TITLE: Novel system for an intelligent web-based applications platform and personal assistant

INVENTOR(S): Khaled Talaat

BACKGROUND OF THE INVENTION:

This present invention is in the technical field of computer software. More particularly, this invention is in the technical field of web-based applications.

BRIEF SUMMARY OF THE INVENTION:

The present invention is a novel system and platform for a web-based, open applications platform that may be viewed from an end-user’s point of view as an advanced personal assistant. The system receives queries in natural language from clients connected to the server through a network and returns responses from remote third-party web services known to the server. The system allows users to build public and private applications onto the system. Applications allow the platform to interpret and analyze queries, communicate with remote web services to perform tasks and get information, and even decide how the output is to be displayed to the user. The system is beneficial for end-users as it is scalable and can interpret and execute complicated commands and queries that often can’t be analyzed by personal assistants that don’t allow user contribution. The system acts as an interpreter between end users and web services and operates in both learned natural language and registered commands. Novel algorithms are used to achieve a dynamic system that anybody can contribute to.

BRIEF DESCRIPTION OF THE DRAWINGS:

Fig.1 shows the different parties and systems involved

Fig.2 shows how different parties involved connect
Fig. 3 shows the main components of an app

Fig. 4 shows the systems in the Central Decision and Analysis System

Fig. 5 shows the stages that a query could go through

Fig. 6 shows how the system identifies the app that can perform the task

Fig. 7 shows how input analysis works

Fig. 8 shows how Input Processing works

Fig. 9 shows how the External Communications System works

Fig. 10 shows how Output Processing works

Fig. 11 shows how the sessions system works

Fig. 12 shows an exemplary user interface for an embodiment of the invention

DETAILED DESCRIPTION OF THE INVENTION:

Referring now to the invention in more detail, FIGS. 1-4 show the different components and elements of the system. These elements include the client 10, the server 11, and third party applications 19. The server consists of components found in typical general purpose computers like processor 12 and memory 13. Memory 13 in this context includes both temporary memory and permanent storage units found in general purpose computers.

Client 10 refers to the computers or devices that can communicate with the server 11. Clients 10 may connect to the server 11 through a browser or through an application programming interface. Instructions 14 refers to data such as code, form data, or database tables that guide the
central decision and analysis system 18 when necessary and introduce third-party applications 19 to the system. Instructions 14 may be registered directly by clients or crawled by the server 11, using the instruction crawling logic 15, from remote servers that contain the instructions. Instructions 14 are expected to contain certain information to enable the central decision and analysis system 18 to identify queries 600 that can be processed by the third-party application 19 associated with the instructions 14. Instructions 14 may contain information to allow the central decision and analysis system 18 to extract useful information and variables from the queries 600. Instructions 14 may also contain information to allow the central decision and analysis system 18 to make conclusions about the query 600. Instructions 14 may also contain information to allow the central decision and analysis system 18 to manipulate the query 600 or the extracted information and variables if necessary. Instructions 14 are expected to contain information that define how the server 11 could communicate with the remote third-party application 19 to send and receive data 20. Instructions 14 should define how the data 20 that the server 11 receives should be processed and displayed.

In short, the central decision and analysis system 18 relies on instructions 14 registered by clients 10 or crawled by the server 11 to process queries 600. These instructions 14 allow the server 11 to interpret, analyze, and modify queries 600 and communicate with appropriate third-party applications 19 to perform certain tasks or request data 20. They 14 also allow the server to process the output and display it or send it to the client 10. The instruction processing logic 16 stored on the memory 13 allows the server 11 to extract necessary information from the instructions 14 when needed in order to process queries 600. Instruction processing logic 16 also allows the server to inspect newly submitted instructions 14 to verify that they 14 agree with the guidelines. The memory 13 also stores the query processing logic 17 which enables the processor
to make use of the central decision and analysis system and inspected instructions to process queries.

Third-party applications are independent applications that the server communicates with to get data or perform tasks. Third-party applications are often hosted on remote servers and connect with the server through the internet using an application programming interface. The instructions combined with the third-party application are referred to as the “app” or “application.” Therefore, an app is defined as local instructions registered on the server or remote instructions stored on a remote server in addition to an external, integrated web-service for processing and storing data (“Third-party application”). Hence, the terms “app” and “instructions” may be used interchangeably herein knowing that the app spans the third-party application as well.

