(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 December 2015 (23.12.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/195308 A1

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 17/27* (2006.01)
G10L 15/193 (2013.01)

(21) International Application Number:
PCT/US2015/033481

(22) International Filing Date:
1 June 2015 (01.06.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/014,433 19 June 2014 (19.06.2014) US

(71) Applicant: **THOMSON LICENSING** [FR/FR]; 1-5 rue Jeanne d'Arc, F-92130 Issy-les-Moulineaux (FR).

(72) Inventors: **CLEVENGER, Brian Duane**; 4105 Squire Court, Muncie, Indiana 47304 (US). **NEWBERRY, Thomas P.**; 19145 Tomlinson Road, Westfield, Indiana 46260 (US).

(74) Agents: **SHEDD, Robert D.** et al.; 4 Research Way, 3rd Floor, Princeton, New Jersey 08540 (US).

(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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

(54) Title: SYSTEM FOR NATURAL LANGUAGE PROCESSING

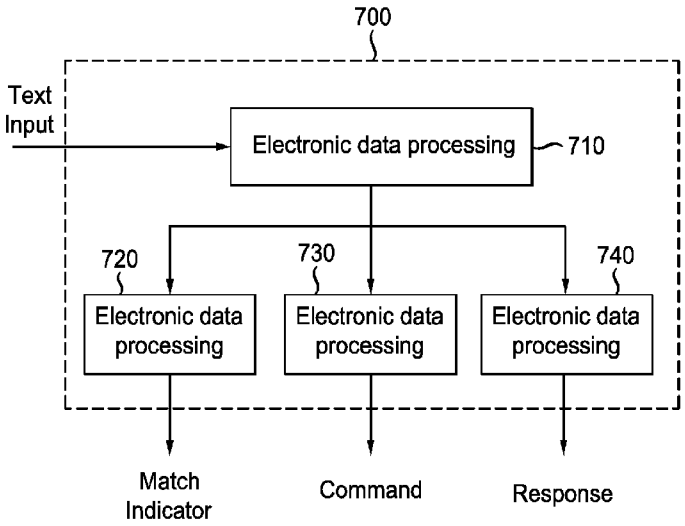

FIG. 7

(57) Abstract: A system (700) that processes natural language text determines whether a match is made to one of a plurality of stored text strings (710), or text string portions. In response to a match having been made (720), the system outputs a command signal, as well as a match indicator signal (730). In addition, a response is optionally sent back to the user to inform him of a status of a device, configuration information, or status of the command (740). The system can operate hierarchically, so that portions of an input text string can be processed sequentially. This feature advantageously enables a reduction in the number of pattern elements that must be searched because, if a top-level pattern does not have a match, then its sub-patterns do not need to be evaluated.

WO 2015/195308 A1

SYSTEM FOR NATURAL LANGUAGE PROCESSING

TECHNICAL FIELD

The present principles relate generally to methods and apparatus for natural language control of an embedded system.

BACKGROUND

Many natural language processing systems exist, but they typically aren't well suited for running on embedded systems. One such example is an AIML (Artificial Intelligence Markup Language) engine. AIML is a standardized markup language used for defining an artificial intelligence (AI) personality. While there are minor implementation differences, all AIML engines are bound by the AIML specification so they work in a similar way. AIML engines use recursive pattern substitution to process text and formulate responses. There are several functional similarities among AIML engines.

First, AIML engines permit very simple string matching and substitution expressions. This reduces the CPU resources required to match across a large number of expressions, which is particularly important because a single command might require multiple recursive passes to process. However, the problem with this approach is that a very large number of expressions are required to implement an AI personality which means lots of memory is required. This makes AIML impractical on an embedded system with limited available memory.

Second, AIML evaluates commands against a flat list of expressions grouped into a few priorities. Because of the simple expression syntax, it's possible for engines to perform optimizations and quickly eliminate many non-wildcard expressions that won't match by ordering them and doing a binary search. This isn't possible with more complex expressions, however.

Third, AIML is really designed for mimicking human interaction, but isn't really designed for performing actions based on the interaction like configuring a device or incorporating external data sources in processing responses, for example, using status and statistics from a modem.

Fourth, AIML provides only limited context information that can be used to alter behavior based on previous interactions. This makes it difficult in practice to

design AI scenarios to walk a user through a specific set of steps – for example, a set of steps to troubleshoot a particular problem.

SUMMARY

These and other drawbacks and disadvantages of the prior art are addressed by the present principles, which are directed to a system for processing natural language text from a user to perform actions and formulate text responses.

According to an aspect of the present principles, there is provided an apparatus for text processing, comprising a natural language electronic text processor having a plurality of electronic data processing stages. The apparatus further comprises a first one of the processing stages receiving a natural language text string and recursively comparing the natural language text string with at least one stored text string until either no more stored text strings remain for comparison, or until the comparison is true. The apparatus further comprises a second one of the processing stages generating a status signal indicative of the comparison being true, a third one of the processing stages executing a command in response to the natural language text if the comparison is true, and a fourth one of the processing stages generating an output message.

According to another aspect of the present principles, there is provided a method for processing text. The method comprises receiving natural language text, and recursively comparing the natural language text with at least one stored text string that is not natural language text until either no more strings remain for comparison, or until the comparison is true. The method further comprises generating a status signal indicative of the comparison, generating a command in response to the text, and generating an output message if the comparison is true.

These and other aspects, features and advantages of the present principles will become apparent from the following detailed description of exemplary embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The present principles may be better understood in accordance with the following exemplary figures, in which:

Figure 1 shows an exemplary pattern element under the present principles.

Figure 2 shows a block diagram of a pattern group element under the present principles.

Figure 3 shows a relationship between a pattern group element and a pattern element under the present principles.

Figure 4 shows a block diagram of an Artificial Intelligence Context under the present principles.

Figure 5 shows a flow diagram of the operation of an Artificial Intelligence engine.

Figures 6 and 6a show a flow diagram of the operation of a Pattern Group Element of Figure 2, and the Pattern Group Element operating with audio commands, under the present principles.

Figure 7 shows one embodiment of an apparatus under the present principles.

Figure 8 shows one embodiment of a method under the present principles.

DETAILED DESCRIPTION

The present principles are directed to a system for processing natural language text from a user, interpreting and performing actions based on that text, and formulating appropriate text responses. The system mimics artificial intelligence and allows a user to interact with a system as if it were a person and not a machine.

Quite often, both technical and non-technical users of complex devices search for a function through an interface to configure a particular feature. It can be difficult, particularly for non-technical users, to find the proper function that they want. For example, a user may wish to disable a firewall, but does not know where the interface for this feature is located. It would be preferable, instead of navigating to a firewall settings page and clicking a disable button, for a user to simply type "Please turn off my firewall" or "Disable the firewall". The system would perform the requested action and respond appropriately with something like "Ok. I turned off the firewall."

