory analyses as well as queries that cannot be reduced to a single, coherent utterance.

With reference to Figures 1 and 2, a system 100 is depicted for extracting data from a database. The system broadly comprises:

- Signal generator 102;
- Domain-specific feature extractor 104;
- Mapped utterances database (MUDB) 106;
- Query builder 108;
- Deterministic finite state machine (dFSM) 110; and
- Data presentation module 112.

The system 100 receives an input 114, which may take the form of a verbal utterance or query delivered directly to the system 100 from a user, or may be a pre-processed signal from an external device or service. The signal generator 102 generates a signal corresponding to the verbal utterance, based on the input.

The domain-specific feature extractor 104, which may constitute part of the signal generator 102, extracts one or more domain-specific features from the signal. The domain-specific features are features that correspond to features of domain-specific attributes stored in the MUDB 106.

The query builder 108 takes the extracted domain-specific features and maps them to features in the MUDB 106. The query builder 108 can therefore identify one or more session attributes corresponding to domain-specific attributes the features of which are stored in the MUDB 106.

The query builder 108 then builds a query based on the session attributes. The session attributes are then used to determine a specific state of the dFSM 110 that is, in effect, a query structure from which the data presentation module can interpret the state and extract data from a database (not shown). The database may be remote, or may comprise part of the system 100.

The dFSM 110 then assumes a unique state depending on the query – i.e. based on the session attributes. This state is used by the data presentation module 112 to extract data from a database and present it to the user. The system 100 outputs the data (at 116), which may be directly displayed on a display, verbally provided or as a message – e.g. email message, short-message-service (SMS) message and others.

With reference to Figure 2, a system 200 is shown that can implement the schematic system 100 of Figure 1. The system 200 includes five major components: an intent handler 202 and interaction model (conversation design) 204 that together provide a Skills Service ("SS"), genomic (or other domain-specific) terminology mapper (GTM) service 206, notifications and information relay service 210, and analysis engine 208.

The intent handler 202 comprises an invocation model that describes the possible set of user intents (which may be domain-specific) and sample phrases to trigger those intents. The intent handler 202 and interaction model 204 are responsible for tracking state attributes based on the intent name and slot values then forming a structured query to retrieve analysis results from the analytics engine 208. The final result is sent back to the user in the form of speech response (in SSML) and visual content in the form of Alexa Presentation Language (APL) which can be rendered in a supported Alexa device if one was used to supply the original verbal utterance.

The analysis engine 208 performs the analysis using a genomic database hosted in the cloud using AWS RDS technology. It uses various components (specifically, entity resolution via GTM service 206 in the present embodiment) to process the structured queries received from SS. The entity resolution is performed via the GTM Service 206 which currently maps an intent slot value to a gene name, cancer type or data type.

In the embodiment shown in Figure 2, the signal generator may comprise device 206 – e.g. a smartphone or smartwatch – that includes a microphone.

The domain-specific feature extractor and query builder may comprise or form part of the SS which, as described above, comprises intent handler 202 and interaction model 204. The domain-specific feature extractor presently also comprises pre-processor 207 that receives a signal from device 209. The pre-processor 207 extracts attributes including user session information, intent name and slot value.

The intent handler 202 handles requests received from the pre-processor 207, to help build queries. The requests received contain session attributes – e.g. the user session information (e.g. a session identifier (ID) to uniquely identify each conversation or session, session attributes including a mapping of slot/key-value pairs, and/or user account information), intent name, and slot values resolved by the pre-processor 204. These slot values are often inaccurate due to lack of domain-specific knowledge in natural language knowledge databases. Therefore, the SS attempts to map the slot value resolved by the pre-processor 204 to its corresponding domain-specific entity type and value via GTM Service 206. The GTM service 206, comprises repository 211 (including MUDB 106) that stores features of domain-specific attributes. The GTM Service 206, forming part of the query builder 108, maps the features extracted by the pre-processor 207 to domain-specific features of attributes stored in repository 211. Thus, the session attributes (e.g. entity or slot type such as 'gene_name', and slot values such as 'TP53') identified by reference to domain-specific attributes the repository 211.

Once the slot value is mapped, intent handler 202 will update its state by storing the corresponding attribute type and value (e.g. "gene_name: TP53") as a session

attribute – i.e. the dFSM 110 determines its state based on the session attributes. Storing state attributes as session attributes enables the intent handler 202 to track the state of the dialogue flow since previously set session attributes are persisted across subsequent user requests in a multi-turn conversation. A multi-turn conversation may include breaking down a single user verbal utterance into multiple components (turns) each of which can be used to identify a single or plurality of session attributes. Alternatively, a multi-turn conversation may include taking a verbal query, forming part of the verbal utterance, in its entirety and extracting all slot values from the verbal query. For a verbal utterance comprising a series of verbal queries, the slot values from each previous verbal query in the utterance, or in a previous verbal utterance in the multi-turn conversation, become session attributes for one or more subsequent verbal queries or utterances.

The intent handler 202 may first validate the state change triggered by the user utterance before it attempts to formulate a structured query based on the current state (attribute values). For example it may not be possible to query for mutations (data_type: mutations) when gene name is not provided (gene_name: NULL). The intent handler 202 therefore awaits sufficient information – i.e. session attributes – to determine a state of the dFSM.

Once the state is validated intent handler 202 will wrap the structured query in an HTTP request. The request is then sent, e.g. by data presentation module 112, for analysis – e.g. over the Internet to a remote endpoint where Analysis Engine 208 is hosted – to retrieve the corresponding result (data). The result may contain structured data which is used to generate a speech response based on a predefined template. In addition to the speech response, most responses contain a visual element (plots, maps, tables, etc.) which is delivered using Alexa APL technology.

To ensure the correct data can be returned, the data presentation module 112 may determine, based on the unique state of the dFSM, that a particular on (or more) database has the desired information and extract the data from the database. This enables the data presentation module to be in communication with multiple databases, without having to consult all of the databases for each

validated dFSM state. Similarly, where databases require a specific query format or specific query content, the data presentation module 112 may comprise a database application program interface (dAPI) for reformatting the query or augmenting the query with the specific content necessary for the database to interpret the query and retrieve the desired data.

It will be appreciated that various portions of the components identified in Figures 1 and 2 and others can be combined, or distributed across multiple parts in a system and, for example, specific blocks shown in Figure 2 do not necessarily comprise the entire functionality of any particular module identified in Figure 1, or may comprise the functionality of more than one such module.

The system may therefore perform a method 300, as shown in Figure 3, for extracting the result (i.e. data) from repository 211. The method 300 may broadly include (step 302) extracting domain-specific feature(s) from a signal – e.g. using pre-processor 204 – (step 304) extract domain specific features using the SS, (step 306) identifying session attribute(s) by mapping session features to domain-specific attributes via GTM Service 206, (step 308) building the query and (step 310) determining a unique state of the dFSM based on the validity of the query, and (step 312) extracting data from a database based on the state of the dFSM – i.e. the query once sufficiently complete to resolve a state of the dFSM.

In a multi-turn conversation the verbal utterance may comprise an initial verbal utterance and a further (i.e. subsequent) verbal utterance. The query engine therefore builds the query based on the session attribute(s) corresponding to the initial verbal utterance and updates the query based on the session attribute(s) of the further or subsequent verbal utterance. This may include adding attributes – e.g. where only the specific gene has been identified in the initial session attributes, the updated query may include the gene and a type of cancer associated with mutations in that gene – or substituting attributes – e.g. replacing "breast cancer" for "gastric cancer". In this way, a complex query can be built from multiple, less complex verbal utterances (queries) in a more conversational context.

The conversational dialogue flow 400 for interrogating genomics datasets is facilitated by implementing a state-tracking framework as depicted in Figure 4. In the present example, 3 attributes are built-in – gene, cancer type, data type – that are being tracked during a user session. It should be noted that any number of attributes can be added into this framework. Users can update the attribute values by using sample phrases and arguments as defined in the interaction model 204. This approach allows users to take multiple navigation paths to get the same result thereby increasing user engagement.

In the example shown, an initial query 402 is received – "Tell me about TP53". The intent 'tell me about' is identified and the query or slot value 'TP53' is identified. Thus, the 'gene' slot value is known. The user then states "show me mutation" at 404 – the slot value 'TP53' remains unchanged, but the dFSM now has a state consisting of 'gene' with slot value 'TP53' and 'data type' with slot value 'mutation', that is used to determine which database or data subset to extract from. The third slot 'illness' or 'cancer' in the present context, is filled at 406 by 'Show me in breast cancer' – the slot value being 'breast cancer'. An attribute is then switched at 408, by the query 'How about gastric cancer'. The query builder updates the query by substituting or replacing 'breast cancer' with 'gastric cancer', but otherwise leaving the 'gene' and 'data type' slot values unchanged.

In some embodiments, the query builder determine a change in context based on the further session attributes – i.e. session attributes of follow-up user utterances – which may include intent. In the present example, the context change materialises as a reference to only a subset of the slot values – i.e. other values are not relevant. At 410, the user asks 'What does it do?'. This may be a verbal fixed query having a known function indicating a change in context. Alternatively, it may be a reference to the 'gene' slot value only, thereby indicating a change in context, and the other two slot values are cleared. The same process is detected where previous slot values are incompatible with an updated or substituted slot value, and the incompatible values are removed from the query.

At 412, the user states 'Show me copy number changes', which adds the new 'data type' attribute. At 414, the user then asks 'How about CDH1', which

substitutes slot value 'TP53' with 'CDH1'. At 416 the user states "Tell me about ovarian cancer". The query builder interprets "Tell me about" as a topic change and clears the slot values except for the slot values delivered with the topic change utterance. At 418 and 420 the user adds slot values 'BRCA1' and 'expression' to completely define values for all slots.

The interaction model 204 defines the voice interface of the SS through which users will interact with the system. The interaction model 204 captures the mapping between user input utterances and their corresponding intents with possible arguments. The interaction model 204 describes conversational sample phrases that can be used to trigger state changes – e.g. "tell me about" or "tell us about" may correspond to a topic change, whereas "show me mutations" may correspond to addition of a slot value "mutations". Similarly, "shown me mutations in" includes slot value "mutations" and would be expected to precede a slot value for the 'gene' slot. Following are some examples of intents and their corresponding sample phrases with argument placeholders that can be used to trigger them.