The central decision and analysis system refers to the logic and instructions, stored on the memory, that allow the server to understand, analyze, and outsource queries based on instructions. The central decision and analysis system consists of multiple systems that allow it to serve its purpose and to provide necessary tools to apps such as access to user data or access to a notifications system. The input interpretation and redirection system refers to the logic and instructions stored in the memory and spanned by the central decision and analysis system that allow the central decision and analysis system to determine which app can execute the query. The input interpretation and redirection system contains the logic to process the when-to-trigger instructions contained in apps. Based on the when-to-trigger instructions contained in apps, the input interpretation system decides which app can perform the task. The when-to-trigger instructions are also referred to as input interpretation instructions. Input interpretation instructions define criteria that query must meet in order to
be passed to app 30. The input interpretation and redirection system 405 and the input interpretation instructions are discussed in more detail later. The input analysis system 410 refers to the logic and instructions stored in the memory 13 and spanned by the central decision and analysis system 18 that allow the central decision and analysis system 18 to extract useful information from the query 600 according to input analysis instructions defined by app 30 that was found to be able to execute the query 600. The input and output programming systems 415 are systems that modify the inputs and the outputs according to programming instructions registered on apps 30. The external communications system 420 refers to the instructions spanned by the central decision and analysis system 18 and stored on the memory 13 that allow the server 11 to execute external communication instructions registered on apps 30. In other words, the external communications system 420 is the system responsible for communicating with third party applications 19 hosted on remote servers based on instructions registered on app 30. The output display system 425 is the system that controls how the output is displayed if requested by a web browser. The output display system 425 displays the output based on output display instructions registered on app 30 that spans the third party app 19 that sends the output. If no output display instructions are contained in app 30, the output display system may decide how the output is to be displayed on its own based on output display instructions from similar apps 30. The output display system 425 is unnecessary if the client 10 communicates with the server through an application programming interface. The session system 430 is a fairly simple system that allows apps 30 to register certain cookies in the client’s browser. These cookies may be used to put the user in a session skipping the input interpretation stage such that every query 600 the client 10 submits would go to app 30 that created the session until the session is destroyed. The output logging system 435 is the system responsible for caching results obtained from third party
applications 19 for a period of time to minimize the load on the server hosting the third party application 19. The output logging system 435 is controlled by instructions contained in apps 30 as apps 30 may decide what can be logged and cached as well as the rights that the server 11 has over the cached data. The user system 440 is the system responsible for both storing user data and sending necessary user data to apps 30 when the data is requested by the app 30. User data is defined here as: 1) Information given by the client’s browser such as IP address, browser, operating system, and screen resolution. 2) Information submitted directly by the client 10 such as queries 600, name of the user running the client if any, their age, and their education. 3) Data sent by apps 30 that is associated with a certain user such as files the user uploaded to an app’s servers. 4) Information on the client 10 that is determined by the server 11 such as the website making the request assuming that the client 10 in this case is associated with a website not a human user. Apps 30 may request user data through an application programming interface. The user system 440 is also the system that allows apps 30 to search for users who meet certain criteria. For instance, an app 30 may query the user system 440 to obtain 100 random IDs of users who study medicine for the purpose of sending them a notification or for the purpose of gathering data. The notifications system 445 is the system that allows apps 30 to send output to the client 10 regardless of prior action from the client 10. Authorized third party applications 19 may communicate with the notifications system 445 through an application programming interface to send notifications to the client 10 at any time. Notifications may take various shapes and forms. However, it is out of scope of this patent to discuss them in detail as the notifications system doesn’t play a critical role in the process.