As mentioned, natural language processing systems exist, but aren't well suited for running on embedded systems with complex, natural language expressions.

Rather than being confined to very simple expressions, the present principles use a different approach, defining a more flexible expression that can be

considerably more complicated. This allows an AI personality to be defined using far fewer expressions and thus use substantially less memory, but may require more CPU processing. To solve this problem, this invention uses something called a pattern group. The mechanics of a pattern group are described below.

The architecture defined by the present principles makes it easy to both control devices and use information from external data sources in formulating responses. This is one advantage of the present principles over AIML engines. Another advantage is the ability to include context information to alter behavior based on previous interactions, such as walking a user through a series of steps, for example, troubleshooting. This is solved in the described system using something called a pattern group in conjunction with a pattern stack and context stack. This is fully described in a later section.

The present principles are directed to a system for processing natural language text from a user, interpreting and performing actions based on that text, and formulating appropriate text responses.

The system operates by recursively evaluating a user's command against a set of pattern elements. Figure 1 shows an exemplary pattern element. Each pattern element takes a command as an input and provides a match indicator and a command as outputs. A pattern element can optionally provide a response as an output. A pattern element can also read and write data from external sources. For example, a pattern element can read or write to an AI context that stores context information from previously processed commands. In a gateway device, a pattern element may need to write a new configuration value to perform an action specified by a user or read status information to formulate a response. It is expected that different pattern elements will need to access different external data sources based on the actions the specific pattern element needs to perform. Some pattern elements may not require any external data interaction.

When a pattern element processes an input command, it can set a response, modify the input command, or take no action. If the pattern element takes no action, the output command must be equal to the input command and the pattern element must set the match indicator to false. Otherwise, the output match indicator must be set to true. The AI engine recursively evaluates the commands against pattern elements until either a response is set or until no pattern sets the match indicator.

The purpose of pattern elements is to break down the processing of text into common elements that can be used in processing many different types of commands. This is probably best described with some examples.

Regular expressions are sequences of characters in computer science theory that form a search pattern, mostly for pattern matching of strings. A pattern element can be used to expand common contractions by implementing the regular expression s/\<(who|what|where|when|why|how|he|she|it)'s\>/\1 is/. This expression would convert "he's" to "he is" and "what's" to "what is". Another pattern expression might convert common negative responses to "no", such as s/\<(nope|nah|definitely not)\>/no/. These are just simple examples, not complete implementations, but it shows how individual patterns can be used to perform common substitutions until eventually a pattern can be matched against an expression that is able to provide a response output to the user.

In many cases it is useful to chain a top level pattern to a set of sub-patterns. As an example, if implementing patterns to provide the user with definitions for certain terms, it can be useful to have the top-level pattern s/^what does ([a-z]+) (mean|do).*/\1/. This would translate "What does a firewall do?" to "a firewall" or "What does NAT mean?" to "NAT". These could then be passed to a set of sub patterns that match a keyword and provide a definition for that keyword. However, if none of the sub-expressions match, the top-level pattern can revert the output command to the original input and treat the entire set of expressions as a no-match. This functionality is implemented by a special type of pattern element called a pattern group.

Figure 2 shows a block diagram of a pattern group element. Notice that the inputs and outputs of a pattern group element are identical to the pattern element described in Figure 1. This means a pattern group element is a type of pattern element. This relationship is depicted in the UML diagram in Figure 3.

As shown in Figure 2, a pattern group element contains a top level pattern element and one or more sub-pattern elements. The contained pattern elements can be any type of pattern element. So it is possible for a pattern group element to contain other pattern group elements. This allows a hierarchy to be defined where a set of patterns are only evaluated conditionally based on the top level pattern matching. Allowing hierarchies like this to be defined greatly reduces the number of pattern elements that must be evaluated during each pass. So, if a top-level pattern

in a pattern group does not match, none of the patterns in the sub-pattern list will be evaluated. In cases where patterns need to be grouped together without any top-level conditional criteria, a simple top-level pattern that does nothing but set its match indicator to true can be used.

Figure 4 shows a block diagram of the AI Context. The AI Context is used to store information across multiple interactions and also to define the set of patterns to use when evaluating an expression. The AI Context contains a pattern stack. When the AI engine processes input received from a user, it sends the input only to the top pattern on the pattern stack. This will typically be a pattern group element that contains a hierarchy of patterns. It does this repeatedly until either a response is set or until the match indicator is set to false. As patterns are evaluated, they have the ability to push new patterns on the stack or pop them off. This allows pattern elements to modify the set of patterns that the AI Engine will use on the next pass. This can be useful when implementing an interactive troubleshooting scenario. As the user progresses through the troubleshooting scenario, only patterns related to the next troubleshooting step can be included on the top of the stack. When the troubleshooting steps are complete, the modified pattern group can be popped off the stack, restoring the base set of patterns defined by the default pattern group.

In addition to a pattern stack, the AI context provides a Context Stack and a Global Context. The Global Context is simply an associative array of variables that patterns can use to preserve state information between interactions. The Context Stack works in the same way as the Global Context except patterns have the ability to push an entire set of variables onto the stack or pop an entire set of variables off the stack. This is useful when a defined set of variables are only needed for a particular time. Again, the troubleshooting scenario is an example of an embodiment in which this is useful. At the start of the troubleshooting scenario, a new set of variables related to the scenario can be pushed on the context stack. At the end of the scenario, these variables specific to the scenario can be popped off the stack, thus restoring the variables to the original state before the troubleshooting scenario began.

Figure 5 shows one embodiment of a method implementing the AI engine. At 501 a command is received from the user. This command is sent to the top element of the pattern stack for processing at 502. At 503, if the pattern element sets a response, the response is sent to the user at 506 and then processing returns to 501

to retrieve the next command from the user. If the response is not set at 503, then processing moves to 504 to check if the match indicator is set. If the match indicator is set, the command is replaced with the command output from the pattern and processing returns to 502 to process the updated command again. If, at 504, the match indicator is not set, it means that the AI engine was unable to process the command from the user and processing then proceeds to 507 where a default response is sent to the user, such as "I'm sorry. I don't understand", for example. Processing then returns to step 501 to retrieve the next command from the user.