- NavigateMutationsIntent
 - "show me mutations"
 - "show me mutations in {query}"
- NavigateStartIntent
 - "tell me about {query}"
 - "tell us about {query}"

The notifications and information relay service 210, forming part of the data presentation module 112 of the embodiment in Figure 1, handles sending information collected during a user session back to the user via email and/or other notification service (e.g.: SMS, Slack, Messenger). It also implements account linking features – e.g. to an Alexa account – that allows the present system to integrate with third-party tools to extend its functionality. This may also be used to extract pronunciation information for the individual user from a speech recognition database, to enable refinement or interpretation of verbal utterances that would otherwise be considered unclear.

Analysis Engine 208 may be developed using various architectures to achieve the foregoing functionality. Presently, a microservices architecture has been used for scalability and interoperability to enable penetration into different market segments by, for example, updating the domain-specific knowledge (terms, intent etc) for the market segment of interest. Analysis Engine 208 is composed of multiple subcomponents – the data model 212, analytics modules 214 – one for each slot –, plotting modules 216, and REST API 218. The data model 212 is hosted as a MySQL database which may be in, for example, Amazon Web Service (AWS) Relational Database Service (RDS) Aurora platform. Analytics modules (e.g.: Mutations Analysis, CNA Analysis) may use a Python-MySQL Object Relational Mapper toolkit called SQLAlchemy to build SQL queries to access data. This allows for seamless retrieval and integration of complex query inputs pertaining to any data in the database. The analytics modules 214, individually, also generate visual outputs to supplement the complex query retrievals by calling sub methods that have plotting functionality.

In an example implementation relating to genomics queries, the present system uses Amazon Aurora, managed by Amazon RDS to store updated versions of select datasets from large public data resources such as the GDC (Genomic Data Commons) and TCGA (The Cancer Genome Atlas). Gene definitions and location metadata may be sourced from National Centre for Biotechnology Information (NCBI) portal. Gene mutation data may be obtained from GDC portal while the RNA expression data may be from UCSC Toil RNAseq Reanalyze Compendium – thus multiple databases are connected to the data presentation module that selects the appropriate database(s) from which to extract the data, based on the state of the dFSM (i.e. the query). The data landscape is built around patient information ('CASE'), genes ('GENE'), cancer types ('STUDY'), mutations ('MUTATION') and gene expression ('EXPRESSION'). These are extracted, transformed, and loaded (ETL) as tables in the Aurora RDS and pre-computed data tables are generated for copy number alterations ('CNA') and gene expression correlations ('CORRELATION'). Melvin interacts with its database using SQLAlchemy Object Relational Mapper toolkit.

Various functions within the Analysis Engine 208 serve to generate answers to analytical/computational queries pertinent to the above mentioned data

landscape. Every analysis subsystem has its own visualization function that produces relevant plot accompaniments to go with the speech response to a query. For instance, a query related to Mutation Analysis going by "Tell me about CDH1 ... Show me mutations ... How about Ovarian Cancer? " would result in a speech response coupled with a plot at each stage of response. This results in data-driven actionable insights that could be utilized for continual research. An example plot from the query "Tell me about CDH1... show me mutations" is shown in Figure 11. Similarly, an example plot from the subsequent query "How about ovarian cancer" is shown in Figure 12.

The GTM service 206 is the component in the software infrastructure that helps pinpoint accurately what domain specific term or phrase a user has uttered. The algorithmic workflow of the GTM Service is shown in Figure 5. Device 209 captures the voice input from the user and provides its automatic speech recognition (ASR) outputs for further processing (the ASR functionality may alternatively be incorporated into the signal generator). These, termed 'query utterances' or 'verbal utterances', undergo a train of processes through 3 main pipelines - Exact Matching, Mapped Utterances Mapping, and Fuzzy String Matching.

In the **Exact Matching** pipeline, the incoming query utterance is cleaned, case transformed and lemmatized to match exact entities (gene, cancer type, or data type) from a lookup. In cases where least disambiguation is required, this stage should yield result, thereby foregoing subsequent pipelines.

The next pipeline, **Mapped Utterance Matching**, exploits the orthogonal skills, such as orthogonal Alexa skill, used to crowd-source OOV pronunciations as described with reference to Figures 8 and 9. These labelled OOV utterances are stored in a table in the Aurora RDS database for convenient updating and lookup. The query utterance that doesn't have an exact match with the 'true' form entities, is now matched with the mapped utterances. The performance of this pipeline is dependent on the distribution of entity specific pronunciation variations captured in the utterance database.

Failing exact matches, the query utterance is fed to the third pipeline that does **Fuzzy String Matching**. This pipeline is an approximate match algorithm using

multiple transformations and data representations to compare and calculate a quantifiable similarity score commonly used in fuzzy string matching methods. Since genes consist of numbers in their names, they need to be represented in their textual form. For instance, 'BRCA2' is inflected to 'BRCATWO'. The key element in this stage is the phonetic encoding of pre-processed query utterances (e.g. processed using heuristic-based string manipulation). This can be done, for example, using the Metaphone algorithm that reduces any text token into a combination of 16 consonants, preserving the phonetic representation of the text. The target entities from the mapped utterances database and the exact match pool are transformed into this representation. Cosine similarity metrics are calculated between the *processed*, *processed + encoded*, and *processed + inflected + encoded* utterance-target pairs. The ranked top match target entity is delivered as the output of this pipeline.

In addition to building queries through multiple utterances, the present methods can also be applied to comparative analyses – per step 314 of Figure 3, shown in broken lines as it is not compulsory in some applications of the method 300. Examples of such comparative analyses include **Compare By** and **Split-By** functionality. To that end, the data presentation module 112 may extract data from the database based on the unique state, as previously discussed. The method 300 may then present the data to the user as previously discussed. Step 314 then involves receiving a verbal utterance or utterances corresponding to a compare request (i.e. receive a signal corresponding to that utterance or those utterances) and updating the query based on one or more sessions attributes that correspond to the utterance or utterances and compare functionality.

Taking Compare By functionality for example: data analysts and researchers may want to compare one attribute value to another within the same cohort or to compare different cohorts with respect to a particular attribute value or values. Thus the session attribute or attributes corresponding to the compare functionality may include an attribute corresponding to Compare By functionality, and an attribute value in respect of which the comparison can be made and, optionally, attributes corresponding to the cohorts (e.g. cohorts for two cancer types) being compared with respect a particular attribute (e.g. a particular mutation).

In an illustrative genomics example for comparison based on a particular attribute, each patient sample will have values for multiple data types (mutations – e.g. insertion and deletion variants and single nucleotide variants [indels and SNVs], copy number alterations [amplifications and deletions], and messenger ribonucleic acid (mRNA) expression). An analyst may want to know the relationship between mutations and copy number alterations (CNA) in the TP53 gene in breast cancer patients. An example dialogue flow (i.e. query or sequence of utterances) that can invoke the Compare By functionality on this basis may be:

User/analyst: "Tell me about TP53"

System output (e.g. audio): "Ok, TP53. What would you like to know?"

User/analyst: "Show me mutations"

System output: "TP53 mutations are found in 31 cancer types. It is most mutated in Ovarian serous cystadenocarcinoma at 70 percent followed by Lung squamous cell carcinoma at 68 percent."

User/analyst: "Compare to copy number alterations"

System output: "Across all of TCGA, TP53 is mutated in 19.4 percent of all cases and has copy number alterations in 5.7 percent of cases."

The above responses (i.e. outputs) may be audio responses, or text produced on a screen. In some cases, the response may be one or more figures showing the comparison data. Such figures can be progressively refined as more attributes of a query are received and/or updated in a dialogue flow (this can take place for all methods described herein and not only those that employ compare functionality). A representation of a pictorial output (i.e. figure) of a compare function is shown in Figure 13. For each cancer type, Figure 13 displays the proportion of patients that have mutations only, CNA only, both mutations and CNAs, or neither.

The compare functionality is not limited to the data type attribute. As mentioned above, the compare functionality can be utilised to contrast a single data type (e.g. mutations) across two cohorts – e.g. cancer types or genes (e.g. TP53 & CDH1). An example dialogue flow (i.e. multi-turn conversation) for achieving this is:

User/analyst: "Tell me about TP53"

System: "Ok, TP53. What would you like to know?"

User/analyst: "Show me mutations"

System: "TP53 mutations are found in 31 cancer types. It is most mutated in Ovarian serous cystadenocarcinoma at 70 percent followed by Lung squamous cell carcinoma at 68 percent."

User/analyst: "Compare to CDH1"

System: "CDH1 and TP53 mutations are found in 74 and 94 percent of cancer types respectively. Across all of TCGA, CDH1 is mutated in 18 percent of all cases while TP53 is mutated in 42 percent"

Now with respect to the Split-By functionality: analysts and researchers may wish to understand relationships within or across data types. The Split-By functionality accommodates such queries. Split-By contrasts a categorical or a quantitative variable that has been partitioned by a second categorical variable (i.e. split-by the second categorical variable). In this case, session attributes extracted from the utterance or utterances corresponding to the Split-By function will include the categorical or quantitative variable and the second categorical variable. An example of a Split-By function invoked for data type comparison may be:

User/analyst: "Tell me about TP53"

System: "Ok, TP53. What would you like to know?"

User/analyst: "Show me mutations"

System: "TP53 mutations are found in 31 cancer types. It is most mutated in Ovarian serous cystadenocarcinoma at 70 percent followed by Lung squamous cell carcinoma at 68 percent."

User/analyst: "Show me breast cancer"

System: "TP53 mutations are found in 30 percent of Breast invasive carcinoma patients with 53 amino acid residues recurrently mutated."

User/analyst: "Split-by mutations"

System: "Which gene would you like to split by?"

User/analyst: "CDH1"

System: "In Breast invasive carcinoma, TP53 mutations show mutual exclusivity with CDH1 mutations with a Fisher's Exact P value of 0.02"

As a result of the above query flow or sequence of utterances (i.e. multi-turn conversation), breast cancer patients are grouped into four categories: 1) TP53 mutant & CDH1 mutant; 2) TP53 wild type & CDH1 mutant; 3) TP53 mutant & CDH1 wild type; and 4) TP53 wild type & CDH1 wild type. The counts in each category can be presented in a 2x2 contingency table structure – e.g. TP53 mutant/TP53 wild type on one side, and CDH1 mutant/CDH1 wild type on the other side. Similar to the previous dialog flows, audio responses will can be replaced with or supplemented by visual responses at several steps in these multi-turn conversations.