Now referring to Fig.5, the figure describes the query processing logic 17. First, a client 10 submits a query 600 to the server 11. The query 600 is sent to the input interpretation and
redirection system 405 in a stage referred to as input interpretation and redirection 520. The purpose of the stage 520 is to determine which app 30 can perform the task in the query 600 or which app 30 the query 600 is meant to reach. The query 600 and the ID of the app 30 found in stage 520 are sent to the input analysis system 410 in a stage referred to as input analysis 520. The purpose of the input analysis stage 520 is to determine the important and variable information in a query 600. For instance, a query 600 could be “I am looking for a hotel in Paris from 1 Jan to 7 Jan.” The important information in this query 600 is the place, Paris, and the dates. This important information is referred to as input data. Therefore, input data is the variable or important information in a query 600 as defined by the app 30 that processes the query 600. Once the input data is determined, the app ID determined in stage 520, the input data and the query 600 are sent to the input programming system 415 in a stage known as input processing 540. The purpose of the input processing stage 540 is to perform conditional and non-conditional operations on the input data and to further analyze the query 600. The input processing stage 540 allows apps 30 to use the server 11 to perform calculations locally without having to send the input to a third party application 19 to perform the calculations. The app ID determined in stage 520, the query 600, the input data determined in stage 530, the new variables created in stage 540, and necessary information on the client 10 are all sent to the external communications system 420 in a stage known as external communication 550. The purpose of the external communication stage 550 is to send the inputs to the third party application 19 associated with the app 30 determined in stage 520 and to receive output from the third party application 19. To explain the role of third party applications, assume the client 10 submitted a query 600 like “I am looking for a hotel in Paris that costs less than $200 a day.” For the app 30 to return useful output, the app 30 would have to query a third party application 19 that is associated with it to
get data or perform further analysis of the query \texttt{600}. This communication between the server \texttt{11} and the third party app \texttt{19} is done in stage \texttt{550} based on instructions defined by app \texttt{30} that was found to be relevant to the query \texttt{600} in stage \texttt{520}. Once the third party application \texttt{19} sends the output, the output is sent to the output programming system \texttt{415} in a stage known as output processing \texttt{560}. The purpose of the output processing stage \texttt{560} is to alter the output if necessary according to app \texttt{30} instructions and produce front-end code to be sent to the output display system. The processed output is then sent to the output display system \texttt{425} in the output display stage \texttt{570}. The output display system \texttt{425} displays the output to the client \texttt{10} that submitted the query \texttt{600} based on instructions in app \texttt{30} that was found in stage \texttt{520}. The output is then logged in the output logging system \texttt{435} in the output logging stage \texttt{580}.

Now referring to Fig.6, the figure explains how the input interpretation and redirection system \texttt{415} works. The operations \texttt{600} to \texttt{655} aren’t necessarily in sequence. Understanding each operation is, however, necessary to understand how the system \texttt{415} operates. The purpose of the stage \texttt{520} was previously defined which was to determine the app \texttt{30} that is relevant to the query \texttt{600}. There might be more than one app \texttt{30} that is relevant to the query \texttt{600}. In case the user/client \texttt{10} hadn’t defined preferences in his account settings, the system \texttt{415} selects the first public app in the public apps database that can perform the task in the query \texttt{600}.

First, the system \texttt{415} receives a query \texttt{600}. This query \texttt{600} could be something like “Do I need to lose weight? I am 200 pounds and 6 ft 10 in tall.” Or something like “Street view: 300 W Bellows” Or even something like “Change my default search provider to Google.” A query \texttt{600} may as well include an attached file such as “Send this file to <phone number>.” Ideally, the system \texttt{415} would give priority to system apps in its search for apps that can handle the query. System apps are apps that are built into the system and perform operations that normal apps \texttt{30}
may not be allowed to perform such as resetting the user’s password, changing the email associated with the account, or changing certain privacy settings. System apps, like public apps, must have trigger conditions. Trigger conditions could be keywords that must exist in the query, keywords that may exist in the query, word count of the query, phrases that must exist, phrases that may exist, words that must not exist, phrases that must not exist, existence of an attached file, mention of a web address, mention of a phone number, mention of an email address, mention of a user, mention of a certain pattern of letters and numbers such as an IP address or a mail tracking ID, mention of certain words at certain positions in the query, or mention of equations. The query must satisfy a certain percent of the trigger conditions to be considered relevant to the app. This percent is defined by the app. If the system 415 determines in 605 that a system app can handle the query 600, the query gets sent to system command execution 610. If no system app can handle the query 600, the system 415 checks if there’s an account-linked app, such as an app that the client 10 prefers or had used before, that is relevant to the query 600. If the system 415 finds an app 30 that is relevant to the query 600 in 615, the query and app id are sent to input analysis 530. If no account-linked app can handle the query 600, the system 415 checks if a public app can perform the task 615 the same way it checks system apps and account-linked apps. If no public app is found to be able to handle the query 600, the system 415 determines whether or not the query has a recognizable pattern 635. For instance, if the client 10 submits a query 600 like “(989) 123 4567” or “40012345678”” It is difficult to determine the app 30 that can handle the query 600 by the trigger conditions approach used in 605 to 625 because the query lacks keywords and information that would allow the system 415 to determine the app with a reasonable degree of certainty. Therefore, such queries are dealt with uniquely in 640 when the query 600 is found to contain a known pattern registered by an app 30. Mathematical
queries are also hard to identify using the trigger conditions approach. Mathematical queries don’t often have a certain pattern either. Mathematical queries could be arithmetic or algebraic. Non-algebraic, mathematical queries can be efficiently detected by testing through performing the calculation on the server 11 and checking the result. Algebraic queries may be detected through calculating the abundance of certain letters, checking the query against a dictionary, and existence of arithmetic operators. If an algebraic query is detected, the query is then sent to the relevant app that is known to handle such queries. If 645 determines that the query 600 is not a mathematical problem, the query 600 is sent to web search 655, and web search results will be sent to the client 10.