Figure 6 is a flow chart showing the internal operation of the Pattern Group Element described in Figure 2. Processing begins at 601 where a command is received as an input. At 602 the input command is stored in temporary memory as CMD1. At 603, the input command (CMD1) is then sent to the Top Level Pattern Element for processing. At 604, the Top Level Pattern Element is checked to see if the match indicator is set. If the match indicator is not set, then processing moves to 613, the output command is set to the original input command (CMD1), the match indicator is set to false, and processing ends. If, at 604, the match indicator is set, processing moves to 605 and the output command for the top level pattern is stored in temporary memory as CMD2. Processing then moves to 606 to check if there are any more patterns in the Sub-Pattern List. If no more patterns exist, processing then proceeds to 613. Otherwise, processing proceeds to 607 and the next pattern is retrieved from the Sub-Pattern List. At 608 CMD2 is sent to the pattern retrieved at 607 for processing. At 609, if the response is set by the pattern processed at 608, processing proceeds to 611, the output command, match indicator, and response from the pattern are sent as outputs and processing ends. If at 609 the response is not set, processing proceeds to 610 to check if the match indicator is set. If the match indicator is not set at 610, processing proceeds back to 606 to process any remaining patterns in the Sub-Pattern List. If at 610 the match indicator is set, then the command and match indicator from the last pattern are set as outputs and processing ends.

Figure 6a shows the same flow chart of Figure 6, however, in another embodiment, the system can interface to an audio to text conversion device to receive audio commands, and convert them to text. At the output, the response messages can be converted from text to audio. The audio to text converter, as well

as the text to audio converter could be standalone units, or integrated as part of the present system.

FIG. 7 shows a hardware block diagram of an exemplary embedded system 700, in accordance with an embodiment of the present principles. Of course, it is to be appreciated that the present principles are not limited to the embedded system 700 shown and described in FIG. 7 and, thus, other systems 700 having, e.g., different configuration and/or different elements, can also be used in accordance with the teachings of the present principles.

In embedded system 700, input text is received from a user to an input of system 700. The input is in signal connectivity with a first input of electronic data processing stage 710. Processing stage 710 performs a comparison of the text input with a stored text strings. The stored text strings can be in memory (not shown) within system 700, or input from external memory via another input (not shown) of system 700 and sent to processing element 710.

In another embodiment of system 700, processing stage 710 processes a plurality of portions of the input text string one after another. Control logic within processing stage 710 controls the processing of the input text string portions, so that when a match is made, a next portion can be searched against the stored text strings. If a match is not made on any particular portion, processing stage 710 can decide to abort comparisons on additional text string portions, as the input text will not be a valid command. Processing stage 710 has a signal output that is sent to electronic data processing stages 720, 730 and 740..

After processing stage 710 finds a match for an entire text string, or all of the text string portions, it generates signals in response to the match. Processing stage 720 is used to generate the match indicator signal that provides a metric indicative of whether the input text string has matched a stored text string. The match indicator signal can be a single binary signal, indicating true or false, or a multilevel signal indicative of the level of match. For example, if the input text string has been processed in portions as described in the preceding embodiment, a multilevel match indicator signal can provide a measure of the degree to which the entire string has been matched up to that point.

Processing stage 730 is used to generate a command. Commands are used to react to the input text string, such as to control other devices, for example. If processing stage 710 has made a match to the input text string, the command signal

can be used to modify the input text command. If processing stage 710 has not made a match for the input text string, processing stage 730 can simply pass the input text on to another circuit.

Processing stage 740 is used to generate a response to the input text, which is sent to a user. The response can be a text string stating, for example, that a configuration setting has been altered, or giving the status of a device. If no matches are made to the input text string, the response signal can be used to state, "Invalid command", for example.

It is important to note that several components and interconnections necessary for complete operation of system 700 are not shown in the interest of conciseness, as the components not shown are well known to those skilled in the art.

Figure 8 shows an embodiment of a method 800 for processing text input, in accordance with an embodiment of the present principles. The method commences at the start block 801, then proceeds to step 810 for receiving text from a user. Control then proceeds to step 820 for comparing text with stored strings of text. Following step 820, step 830 is used to determine whether there has been a comparison in step 820. If there has been a comparison, control proceeds to steps 850, 860, and 870. If there has not been a comparison, control proceeds to step 840 for determining whether there are more strings to compare to the input text. If there are, control proceeds back to step 820 for comparing text with another stored string. As there continues to be no matching strings to the input text, control will sequence from step 820 to step 830, step 840 and then back to step 820.

If step 830 determines that there is no comparison, a determination is made in step 840 that there are also no more strings, then control proceeds to step 860 for generating a response to a user. In this case, such a response might be, "Invalid command", "Error", or another response, for example.

If, at step 830 a determination is made that a comparison has been made, control proceeds to step 850 for generating a match indicator, step 860 for optionally generating a response to a user, and step 870 for generating a command.

The present description illustrates the present principles. It will thus be appreciated that those skilled in the art will be able to devise various arrangements that, although not explicitly described or shown herein, embody the present principles and are thereby included within the present principles.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the present principles and the concepts contributed by the inventor(s) to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions.

Moreover, all statements herein reciting principles, aspects, and embodiments of the present principles, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

Thus, for example, it will be appreciated by those skilled in the art that the block diagrams presented herein represent conceptual views of illustrative circuitry embodying the present principles. Similarly, it will be appreciated that any flow charts, flow diagrams, state transition diagrams, pseudocode, and the like represent various processes which may be substantially represented in computer readable media and so executed by a computer or processor, whether or not such computer or processor is explicitly shown.

The functions of the various elements shown in the figures may be provided through the use of dedicated hardware as well as hardware capable of executing software in association with appropriate software. When provided by a processor, the functions may be provided by a single dedicated processor, by a single shared processor, or by a plurality of individual processors, some of which may be shared. Moreover, explicit use of the term “processor” or “controller” should not be construed to refer exclusively to hardware capable of executing software, and may implicitly include, without limitation, digital signal processor (“DSP”) hardware, read-only memory (“ROM”) for storing software, random access memory (“RAM”), and non-volatile storage.

Other hardware, conventional and/or custom, may also be included. Similarly, any switches shown in the figures are conceptual only. Their function may be carried out through the operation of program logic, through dedicated logic, through the interaction of program control and dedicated logic, or even manually, the particular technique being selectable by the implementer as more specifically understood from the context.

In the claims hereof, any element expressed as a means for performing a specified function is intended to encompass any way of performing that function including, for example, a) a combination of circuit elements that performs that function or b) software in any form, including, therefore, firmware, microcode or the like, combined with appropriate circuitry for executing that software to perform the function. The present principles as defined by such claims reside in the fact that the functionalities provided by the various recited means are combined and brought together in the manner which the claims call for. It is thus regarded that any means that can provide those functionalities are equivalent to those shown herein.