Split-By can also be used across data type, the following multi-turn conversation being an example for invoking such Split-By functionality:

User/analyst: "Tell me about TP53"

System: "Ok, TP53. What would you like to know?"

User/analyst: "Show me mutations"

System: "TP53 mutations are found in 31 cancer types. It is most mutated in Ovarian serous cystadenocarcinoma at 70 percent followed by Lung squamous cell carcinoma at 68 percent."

User/analyst: "Show me breast cancer"

System: "TP53 mutations are found in 30 percent of Breast invasive carcinoma patients with 53 amino acid residues recurrently mutated."

User/analyst: "Split-by expression"

System: "Which gene would you like to split by?"

User/analyst: "CDH1"

System: "In Breast invasive carcinoma, TP53 is not differentially expressed between CDH1 mutant and wild type samples with a Mann Whitney P value of 0.44"

In the above example, data type value 1 (mRNA expression of TP53) is being partitioned by data type value 2 (patients who are wild type [WT] or mutant [MUT] for CDH1), each data type value being a session attribute in this instance. A graphical or pictorial output depicting the result of this query-based compare function invocation is set out in Figure 14.

The Split-By function can further include additional statistical analyses – e.g. for hypothesis testing. Step 314 may therefore further involve receiving another utterance comprising a test (e.g. statistical test) to be applied as part of – i.e. to the output of – the comparison function.

In the Split-By case, Split-By will perform statistical tests that evaluate the cohorts against one another. In the above multi-turn conversation resulting in production of a 2x2 contingency table, a Fisher's Exact test can be performed as the underlying data can be represented by a 2x2 contingency table. When two categorical groups (e.g. mutant vs non-mutant) are compared by a quantitative variable, hypothesis testing can be performed with a statistical method such as a Mann-Whitney Rank test – the example output being shown in Figure 14. The application of statistical hypothesis testing via voice commands can be extrapolated to many different statistical methods and attribute types – both within and outside of the genomics domain.

The present method and system have foreseeable applications in a wide variety of fields of research where large amounts of data are continually generated. It is foreseeable that new fields will have varying lexicons depending on the origin of the data, terms of the art in particular geographical regions and so forth. At least initially, there are likely to be terms that cannot be resolved with a high degree of confidence or terms that are simply not present in the lexicon known to the system – these will be called out-of-vocabulary (OOV) terms.

Figure 6 broadly sets out a method 600 for recognising OOV terms. The method includes:

602: storing a list comprising one or more OOV terms in a mapped utterances database (MUDB) – the terms in the list are labelled and, as yet, may not be mapped to utterances. Thus the MUDB may comprise a memory associated with mapped terms (i.e. utterances), and a memory associated with OOV terms that are unmapped;

604: identifying, from the list (i.e. from the labels), at least one entity type and entity value for each OOV term;

606: receiving a signal from each of one or more remote terminals – these are the terminals of the contributors of the crowd sourced speech recordings the correspond to known ones of the OOV terms;

608: updating the mapped utterances database based on the one or more speech recordings; and

610: producing an ASR correction bundle based on the updated mapped utterances database. An output of the ASR model can then be transformed based on the correction bundle produced at step 610. This facilitates correction of interpretation of the ASR model output. The bundle is the artefact produced by the framework, which will be deployed as a service to be consumed.

This process is described further, with reference to the framework set out in Figures 7 and 8. The framework consists of three subsystems: pronunciation crowdsourcing phase 702, model training phase 704 and service deployment phase 706.

Phase 702 is a data collection phase intended for generating the training data necessary for machine learning modelling in phase 704. The key components of the system 700 for implementing phase 702 include a crowdsourcing executor 708 and a crowdsourcing marketplace 710. An organization/user can submit a list of out-of-vocabulary (OOV) terms (received and stored per step 602). These OOV terms need automatic speech recognition (ASR) correction. The executor subsystem handles routing incoming OOV terms through text-to-speech (TTS) conversion modules and ASR modules to fill the mapped utterance repository. This may be achieved using a purely software approach, or by external crowdsourcing platforms to collect pronunciation samples corresponding to known ones of the OOV terms, in a prescribed format (e.g. based on labels accompanying the list of OOV terms). A task user interface may be used for on-site participants – i.e. internal crowdsourcing.

The marketplace subsystem 710 is utilized when an external crowdsourcing platform is employed in the executor subsystem. For instance, a popular crowdsourcing platform is Amazon's Mechanical Turk (MTurk) that provides the required infrastructure for *workers* to complete *requester* defined (us, in this case) Human Intelligence Tasks (HITs). The marketplace subsystem takes care of

managing the worker information, worker recruitment, and MTurk's worker reward system.

The service user organization submits a list of out-of-vocabulary terms tagged by their entity type (for instance, {'car': 'Maserati', ...}). The requester builds a task and deploys it to MTurk with a price assigned. Workers receive this monetary value as payment for satisfactory task completion – i.e. useable utterances of specific ones of the OOV terms. Each task will stem from the input list split into batches for tracked parallel processing. The task would involve obtaining a minimum of 1 pronunciation sample for each list term for a single worker. The workers will be vetted before being given the task, for their proficiency in spoken English, familiarity with the domain, and demographic information – the demographic information may assist in adapting the SS to account for geographical variations in pronunciation of domain-specific terms. The recording service used for this HIT can be a multimodal web interface designed, for example, using local Amazon MTurk features or an external service that uses MTurk only for worker recruitment. MTurk allows iterative improvement of the interface and subsequently the task to fine-tune worker experience in initial trial runs.

A task validation service may be set up as a quality control to weed out poor or malicious data. In scenarios where standard pronunciations curated from domain experts is available, they shall also be used for validation of collected voice data.

In the training phase – phase 704 – the data collected from phase 702 is stored in the MUDB 712 that is used to train a machine learning model – see Figure 8. The key components of phase 704 mimic those of a production level machine learning (ML) pipeline. The incoming database consists of ASR output samples and their labelled OOV terms. An utterance processor 714 performs the data cleaning, data pre-processing, and feature extraction/transformation. The resulting representation of utterance data could be a feature-set, an embedding space, or a combination of both. A machine learning model will be then trained-validated-evaluated to learn these representative patterns on split data (716). Once complete, the resulting model is packaged into an ASR Correction Bundle 718 that is ready to be deployed as a stand-alone service.

Phase 706 is the service deployment phase. The packaged ASR correction model from the training phase 704 is deployed onto the system. For a cloud-based application, deployment will involve using cloud infrastructure or other methods to meet the requirements of the user/organization. Any external/internal client could then interact with the service endpoint by passing their ASR output as an input – in this sense, some embodiments of the signal generator may involve a distributed signal generator where part of the functionality is provided in the form of an ASR module on each user terminal. The service would then apply the deployed machine learning model and return the predicted entity type and value.

The present method has been described in the form of a system that implements it – i.e. comprises instructions that when executed result in performance of the method. That system may be a stand-alone system or, in a preferred embodiment, is deployed on the AWS Cloud platform as described above. A serverless computing model is used to build and host Analysis Engine and OOV Mapper components 206, 208 under, for example, AWS Lambda platform. This allows the system to be instantaneously scalable without manual intervention. The serverless-framework toolkit may be used to package multiple system components (Genomics Terminology Mapper service 206, Notifications and Information Relay Service 210, and Analysis Engine 208) into bundles that can be deployed via AWS Cloud Formation platform.

Figure 10 is a block diagram showing an exemplary mobile computer device 1000 in which embodiments of the invention may be practiced. The mobile computer device 1000 may be a mobile computer device such as a smart phone, a personal data assistant (**PDA**), a palm-top computer, and multimedia Internet enabled cellular telephones such as those providing voice assistant functionality. For ease of description, the mobile computer device 1000 is described below, by way of non-limiting example, with reference to a mobile device in the form of an iPhone™ manufactured by Apple™, Inc., or one manufactured by LG™, HTC™ and Samsung™, or the Echo device manufactured by Amazon™ for example.

As shown, the mobile computer device 1000 includes the following components in electronic communication via a bus 1006:

- (a) a display 1002;
- (b) non-volatile (non-transitory) memory 1004;
- (c) random access memory ("RAM") 1008;
- (d) N processing components 1010;
- (e) a transceiver component 1012 that includes N transceivers; and
- (f) user controls 1014.

Although the components depicted in Figure 10 represent physical components, Figure 10 is not intended to be a hardware diagram. Thus, many of the components depicted in Figure 10 may be realized by common constructs or distributed among additional physical components. Moreover, it is certainly contemplated that other existing and yet-to-be developed physical components and architectures may be utilized to implement the functional components described with reference to Figure 10.

The display 1002 generally operates to provide a presentation of content to a user, and may be realized by any of a variety of displays (e.g., CRT, LCD, HDMI, micro-projector and OLED displays).

In general, the non-volatile data storage 1004 (also referred to as non-volatile memory) functions to store (e.g., persistently store) data and executable code.

In some embodiments for example, the non-volatile memory 1004 includes bootloader code, modem software, operating system code, file system code, and code to facilitate the implementation components, well known to those of ordinary skill in the art, which are not depicted nor described for simplicity.

In many implementations, the non-volatile memory 1004 is realized by flash memory (e.g., NAND or ONENAND memory), but it is certainly contemplated that other memory types may be utilized as well. Although it may be possible to execute the code from the non-volatile memory 1004, the executable code in the non-volatile memory 1004 is typically loaded into RAM 1008 and executed by one or more of the N processing components 1010.

The N processing components 1010 in connection with RAM 1008 generally

operate to execute the instructions stored in non-volatile memory 1004. As one of ordinary skill in the art will appreciate, the N processing components 1010 may include a video processor, modem processor, DSP, graphics processing unit (GPU), and other processing components.