Now referring to Fig.7, the figure explains how the input analysis system 410 works. First the input analysis system 410 receives the query 600 and the input interpretation results. The system 410 then separates the words in the query 600 and stores the order of each word (process 700). The system 410 relies on input analysis instructions registered on the app 30 that was found to be relevant to the query 600 to perform natural language analysis of the query 600. Apps 30 that use the input analysis system 410 are required to define critical words. Critical words are words that define the location of the important variables in the query. For example, consider this query “What is the BMI for somebody who is 6 ft tall and weighs 200 lb.” The words “ft” and “lb” in this example are critical words that are pre-defined by the app 30 that is relevant to the query. The query in the example could be written in many different ways. However, the position of the important variables will always be related to the position of the critical words that exist regardless of how the query is phrased. In the input processing instructions, apps 30 define the critical words, their alternatives, the position of the input data with respect to the critical words, the type of the input data, words that shouldn’t exist in the query, and whether or not the critical
word must exist. The critical word could either be a substring or a word. The critical word may have alternatives in case it has synonyms. Some critical words may not be very important as they define variables that give additional, unnecessary information. Apps 30 that use the input analysis system 410 must define what critical words are required to be present in the query 600 and what aren’t. Apps 30 should also define the action that should take place in case the critical word isn’t present in the query 600. One possible action is requesting missing data 740. The system requests data input from the user in order to proceed with the analysis. Another possible action is terminating the analysis and sending the query 600 back to input interpretation and redirection 520 indicating that the query 600 isn’t relevant to the app 30. Furthermore, apps 30 are required to define the position of the input data with respect to the critical words. Input data could be located before or after or generally near the critical word. Apps 30 are also required to define the input data type. The term “input data” refers to variable, important data associated with a critical word in the query 600 submitted by the client 10. Input data could be a word, a sentence, a paragraph, a number, a URL(s), a phone number, a location, a date, an email, a person, or even an array of numbers. The input analysis system 410 uses instructions on the memory 13 to extract the input data from the query 600 based on the input analysis instructions registered on app 30 (process 710). The system then corrects the input data based on data type 720. For instance, if the input data type is date, the system recognizes the date associated with the critical word regardless of how it is formatted in the query 600 and returns the date in a standard format that the third party application 19 associated with the app 30 can understand. The process of recognizing entities associated with critical words and returning them in standard formats is referred to here as correction (process 720). For certain input data types, such as paragraph and word, the system may need to eliminate undesired words that are likely to exist
between the critical word and the input data (process 730). For instance, “Save: I am a pizza” could be written as “Save this: I am a pizza.” If the critical word is set to “save,” the words “this” and the colon should be listed as undesired words. Undesired words are words that might exist between the critical data and the input data in the query but are not desired. These words are eliminated from the query 600. Once the system determines the input data for all the critical words registered by the app 30, the system sends the input data in a standard format to the input processing system (process 750).