Reference in the specification to "one embodiment" or "an embodiment" of the present principles, as well as other variations thereof, means that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present principles. Thus, the appearances of the phrase "in one embodiment" or "in an embodiment", as well any other variations, appearing in various places throughout the specification are not necessarily all referring to the same embodiment.

It is to be appreciated that the use of any of the following "/", "and/or", and "at least one of", for example, in the cases of "A/B", "A and/or B" and "at least one of A and B", is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of both options (A and B). As a further example, in the cases of "A, B, and/or C" and "at least one of A, B, and C", such phrasing is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of the third listed option (C) only, or the selection of the first and the second listed options (A and B) only, or the selection of the first and third listed options (A and C) only, or the selection of the second and third listed options (B and C) only, or the selection of all three options (A and B and C). This may be extended, as readily apparent by one of ordinary skill in this and related arts, for as many items listed.

These and other features and advantages of the present principles may be readily ascertained by one of ordinary skill in the pertinent art based on the teachings herein. It is to be understood that the teachings of the present principles may be implemented in various forms of hardware, software, firmware, special purpose processors, or combinations thereof.

Most preferably, the teachings of the present principles are implemented as a combination of hardware and software. Moreover, the software may be implemented as an application program tangibly embodied on a program storage unit. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPU"), a random access memory ("RAM"), and input/output ("I/O") interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit.

It is to be further understood that, because some of the constituent system components and methods depicted in the accompanying drawings are preferably implemented in software, the actual connections between the system components or the process function blocks may differ depending upon the manner in which the present principles are programmed. Given the teachings herein, one of ordinary skill in the pertinent art will be able to contemplate these and similar implementations or configurations of the present principles.

Although the illustrative embodiments have been described herein with reference to the accompanying drawings, it is to be understood that the present principles are not limited to those precise embodiments, and that various changes and modifications may be effected therein by one of ordinary skill in the pertinent art without departing from the scope of the present principles. All such changes and modifications are intended to be included within the scope of the present principles as set forth in the appended claims.

CLAIMS:

1. A natural language electronic text processor having a plurality of electronic data processing stages, comprising:

a first one of said processing stages receiving a natural language text string and recursively comparing said natural language text string with at least one stored text string until either no more stored text strings remain for comparison, or until said comparison is true;

a second one of said processing stages generating a status signal indicative of said comparison being true;

a third one of said processing stages executing a command in response to said natural language text if said comparison is true; and,

a fourth one of said processing stages generating an output message.

2. The natural language electronic text processor of claim 1, wherein:

at least one of said plurality of processing stages converts audible speech to electronic text; and,

at least one of said plurality of processing stages converts electronic text to audible speech.

3. The natural language electronic text processor of claim 2, wherein:

at least one of said processing stages receives said natural language text string in one of audible form and electronic text; and,

at least one of said processing stages generates said output message in one of audible form and electronic text.

4. The natural language processor of claim 3, wherein:

said natural language text string is received in audible format; and,

said output message is generated in audible format.

5. The natural language processor of claim 3, wherein:

said natural language text string is received in non-audible format; and,

said output message is generated in non-audible format.

6. The natural language processor of claim 3, wherein:

said natural language text string is received in one of audible and non-audible format; and,

said output message is generated in one of audible or non-audible format.

7. A method for processing text, comprising:

receiving natural language text;

recursively comparing said natural language text with at least one stored text string that is not natural language text until either no more strings remain for comparison, or until said comparison is true;

generating a status signal indicative of said comparison;

generating a command in response to said text; and,

generating an output message if said comparison is true.

8. The method of Claim 7, wherein:

at least one of said plurality of processing stages converts audible speech to electronic text; and,

at least one of said plurality of processing stages converts electronic text to audible speech.

9. The method of Claim 8, wherein:

at least one of said processing stages receives said natural language text string in one of audible form and electronic text; and,

at least one of said processing stages generates said output message in one of audible form and electronic text.

10. The method of Claim 9, wherein:

said natural language text string is received in audible format; and,

said output message is generated in audible format.

11. The method of Claim 9, wherein:

said natural language text string is received in non-audible format; and,

said output message is generated in non-audible format.

12. The method of Claim 9, wherein:

said natural language text string is received in one of audible and non-audible format; and,

said output message is generated in one of audible or non-audible format.

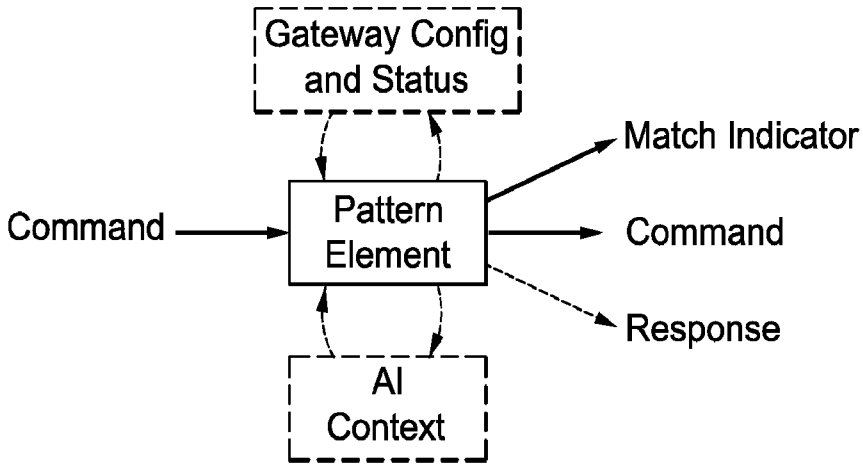
Gateway Config and Status
Match Indicator
Command
Pattern Element
Command
Response
AI Context