The transceiver component 1012 includes N transceiver chains (e.g. audio transceivers in communication with audio signal processors), which may be used for communicating with external devices via wireless networks. Each of the N transceiver chains may represent a transceiver associated with a particular communication scheme. For example, each transceiver may correspond to protocols that are specific to local area networks, cellular networks (e.g., a CDMA network, a GPRS network, a UMTS networks), and other types of communication networks.

It should be recognized that Figure 10 is merely exemplary and in one or more exemplary embodiments, the functions described herein may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored on or transmitted over as one or more instructions or code encoded on a non-transitory computer-readable medium 1004. Non-transitory computer-readable medium 1004 includes both computer storage medium and communication medium including any medium that facilitates transfer of a computer program from one place to another. A storage medium may be any available medium that can be accessed by a computer.

In some examples, the mobile computer device 1000 is embodied by a wearable such as a smartwatch (e.g. Apple Watch) or fitness tracker (e.g. FitBit). Alternatively, the mobile computer device 1000 is in connection with a smartwatch or fitness tracker.

It will be appreciated that many further modifications and permutations of various aspects of the described embodiments are possible. Accordingly, the described aspects are intended to embrace all such alterations, modifications, and variations that fall within the spirit and scope of the appended claims.

Throughout this specification and the claims which follow, unless the context

requires otherwise, the word “comprise”, and variations such as “comprises” and “comprising”, will be understood to imply the inclusion of a stated integer or step or group of integers or steps but not the exclusion of any other integer or step or group of integers or steps.

The reference in this specification to any prior publication (or information derived from it), or to any matter which is known, is not, and should not be taken as an acknowledgment or admission or any form of suggestion that that prior publication (or information derived from it) or known matter forms part of the common general knowledge in the field of endeavour to which this specification relates.

Claims

1. A system for extracting data from a database, comprising:
 - a signal generator for generating a signal corresponding to a verbal utterance of a user;
 - a domain-specific feature extractor for extracting one or more domain-specific features from the signal;
 - a mapped utterances database storing features of domain-specific attributes;
 - a query builder for:
 - identifying one or more session attributes by mapping the one or more domain-specific features to said features in the mapped utterances database, each session attribute corresponding to a domain-specific attribute; and
 - building a query based on the one or more session attributes;
 - a deterministic finite state machine having a unique state corresponding to the query; and
 - a data presentation module for extracting data from a database based on the unique state, and presenting the data to the user.
2. The system of claim 1, wherein the verbal utterance comprises an initial verbal utterance and a further verbal utterance, the query builder being configured to:
 - build the query based one or more said session attributes corresponding to the initial verbal utterance (**initial session attributes**); and
 - update the query based on one or more said session attributes corresponding to the further verbal utterance (**further session attributes**).
3. The system of claim 2, wherein the query builder is configured to update the query by adding the further session attributes to the query.

4. The system of claim 2, wherein the query builder is configured to update the query by replacing one or more initial slot values extracted from the initial verbal utterance.
5. The system of any one of claims 2 to 4, wherein the query builder is further configured to determine:
 - a change of topic based on the further session attributes; and
 - to build the query based only on the further session attributes.
6. The system of any one of claims 2 to 5, wherein the query building is further configured to determine:
 - a change of context based on the further session attributes; and
 - to build the query based on a proper subset of the initial session attributes.
7. The system of any one of claims 1 to 6, wherein the receiver comprises a domain non-specific feature extractor for determining domain non-specific features of the verbal utterance, and the signal is generated based on the domain non-specific features.
8. The system of any one of claims 1 to 8, wherein the data presentation module is in communication with a plurality of databases, and:
 - determines, based on the unique state, at least one database from the plurality of databases, the at least one database comprising the data; and
 - extracts the data from the at least one database.
9. The system of claim 8, wherein the data presentation module comprises a database application program interface (dAPI) for mapping at least one of the unique state and query to a query interpretable by each database of the at least one database.
10. A method for extracting data from a database, comprising:
 - generating a signal using a signal generator, the signal corresponding to a verbal utterance of a user;

extracting, using a domain-specific feature extractor, one or more domain-specific features from the signal;

identifying, using a query builder, one or more session attributes by mapping the one or more domain-specific features to features stored in a mapped utterances database (stored features), each session attribute corresponding to a domain-specific attribute associated with the stored features;

building a query based on the one or more session attributes;

determining a unique state of a deterministic finite state machine (dFSM), corresponding to the query; and

extracting data from a database based on the unique state of the dFSM, for presentation to the user.

11. The method of claim 10, wherein the verbal utterance comprises an initial verbal utterance and a further verbal utterance, and building the query comprises:

building the query based on one or more said session attributes corresponding to the initial verbal utterance (**initial session attributes**); and

updating the query based on one or more said session attributes corresponding to the further verbal utterance (**further session attributes**).

12. The method of claim 12, wherein updating the query comprises adding the further session attributes to the query.

13. The method of claim 12, wherein updating the query comprises replacing one or more initial session attributes with one or more corresponding further session attributes.

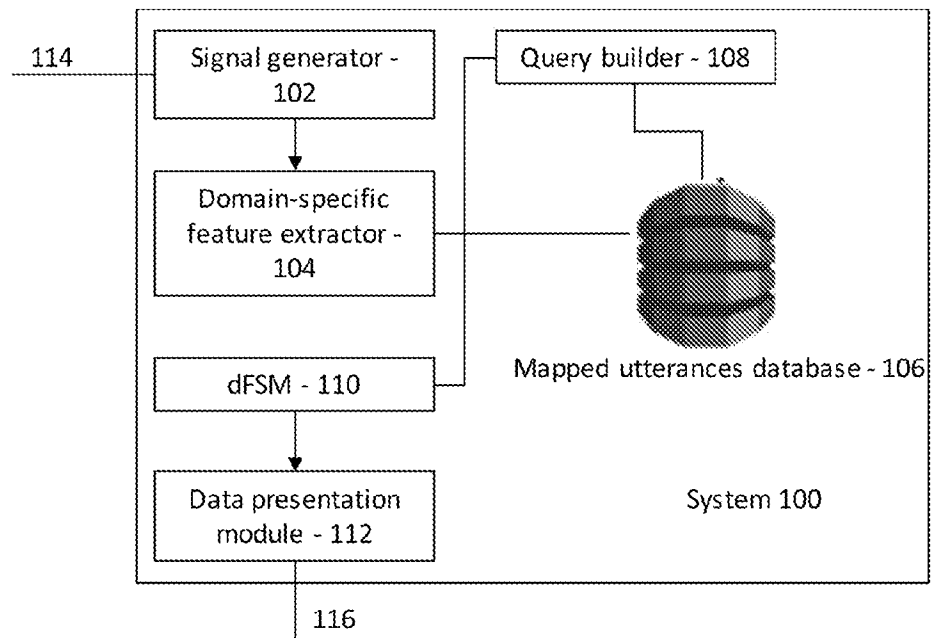
14. The method of any one of claims 11 to 13, wherein building the query comprises:

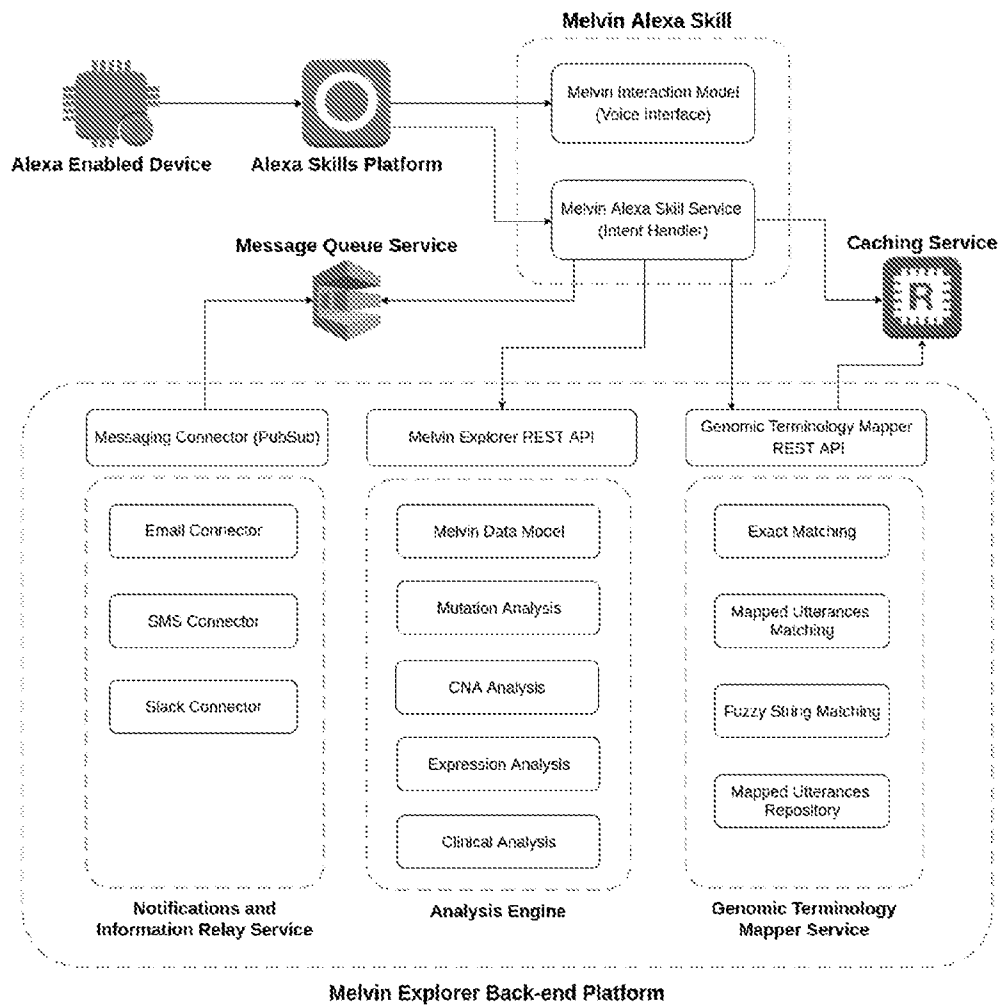
determining, using the query builder, a change of topic based on the further session attributes; and

building the query based only on the further session attributes.