Now referring to Fig.8, the figure explains how input processing 540 works. The purpose of the input processing stage 540 is to perform further analysis on the query 600 and the input data determined in stage 530, modify the input data, and define what variables to be sent to the third party application 19. The analysis in stage 530 is based on critical words. While critical words are useful for extracting variable information from the query 600, they are not as useful when it comes down to static information. For example, in this query, “I am looking for a pet friendly hotel in New York that is cheap.” It is not possible to conclude that the user is looking for a hotel that is both pet friendly and cheap using critical word analysis as in stage 530. Further analysis of the query 600 becomes necessary. Further analysis of the query could include scanning the query 600 for certain words and performing conditional operations. It could also include scanning the whole query 600 for certain types of entities, returning the entities, and returning their count.

Input processing operations include performing conditional and non-conditional arithmetic and advanced mathematical operations on the input data extracted from the query 600 in case the input data is numerical, and performing textual operations such as concatenation and text modification in case the input data is textual. Other operations might be possible depending on the type of the input data.
First, the input processing system, also referred to as the input programming system 415, receives the query 600, the input data, and the ID of the app 30 that can run the query 600 (process 800). The system 415 then reads the processing instructions registered on the app 30 (process 810). The system then analyzes the query 600 according to the instructions registered on app 30. The results of the analysis are stored on variables defined in the instructions given by the app 30. The system then performs both conditional and non-conditional operations on the input data and on newly defined variables. The system may alter existing variables or generate new variables according to the app 30 instructions (process 820). The system then queries the user system for user information requested by the app such as user location, recent history, interests, age, education, etc. (process 830) given that the app 30 has the authority to request such data. The system then sends the processed variables, query 600, user info, and app ID to the external communications system 420 (process 840).

Now referring to FIGS 9-10, figure 9 explains how external communication 550 works. Figure 10 explains generally how the output processing 560 works. External communication 550 is the process where data is exchanged between the server 11 and third party application(s) 19 associated with apps 30. The purpose of external communication 550 is to allow apps 30 to execute complicated queries 600 relying on their own resources. First, the server 11 sends the query 600, processed variables, and user information to the third party application defined in the app instructions (process 900). The server sends the data according to the app 30 instructions which define the method the server 11 should use to reach the third party application 19 as well as the data format required. The app 30 is expected to contain instructions and information that enable the server 11 to communicate with the third party application 19 through an application programming interface. Typical data formats are JSON and XML. However, any data format
could be supported and used. After the third party application 19 receives the request and the
data, the third party application typically processes the request and returns a response to the
server 11. The server 11 receives the response and decodes it to determine the status of the
response (processes 910 and 920). Third party applications 19 are expected to return a status that
indicates whether or not they were able to perform the operation. If the status is negative, the
query 600 is sent back to input interpretation and redirection 520 (process 930). If the status is
positive or if no status is given, the response is sent to output processing 560 (process 940). The
purpose of output processing 560 is to verify that the response is not malicious, alter the output if
necessary, and produce output display code. First, the output processing system, also referred to
as output programming system 415, receives the response and checks it’s in valid format
(process 1000). The response is then decoded according to instructions registered on the app 30
associated with the third party app 19, and variables are extracted (process 1010). The system
then verifies that the response is not malicious by looking for code that violates the terms of
service (process 1020). Similar to input processing 540, the output variables may be altered, new
variables may be created, loops may be supported, and conditional and non-conditional
operations could be performed. The system alters the decoded response according to app
instructions (process 1030). The system then produces output display code according to the app
instructions (process 1040). Output display code is mark-up code that is readable by the output
display system 425. The output display system 425 is the system that communicates directly with
the client 10. It 425 may convert output display mark-up to widely supported languages
depending on how the client 10 is communicating with the server 11 and according to the app 30
instructions. If the client 10 is making an API request, the output display system would typically
not return a response that contains front-end code such as JavaScript to the client 10. However, if
the client 10 is communicating with the server 11 through a browser, then it makes sense for the
app 30 to instruct the system 425 to return a response that contains widely-recognized front-end
code. The output display code is generated according to the app instructions. It defines what code
and data should be sent to the client 10 through the output display system 425. The output
display code is sent to the output display system (process 1050). The code is then converted
according to system instructions to other languages and sent to the client 10 along with session
data. The output display code may then be cached in the memory 13 by the output logging
system 435 to provide faster responses to similar queries 600 if permitted by the app 30.