FIG. 1

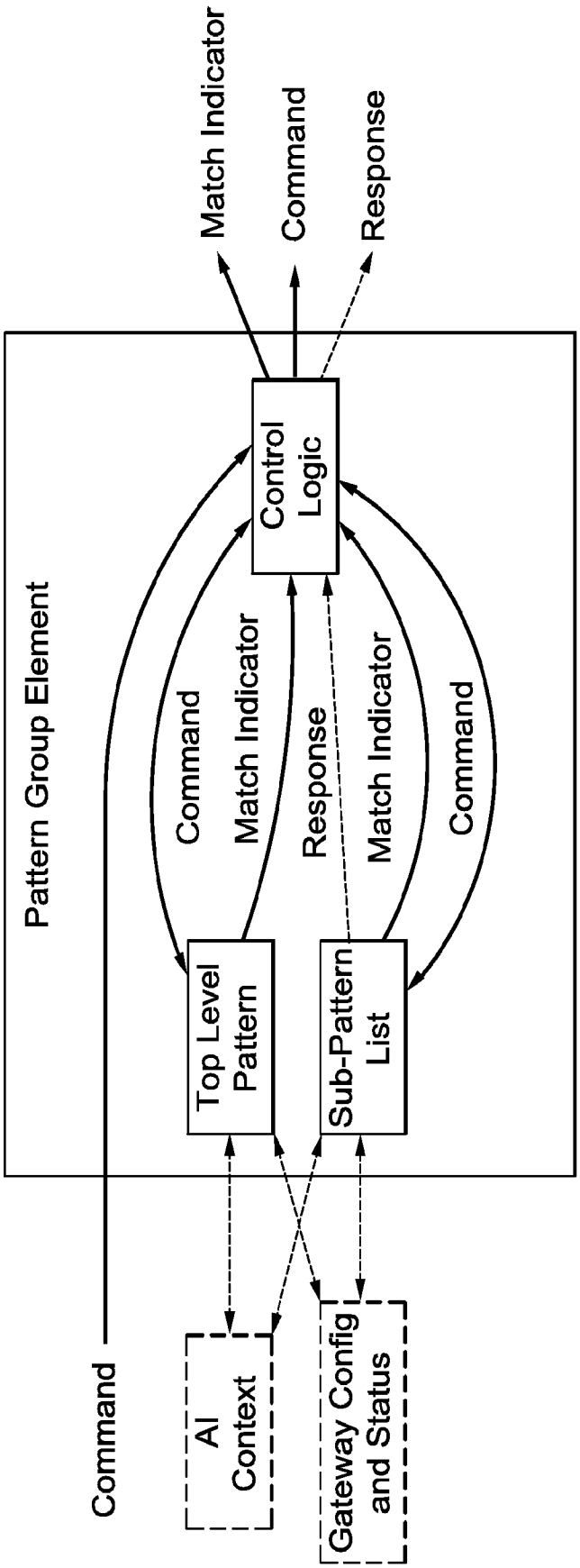
Match Indicator
Command
Response
Pattern Group Element
Control Logic
Command
Match Indicator
Response
Match Indicator
Command
Top Level Pattern
Sub-Pattern List
Command
AI Context
Gateway Config and Status

FIG. 2

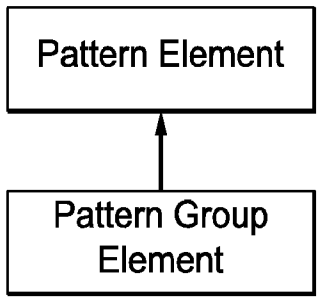
Pattern Element
Pattern Group Element

FIG. 3

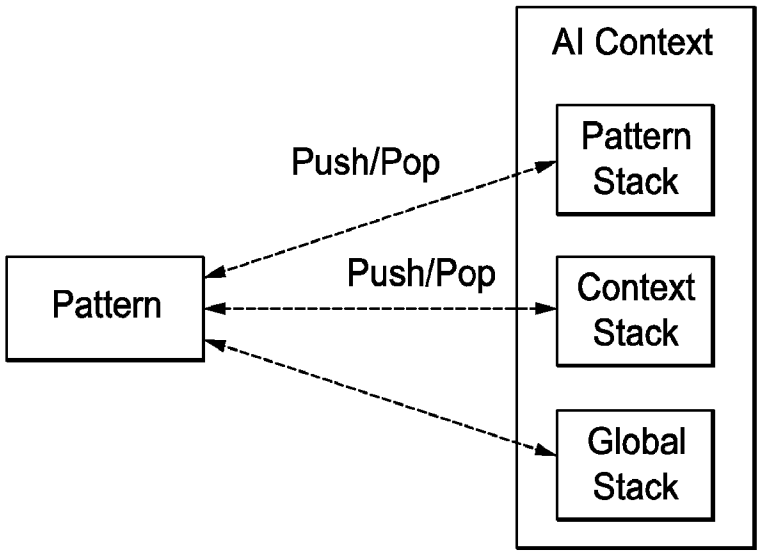
AI Context
Pattern Stack
Context Stack
Global Stack
Push/Pop
Push/Pop
Pattern

FIG. 4

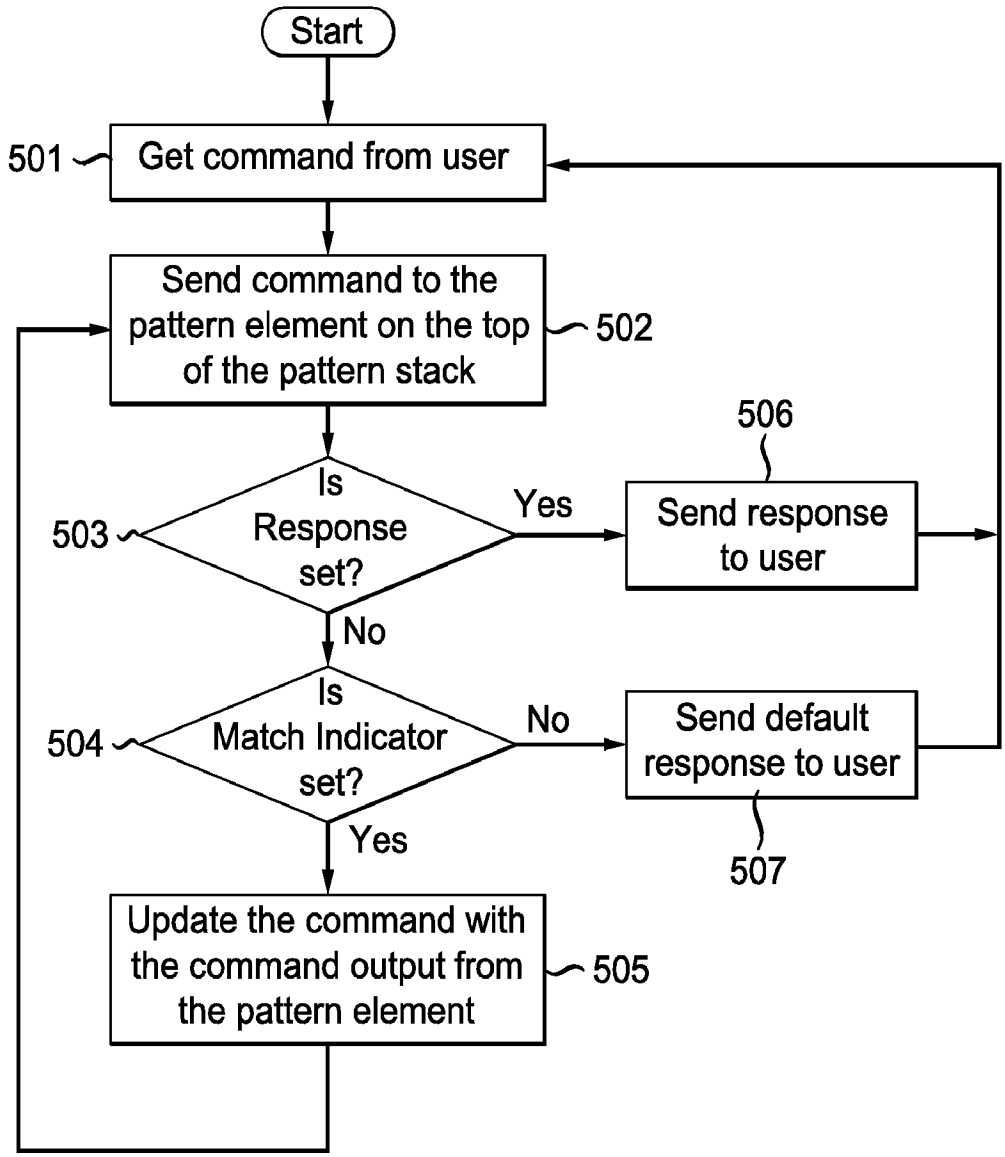
Start
501
Get command from user
Send command to the pattern element on the top of the pattern stack
502
506
503
Is Response set?
Yes
Send response to user
No
504
Is Match Indicator set?
No
Send default response to user
507
Yes
Update the command with the command output from the pattern element
505