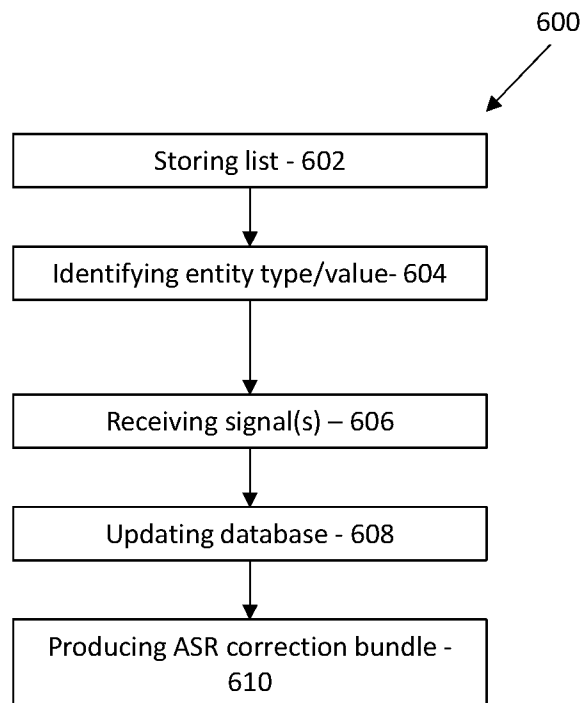
15. The method of any one of claims 11 to 14, wherein building the query comprises:
- determining, using the query builder, a change of context based on the further session attributes; and
 - building the query based on a proper subset of the initial session attributes.
16. The method of any one of claims 10 to 15, further comprises:
- extracting, at the signal generator, one or more domain non-specific features of the verbal utterance; and
 - generating the signal based on the domain non-specific features.
17. The method of any one of claims 10 to 16, wherein extracting data from the database comprises:
- determining, based on the unique state, at least one database, from a plurality of databases, comprising the data; and
 - extracting the data from the at least one database.
18. The method of claim 17, wherein extracting the data from the database comprises mapping at least one of the unique state and query to a query interpretable by each database of the at least one database.
19. A method for recognising out-of-vocabulary terms, comprising:
- storing a list comprising one or more out-of-vocabulary (OOV) terms in a mapped utterances database;
 - identifying, from the list, at least one entity type and entity value for each OOV term;
 - receiving a signal from each of one or more remote terminals, each signal comprising one or more speech recordings, each recording corresponding to a known one of said OOV terms;
 - updating the mapped utterances database based on the one or more speech recordings; and
 - producing an automatic speech recognition (ASR) correction bundle based on the updated mapped utterances database.

20. The method of claim 19, further comprising transforming an output of an ASR model based on the ASR correction bundle.

**Figure 1**

**Figure 2**

3/13

**Figure 3**

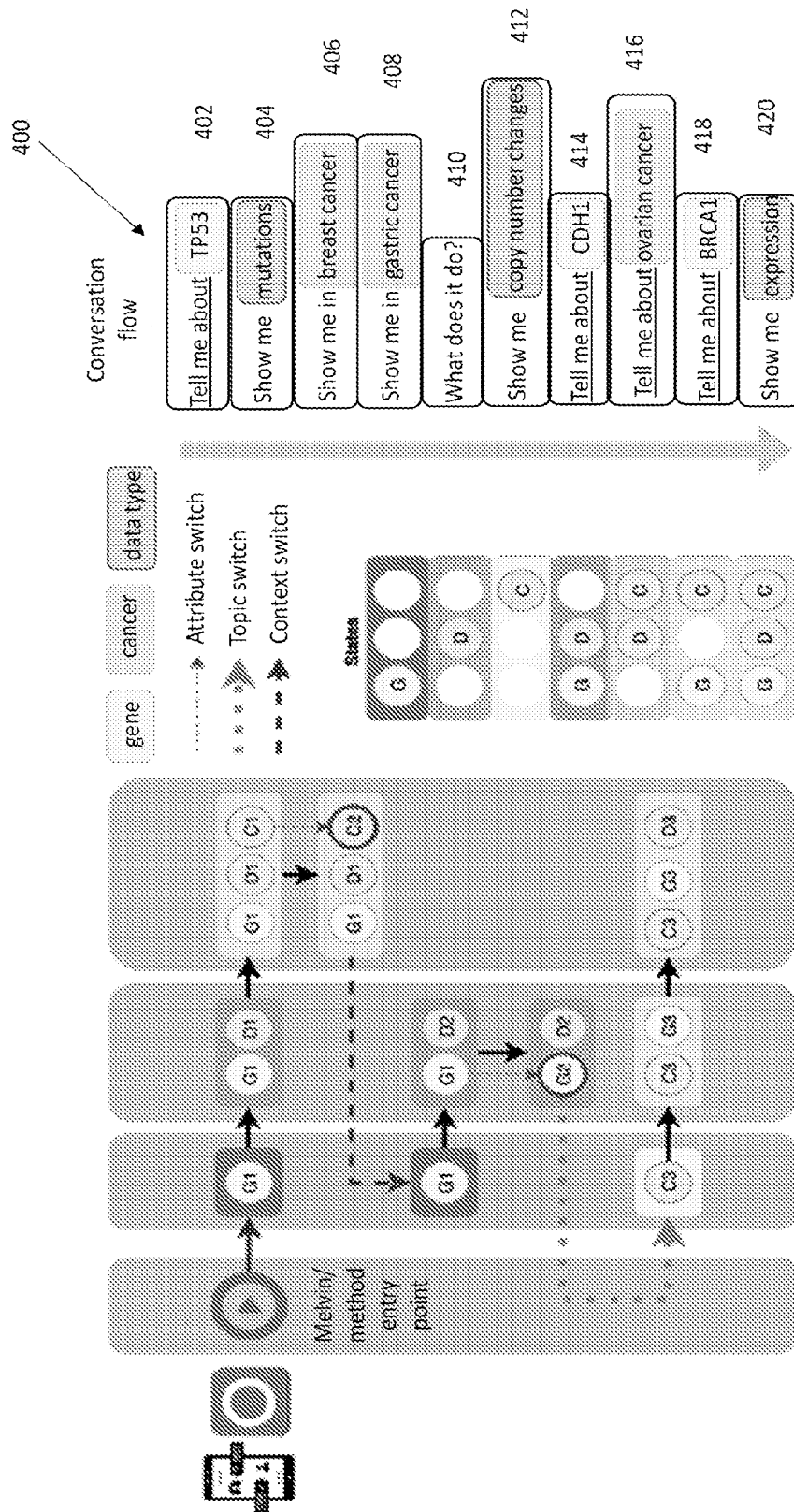


Figure 4

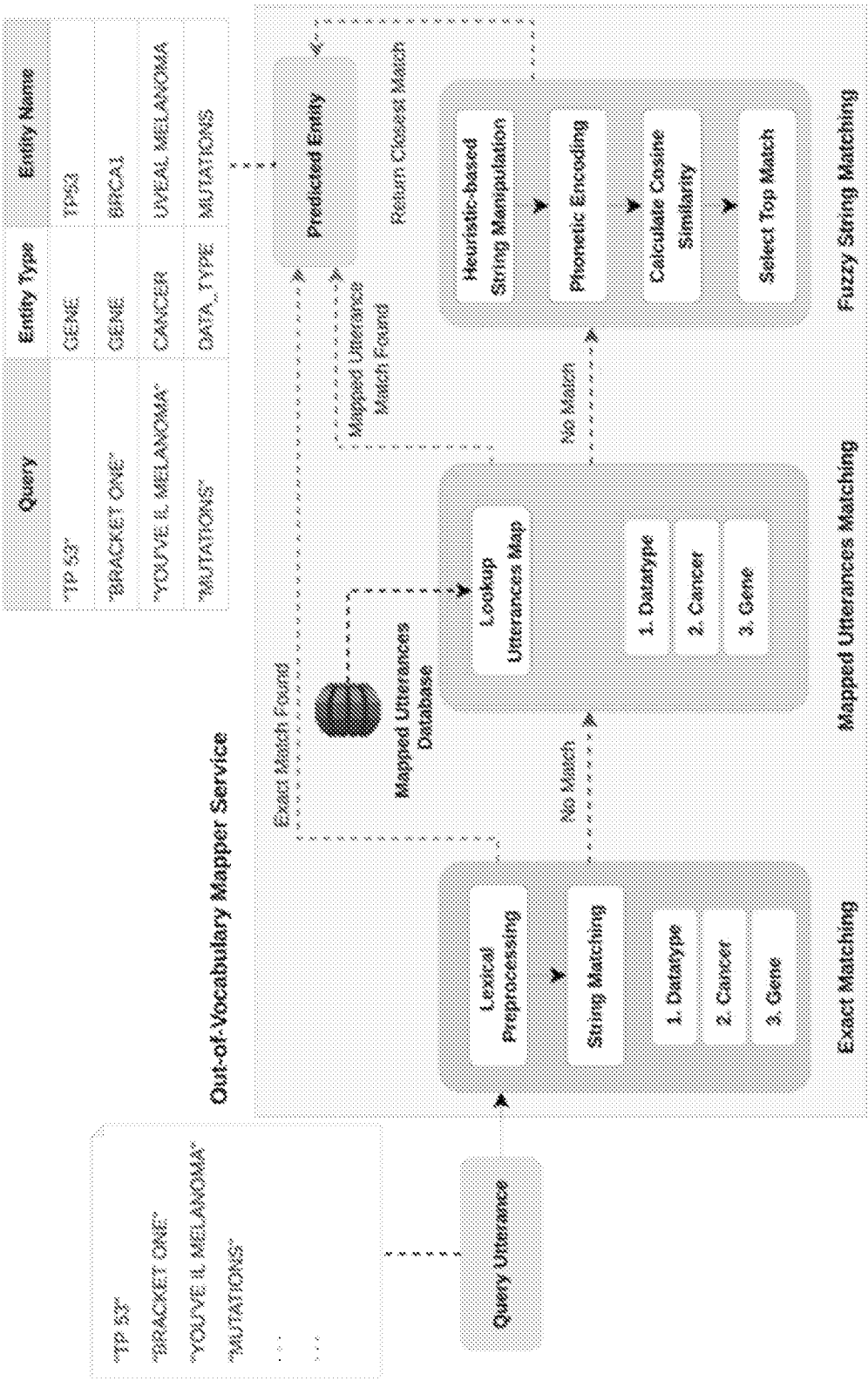
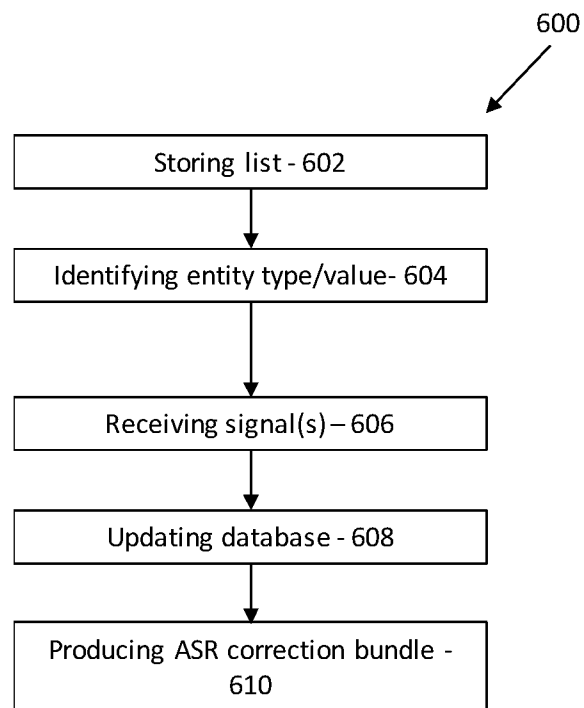