Now referring to Fig.11, the figure explains how the session system 430 works. The purpose of
the session system 430 is to allow apps 30 to lock the client 10 in a session. Once the client 10 is
locked in a session, all queries 600 submitted by the client 10 are sent to the app 30 that owns the
session until the session is terminated by either the app 30 or the client 10. In case the client 10 is
connected through an application programming interface, it’s up to the client 10 to accept the
session or ignore it.

The client 10 might communicate with the server 11 through a web browser or through an
application programming interface without a web browser. The session process starts when the
third party application 19 sends a response to a query 10 submitted by the client 10 (process
1200). The response may contain certain code that instructs the server 11 to lock the client 10 in
a session with the app 30. The server 11 fetches for the code once the response is received
(process 1210). In case the client 10 is connected to the server through a browser, the system
stores a cookie or similar technology onto the client’s browser (process 1220). The cookie or its
alternative might contain the app id and a key to specify the app 30 that owns the session. In case
the client 10 is connected to the server 11 through an application programming interface, the
server 11 may return a variable to the client 10 that defines whether or not the app 30 requested a session. Once the client 10 submits another query 600 to the server 11 (process 1230), the server 11 checks whether or not the client is locked in a session with a specific app 30 (process 1240). In case the client 10 is using a browser, the server 11 checks the cookie sent by the client’s browser when making the request. In case the client 10 makes the request to the server 11 through an application programming interface, it’s the client’s responsibility to declare a session in the query 600. The client 10 in that case is required to let the server know that the query 600 is meant for a specific app 30 and not a general query. In both cases, the system may check the query 600 to determine if it indicates intention to end the session with the app (process 1250). If it doesn’t, the system sends the query 600 straight to the defaulted third party application 19 associated with the app 30 that owns the session (process 1270). If the query 600 indicates intention to end the session, the system terminates the session with the app 30 and sends the query 600 to the input interpretation and redirection system 405. The whole process is repeated as the app 30 responds to the query 600 and as the client 10 submits other queries 600.

Now referring to Fig.12, the figure shows one embodiment of the invention: a web-based operating system. The web-based operating system shown in the figure could be browser-based or boot-loaded. The interface consists of a text area where the user could enter queries 600, a menu, and some room for the results. The system looks for apps registered on its servers 11 that could perform the query 600. The query 600 is processed according to Figure 5 as discussed earlier. The output is then displayed to the user. Results for different queries may be shown on the same screen as shown in the figure which gives room to multi-tasking. App 30 developers may run their own advertisements in the results or allow the system to run relevant advertisements and share revenue with the organization that administrates the operating system.
In another embodiment of the invention, the invention could be used to create an advanced API to API platform where one party makes a request to the server 11 in natural language and the server 11 looks for another party that can process the request. The server 11 relies on instructions registered by third parties on the memory to determine what third party application 19 can process the request. The server analyses the query 600 in the request according to instructions registered by or associated with the third party application 19 that was found to be able to process the request and performs any necessary calculations or adjustments. The server 11 then communicates with the third party application 19 and returns the response to the party that made the request according to instructions registered by the third party application 19 that responded to the query 600. Such system would bring an end to the API integration hassle often faced by developers.

CLAIMS:

1.

ABSTRACT:

This system is a novel system for a user-managed, scalable web-based personal assistant and applications platform. The system allows users to build public and private applications (“apps”) onto the system that can communicate with remote web services to perform tasks or transfer information. The system gives developers access to an advanced input interpretation system, a dynamic input analysis system, and an input processing system that dramatically minimize the
coding requirements for building a service that can understand, analyze, and execute complicated commands. The system receives a command from an end-user. It then interprets the command and searches for a registered or a crawled app that can perform the task or deliver the information requested. The system then analyzes the command according to instructions defined by the app to determine the useful inputs (“Input Data”) in the command/query. Inputs may then be processed according to instructions defined by the app. The useful inputs are sent to the external/remote web service associated with the identified app. The system then receives the output(s) from the remote web service, modifies the output(s) according to the app instructions, and displays the final output to the end-user according to the app instructions. The end-user isn’t necessarily a human being. It could be another web service making an API request or a robot connected to the network.