FIG. 5

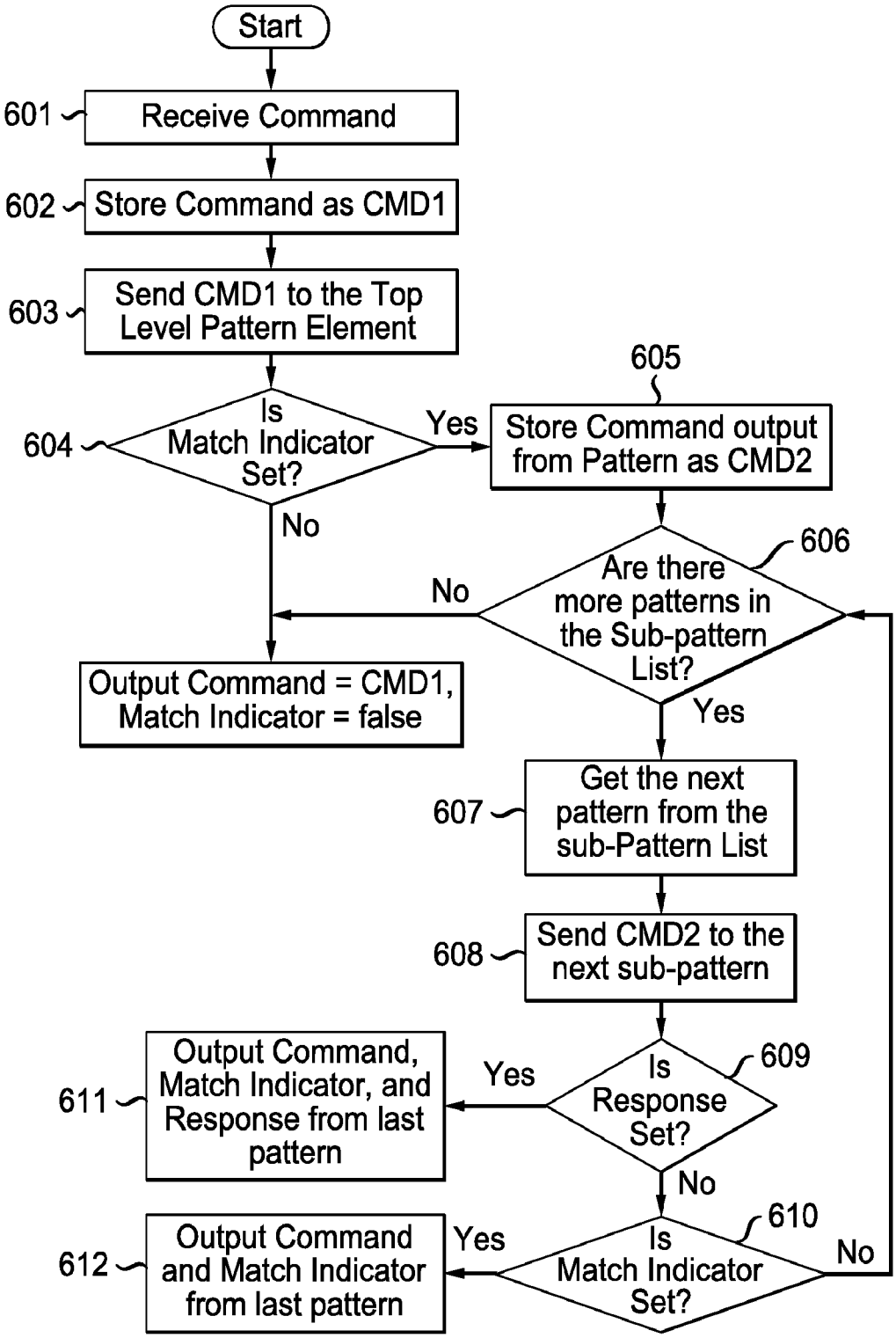
Start
601
Receive Command
602
Store Command as CMD1
603
Send CMD1 to the Top Level Pattern Element
604
Is Match Indicator Set?
Yes
No
605
Store Command output from Pattern as CMD2
606
Are there more patterns in the Sub-pattern List?
No
Yes
Output Command = CMD1, Match Indicator = false
607
Get the next pattern from the sub-Pattern List
608
Send CMD2 to the next sub-pattern
609
Is Response Set?
Yes
No
611
Output Command, Match Indicator, and Response from last pattern
610
Is Match Indicator Set?
Yes
No
612
Output Command and Match Indicator from last pattern