Figure 5

6/13

**Figure 6**

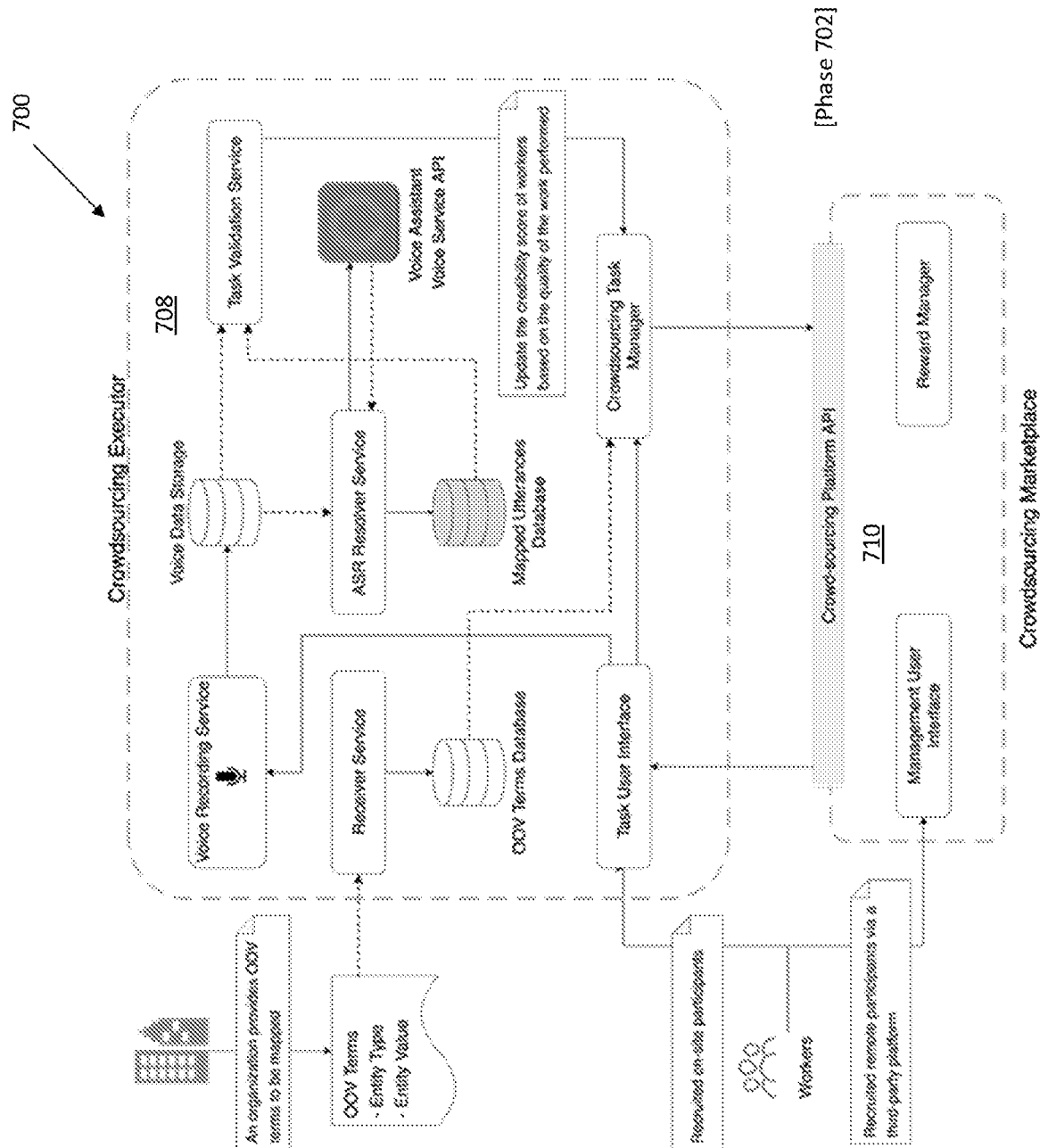


Figure 7

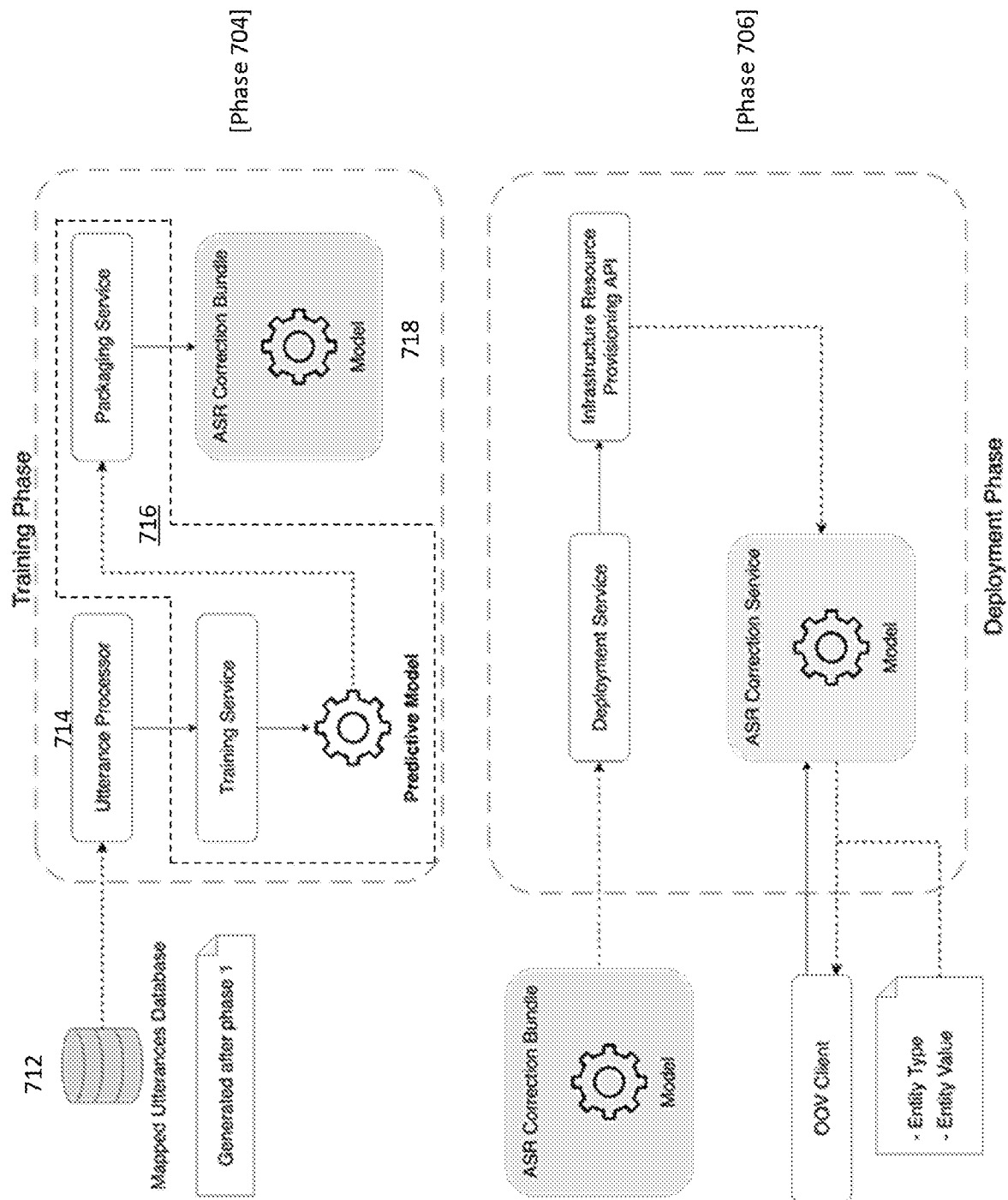


Figure 8

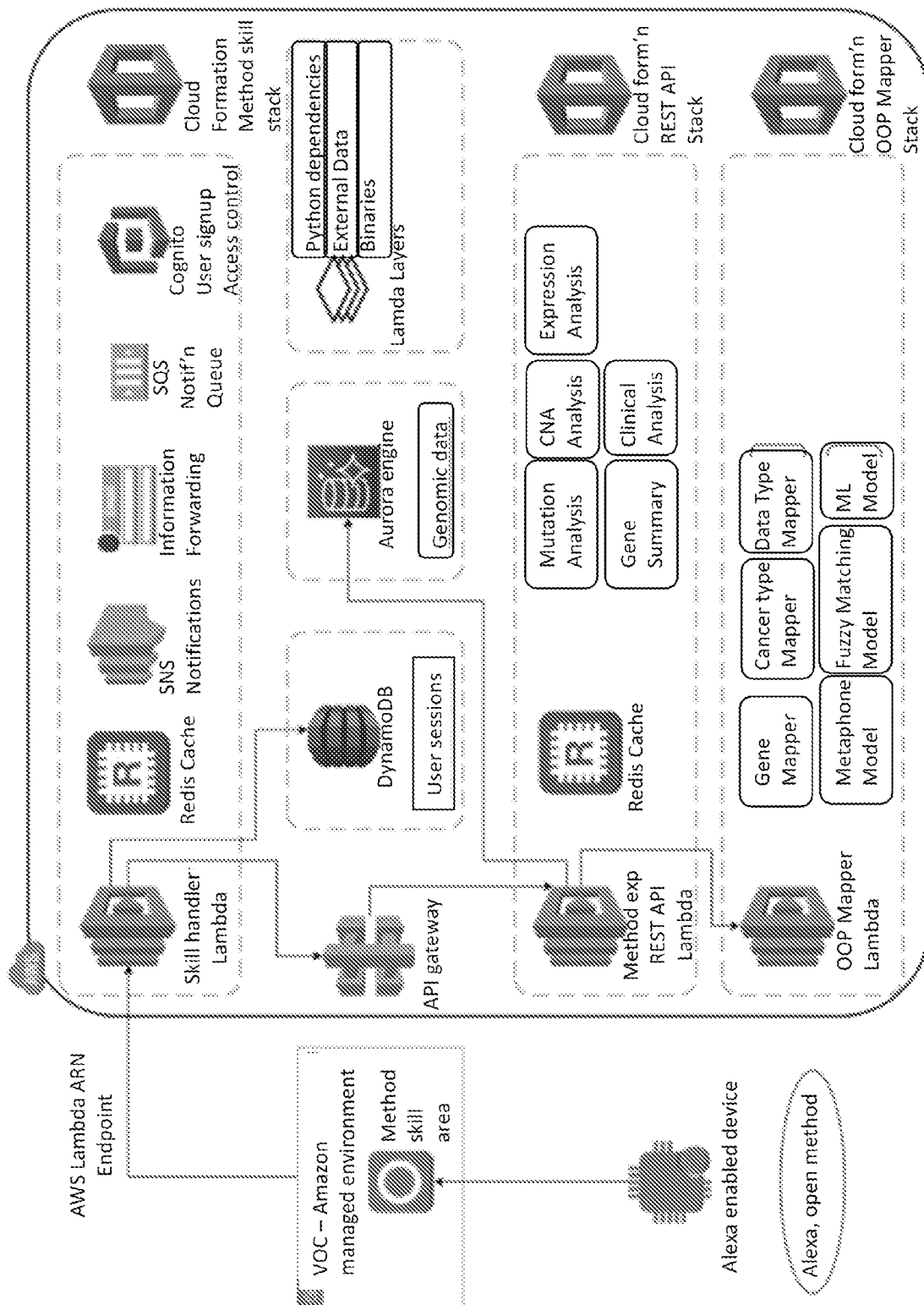


Figure 9

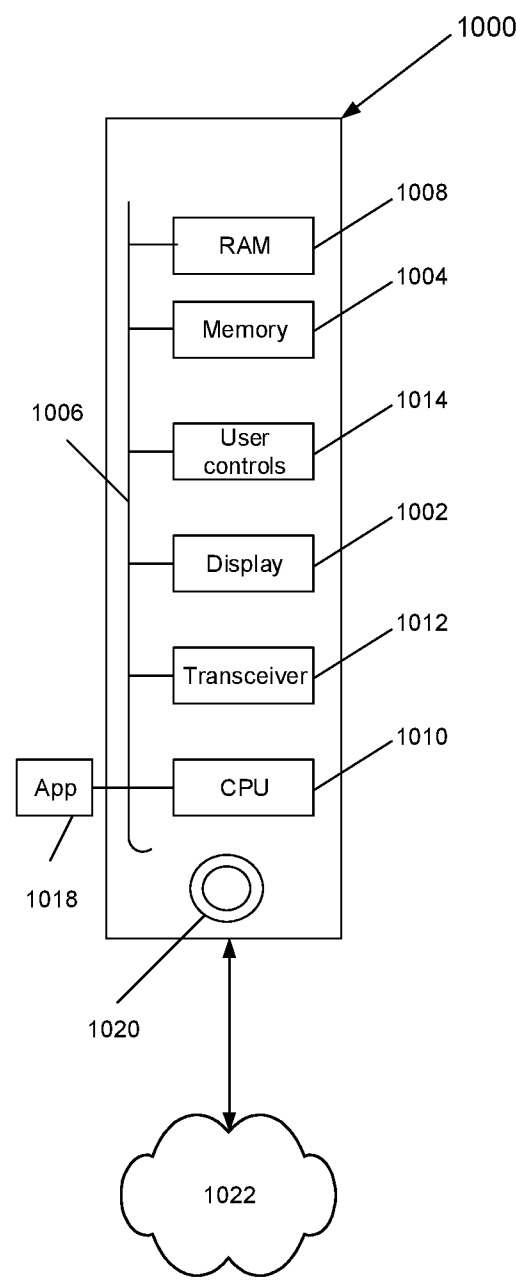
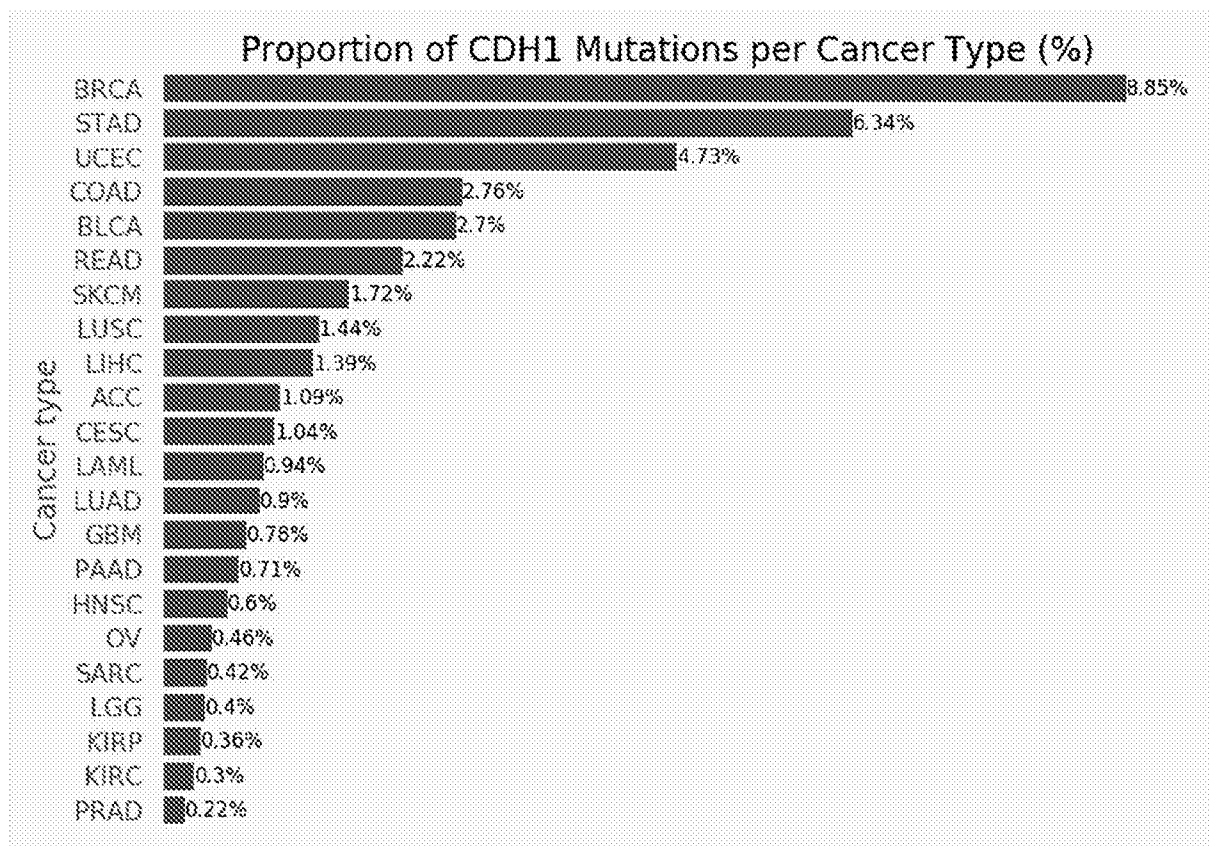
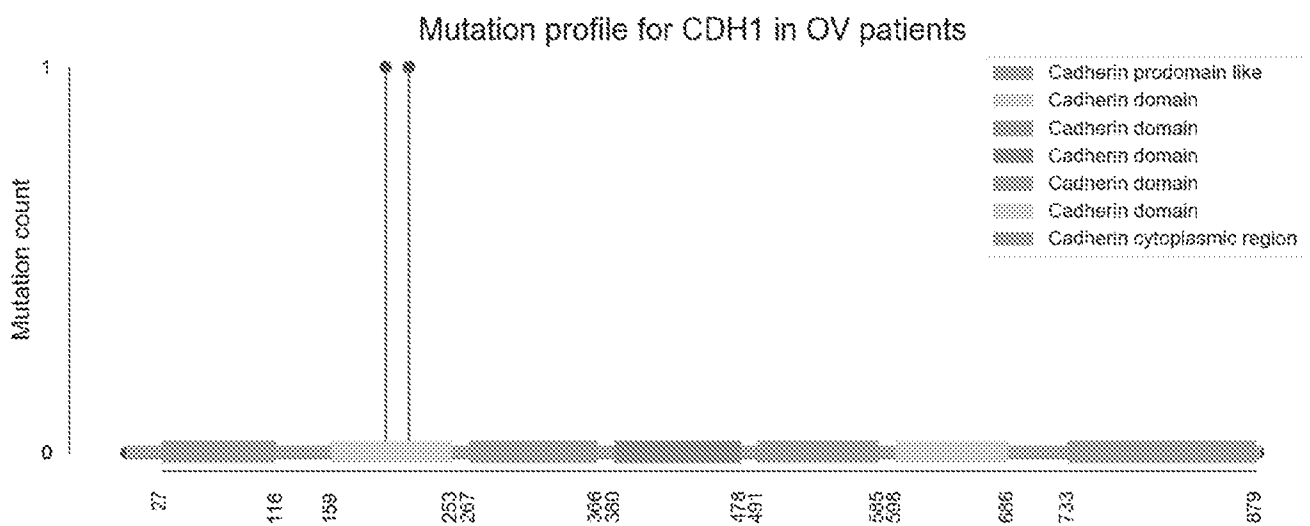