FIG. 6

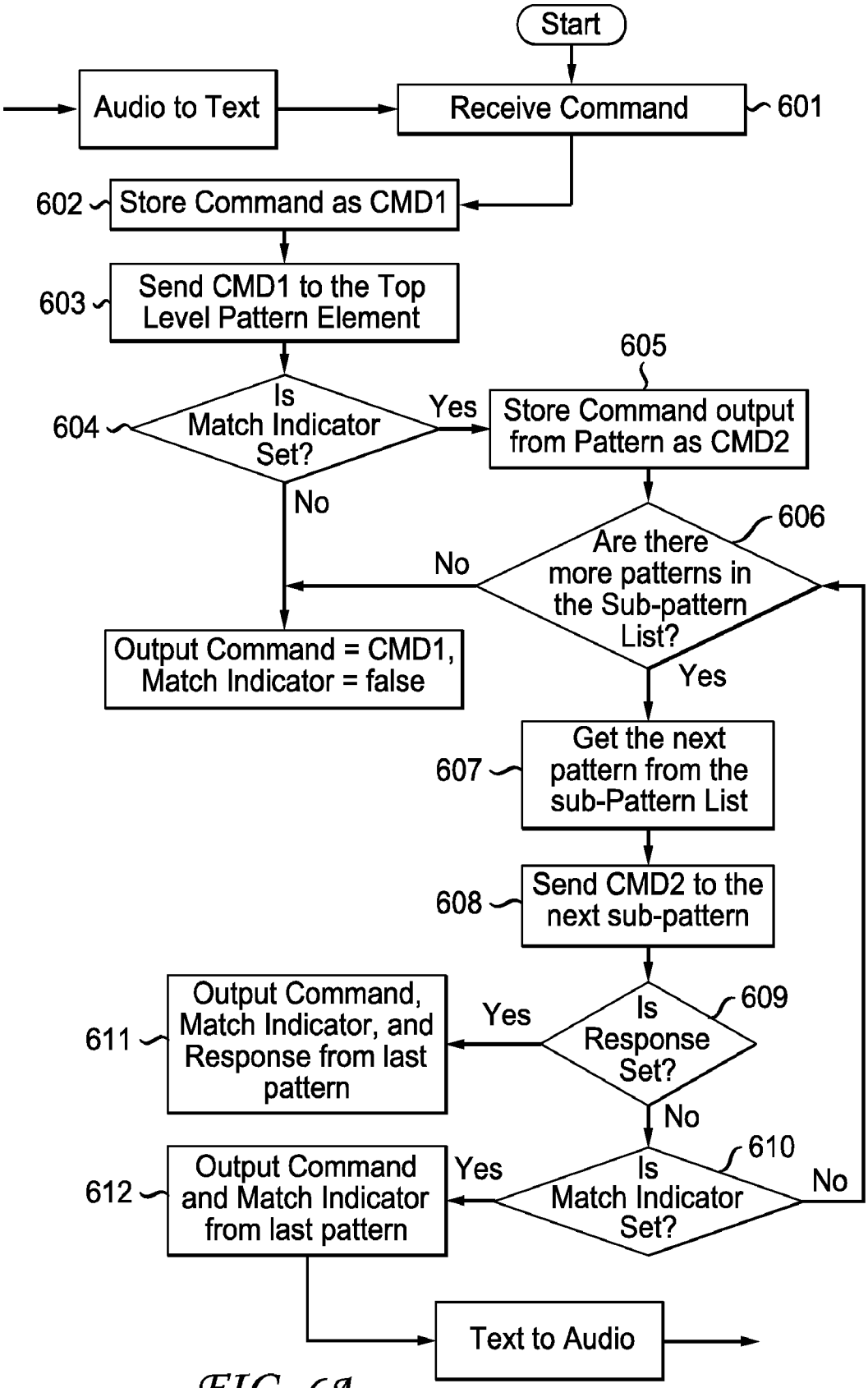
Start
Audio to Text
Receive Command
601
602
Store Command as CMD1
603
Send CMD1 to the Top Level Pattern Element
605
604
Is Match Indicator Set?
Yes
Store Command output from Pattern as CMD2
No
606
Are there more patterns in the Sub-pattern List?
No
Yes
Output Command = CMD1, Match Indicator = false
607
Get the next pattern from the sub-Pattern List
608
Send CMD2 to the next sub-pattern
609
Is Response Set?
Yes
611
Output Command, Match Indicator, and Response from last pattern
No
610
Is Match Indicator Set?
Yes
No
612
Output Command and Match Indicator from last pattern
Text to Audio

FIG. 6A

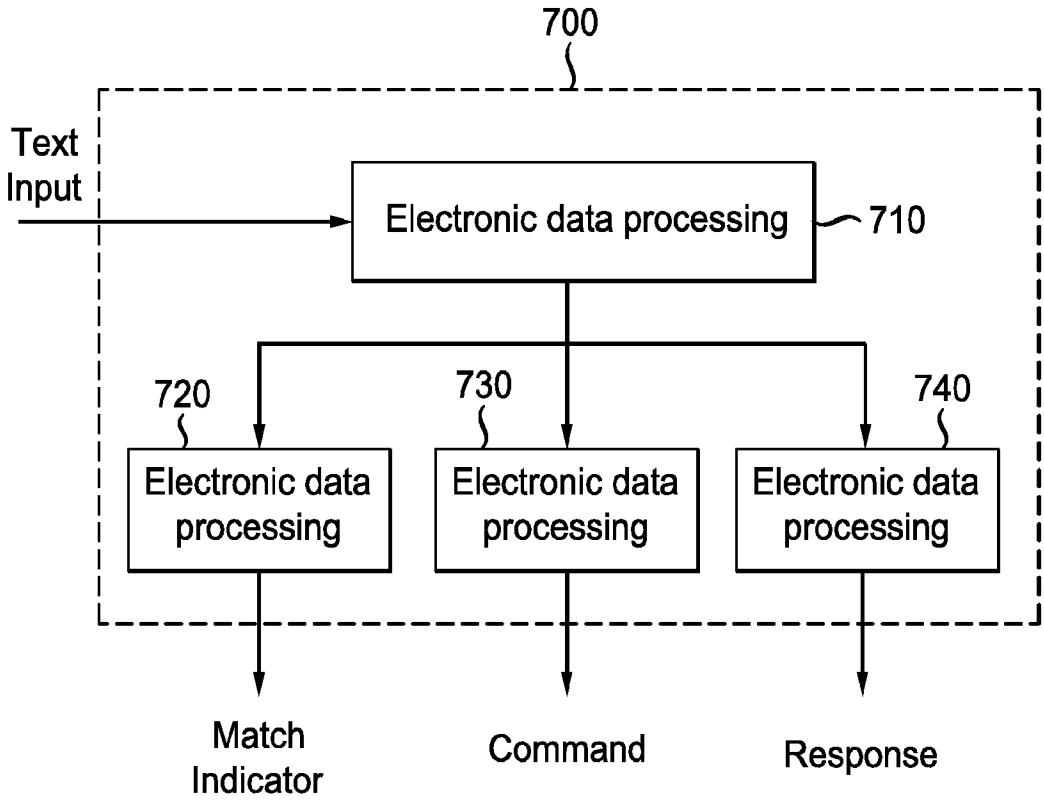
700
Text Input
Electronic data processing
710
720
730
740
Electronic data processing
Electronic data processing
Electronic data processing
Match Indicator
Command
Response