Figure 10

11/13

**Figure 11****Figure 12**

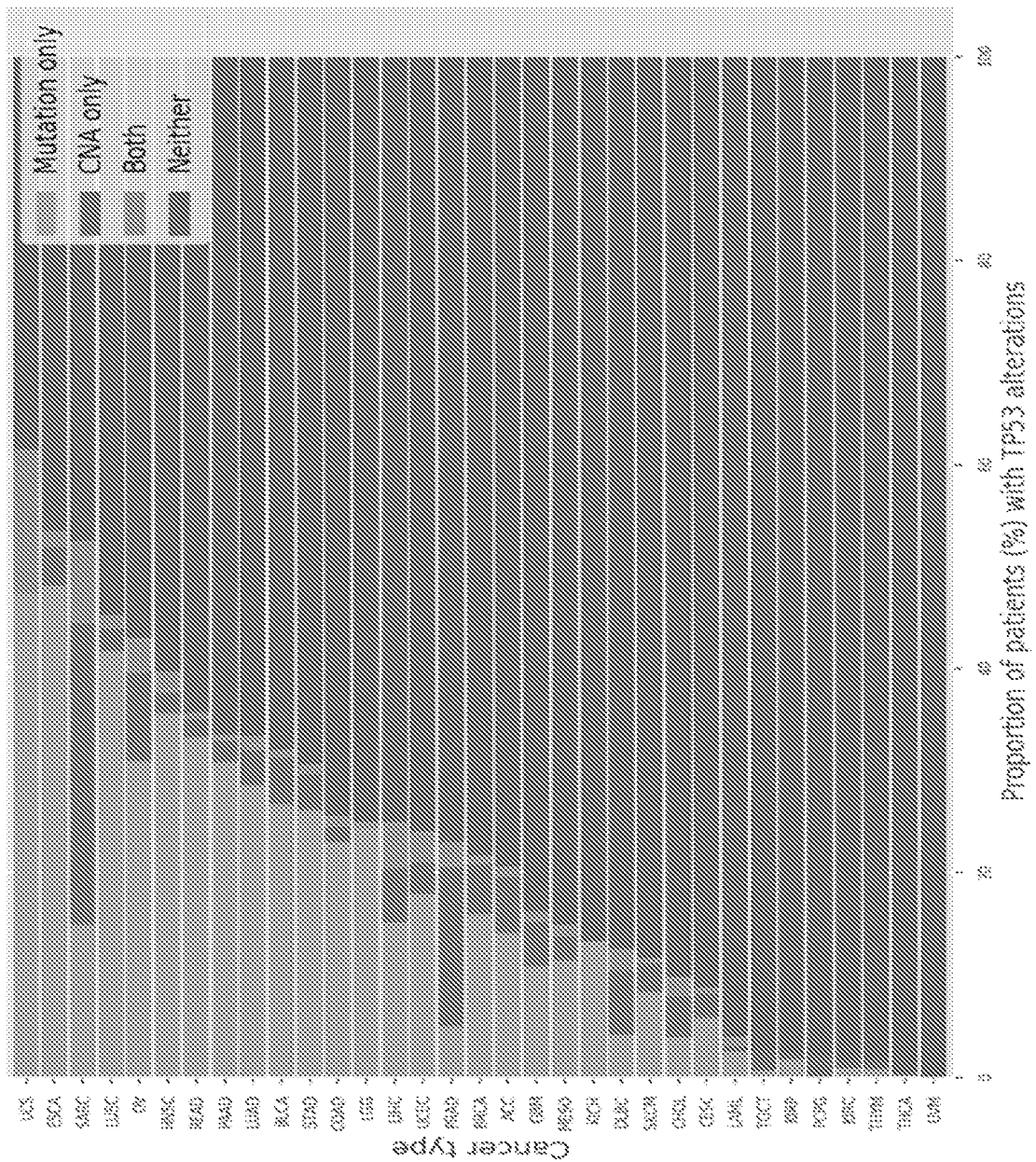


Figure 13

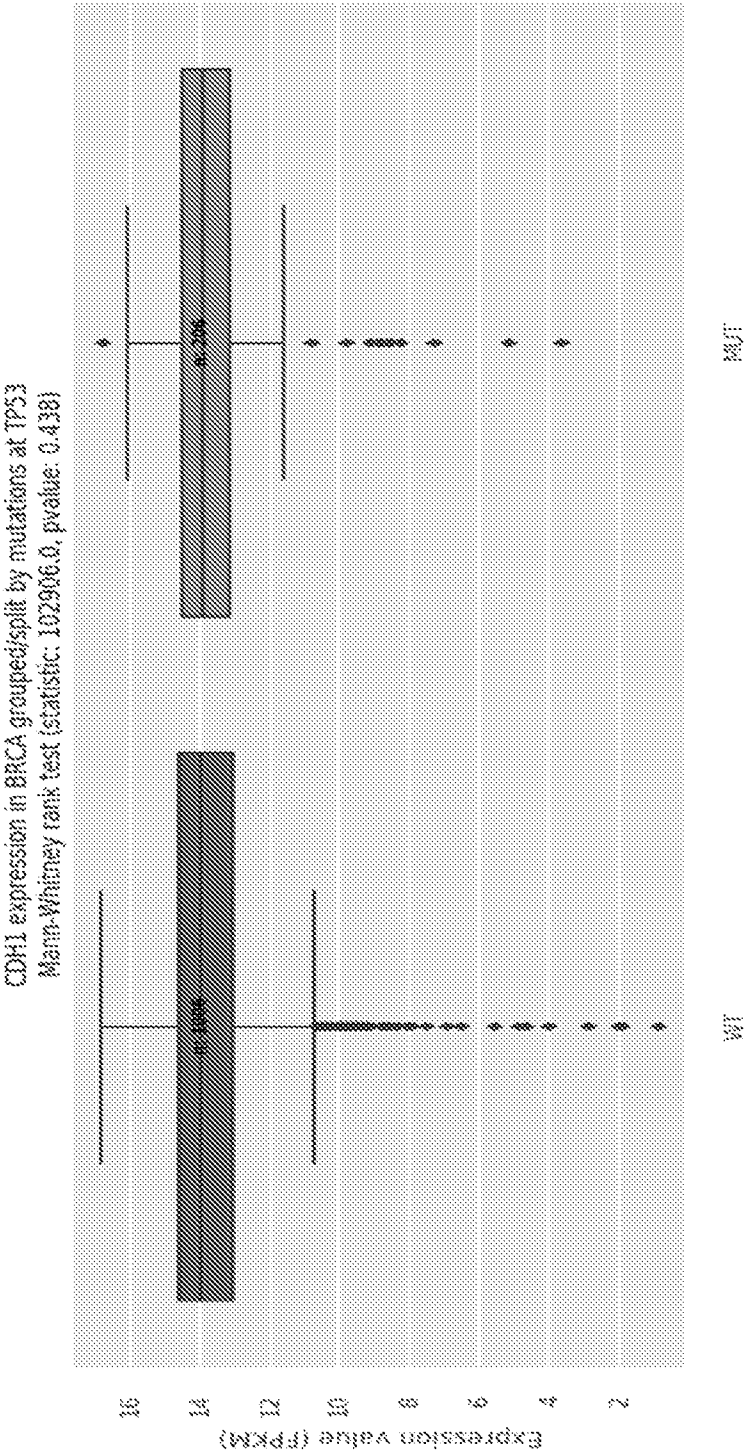


Figure 14

INTERNATIONAL SEARCH REPORT

International application No.

PCT/SG2021/050121

A. CLASSIFICATION OF SUBJECT MATTER

G10L 15/02 (2006.01) G10L 15/18 (2013.01) G06F 16/242 (2019.01)

According to International Patent Classification (IPC)

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G10L; G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Database: FamPat

Search terms: domain, context, area, concept, feature, utterance, speech, voice, map, recognize, extract, map, out of vocabulary, finite, deterministic, unique, state, machine, automata, transducer, 领域, 语境, 景况, 前后关系, 内容, 上下文, 属性, 关键词, 词语, 概念, 段, 子集, 子类, 等级, 最初的, 发声, 发音, 有限状态机, 非词汇, 非词语, 不是 词汇, 不是词语 and other related terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 10032451 B1 (MAMKINA N. V. ET AL.) 24 July 2018 Column 3, Line 56-Column 16, Line 62; Figures 1-3	1-18
X	US 2016/0379626 A1 (DEISHER M. ET AL.) 29 December 2016 Paragraphs [0014], [0018], [0024], [0036], [0037], [0043]-[0053], ; Figures 1-5	19, 20
A	US 6073098 A (BUCHSBAUM A. L. ET AL.) 6 June 2000 Whole document	
A	US 2018/0374484 A1 (HUANG R. ET AL.) 27 December 2018 Whole document	
A	CN 109036391 A (HUAWEI TECH CO LTD) 18 December 2018 Whole document of the machine translation	

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

*Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

02/06/2021

(day/month/year)

Date of mailing of the international search report

08/06/2021

(day/month/year)

Name and mailing address of the ISA/SG

Intellectual Property Office of Singapore
1 Paya Lebar Link, #11-03
PLQ 1, Paya Lebar Quarter
Singapore 408533

Email: pct@ipos.gov.sg

Authorized officer

Nae Win Aung (Mr)

IPOS Customer Service Tel. No.: (+65) 6339 8616

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/SG2021/050121

Note: This Annex lists known patent family members relating to the patent documents cited in this International Search Report. This Authority is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 10032451 B1	24/07/2018	US 10755709 B1 US 2020/0380987 A1	25/08/2020 03/12/2020
US 2016/0379626 A1	29/12/2016	CN 107660303 A WO 2016/209444 A1	02/02/2018 29/12/2016
US 6073098 A	06/06/2000	US 6266634 B1	24/07/2001
US 2018/0374484 A1	27/12/2018	US 2017/0162203 A1	08/06/2017
CN 109036391 A	18/12/2018	WO 2020/001458 A1	02/01/2020