FIG. 7

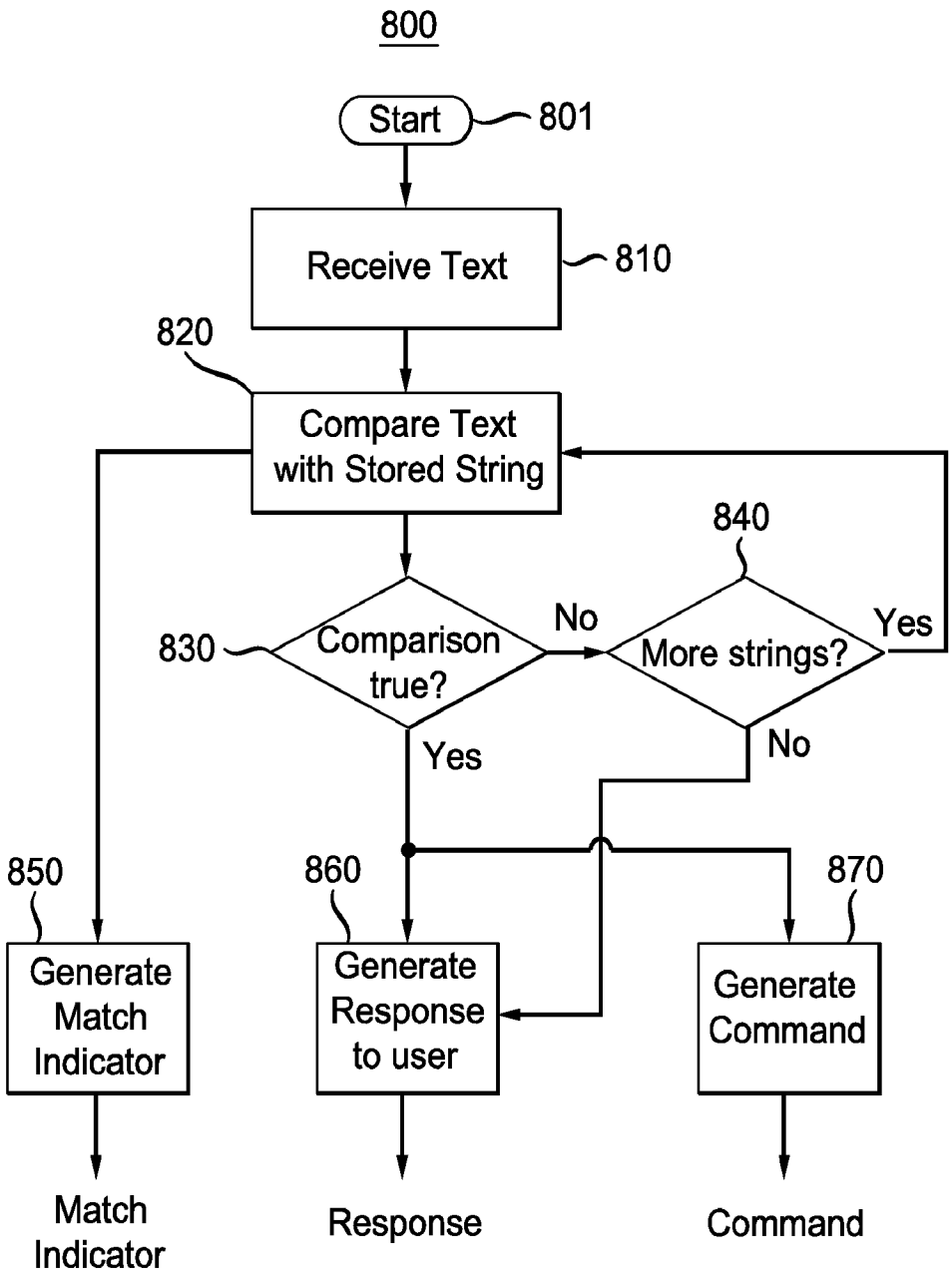
800
Start
801
Receive Text
810
820
Compare Text with Stored String
840
Comparison true?
830
No
More strings?
Yes
Yes
No
850
860
870
Generate Match Indicator
Generate Response to user
Generate Command
Match Indicator
Response
Command

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/033481

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30 G10L15/193 G06F17/27
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F G10L

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2010/332235 A1 (DAVID ABRAHAM BEN [IL]) 30 December 2010 (2010-12-30) paragraphs [0030], [0127] - [0128]; figure 8	1-12
X	US 2007/050191 A1 (WEIDER CHRIS [US] ET AL) 1 March 2007 (2007-03-01) paragraphs [0011], [0013], [0027], [0028], [0042], [0043], [0048], [0117] - [0119], [0129], [0130], [0143], [0144], [0148]; figures 2,5 paragraphs [0188], [0196], [0211]	1-12
A	WO 99/01829 A1 (LERNOUT & HAUSPIE SPEECHPROD [BE]) 14 January 1999 (1999-01-14) abstract	1-12

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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

"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	Date of mailing of the international search report
30 September 2015	07/10/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Bowler, Alyssa

2

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/033481

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 8 165 886 B1 (GAGNON JEAN [CA] ET AL) 24 April 2012 (2012-04-24) abstract ----- US 6 665 640 B1 (BENNETT IAN M [US] ET AL) 16 December 2003 (2003-12-16) abstract -----	1-12
A		1-12

2

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2015/033481

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010332235 A1	30-12-2010	GB 2483814 A	21-03-2012
		US 2010332235 A1	30-12-2010
		WO 2011001370 A1	06-01-2011
US 2007050191 A1	01-03-2007	CN 101292282 A	22-10-2008
		EP 1929466 A2	11-06-2008
		EP 2325837 A2	25-05-2011
		US 2007050191 A1	01-03-2007
		US 2011231182 A1	22-09-2011
		US 2012278073 A1	01-11-2012
		US 2013253929 A1	26-09-2013
		US 2014365222 A1	11-12-2014
		WO 2007027546 A2	08-03-2007
WO 9901829 A1	14-01-1999	AT 223594 T	15-09-2002
		AU 732158 B2	12-04-2001
		AU 8237598 A	25-01-1999
		CA 2289066 A1	14-01-1999
		DE 69807699 D1	10-10-2002
		DE 69807699 T2	08-05-2003
		EP 0993640 A1	19-04-2000
		HK 1027406 A1	17-01-2003
		JP 2002507304 A	05-03-2002
		US 6138098 A	24-10-2000
		WO 9901829 A1	14-01-1999
US 8165886 B1	24-04-2012	US 8165886 B1	24-04-2012
		US 8543407 B1	24-09-2013
		US 8996375 B1	31-03-2015
US 6665640 B1	16-12-2003	NONE	